

**University of BLIDA 1**  
Faculty of Science  
Computer science department



**MASTER THESIS**

**Option: Ingénierie des logiciels**

# **Anomaly detection in power line signals**

By:

Rouizi Sarah

Zella Amalia

**In front of jury composed of:**

|                        |            |
|------------------------|------------|
| Ms. Midoun Khadidja    | President  |
| Mr. Khediri Abderrazak | Examiner   |
| Ms. YKHLEF Hadjer      | Supervisor |
| Mr. YKHLEF Farid       | Supervisor |

2021/2022

# Abstract

Large companies need to integrate a full fault detection system into their industry, especially electric companies. Consequently, automatic fault detection has become of utmost importance to assess the condition of electrical components and has been a long-standing challenge. Partial discharge is a common indication of faults in power systems, such as generators and cables. These PDs can eventually result in costly repairs and substantial power outages. The main goal of this work consists of devising a monitoring system capable of detecting partial discharge patterns in signals acquired from power lines. To accomplish this task, we have built and compared several detectors trained on hand-crafted features extracted using basic statistics and signal processing-based methods. These features are fed to a learning module; we have explored two kinds of learning paradigms: Sequential Learning and Ensemble-based Learning. Specifically, we have invoked four deep sequential models: Long-Short-Term-Memory (LSTM), Bidirectional LSTM (Bi-LSTM), Gated Recurrent Unit (GRU), Recurrent Neural Network (RNN), and four ensemble learning approaches namely: AdaBoost, Random Forest, LightGBM, and XGBoost. We have conducted our experiments on VSB power-line-faults-detection challenge dataset. Our experimental findings indicate that signal processing-based features do not improve the performance and that efficiently preprocessing signals achieves state-of-the-art performance compared to other features extraction techniques. In addition, we have found that gated-based models (LSTMs and GRUs) have shown good results. Moreover, our analysis reveals that the gradient boosting machine (LightGBM, XGBoost) exhibits high scores compared with the other learning models. Most importantly, in order to build a **successful fault detector, the focus has to shift from model-centric to data-centric**, i.e. understand the input training data and preprocess it.

**Keywords: Anomaly detection, Partial discharge, Ensemble learning, Long Short Term Memory, Signal Processing.**

## Résumé

Les grandes entreprises ont besoin d'intégrer un système complet de détection des défauts dans leur secteur, notamment les entreprises qui travaillent dans le domaine électrique. Par conséquent, la détection automatique des fautes est devenue de la plus haute importance pour évaluer l'état des composants électriques et constitue un défi de longue date. Les décharges partielles sont une indication courante de défauts dans les systèmes électriques, tels que les générateurs et les câbles. Ces décharges partielles peuvent éventuellement entraîner des dégâts coûteux et des coupures de courant importantes. L'objectif principal de ce travail consiste à concevoir un système de surveillance capable de détecter la décharge partielle dans les signaux acquis à partir des lignes électriques. Pour accomplir cette tâche, nous avons construit et comparé plusieurs détecteurs formés à partir de caractéristiques extraites à l'aide des statistiques et des méthodes basées sur le traitement du signal. Ces caractéristiques sont transmises à un modèle d'apprentissage ; nous avons exploré deux types de paradigmes d'apprentissage: L'apprentissage séquentiel et l'apprentissage d'ensemble. Plus précisément, nous avons fait appel à quatre modèles séquentiels profonds: réseaux de neurones récurrents RNN, LSTM, Bi-LSTM et GRU; et quatre modèles d'apprentissage supervisé, à savoir : AdaBoost, Forêt aléatoire, LightGBM et XGBoost. Nous avons mené nos expériences sur l'ensemble des données VSB de la compétition « détection des défauts de lignes électriques ». Nos résultats expérimentaux indiquent que les caractéristiques basées sur le traitement des signaux n'améliorent pas les performances et qu'un prétraitement efficace des signaux permet d'obtenir de meilleures performances par rapport aux autres techniques d'extraction de caractéristiques. En outre, nous avons constaté que les modèles basés sur les portes (LSTMs et GRUs) ont donné de bons résultats. De plus, notre analyse a révélé que le gradient boosting machine (LightGBM, XGBoost) présente des scores élevés par rapport aux autres paradigmes d'apprentissage. Plus important encore, pour construire un détecteur de fautes efficace, il faut passer de l'approche model-centric à une approche data-centric, c'est-à-dire comprendre les données d'apprentissage en entrée et les prétraiter.

**Mots clés : Détection d'anomalies, Décharge partielle, Apprentissage ensembliste, Réseau de Neurones Récurrents, Traitement de signal.**

## ملخص

تحتاج الشركات الكبيرة إلى دمج نظام اكتشاف الأعطال الكامل في صناعتها ، وخاصة شركات الكهرباء .وبالتالي ، أصبح الاكتشاف التلقائي للأعطال ذا أهمية قصوى لتقييم حالة المكونات الكهربائية وكان يمثل تحديًا طويل الأمد. التفريغ الجزئي هو مؤشر شائع لأعطال أنظمة الطاقة ، مثل المولدات والكابلات. يمكن أن تؤدي هذه الحالات الشاذة في النهاية إلى إصلاحات مكلفة وانقطاع كبير للتيار الكهربائي. يتمثل الهدف الرئيسي لهذا العمل في تصميم نظام مراقبة قادر على اكتشاف أنماط التفريغ الجزئي في الإشارات المكتسبة من خطوط الكهرباء. لإنجاز هذه المهمة ، قمنا ببناء ومقارنة العديد من أجهزة الكشف المدربة على الميزات المصنوعة يدويًا المستخرجة باستخدام الإحصائيات الأساسية والأساليب القائمة على معالجة الإشارات. يتم تغذية هذه الميزات إلى وحدة التعلم ؛ لقد استعملنا نوعين من نماذج التعلم: التعلم المتسلسل والتعلم القائم على المجموعة. على وجه التحديد استخدمنا أربعة نماذج متسلسلة عميقة: الذاكرة طويلة المدى قصيرة المدى LSTM، الذاكرة طويلة المدى قصيرة المدى ثنائية الاتجاه Bi-LSTM، الوحدة المتكررة ذات البوابات GRU، الشبكة العصبية المتكررة RNN، وأربعة مناهج تعلم المجموعات وهي AdaBoost و Random Forest و LightGBM و XGBoost. لقد أجرينا تجاربنا على مجموعة بيانات تحدي اكتشاف أعطال خط الطاقة VSB. تشير النتائج التجريبية التي توصلنا إليها إلى أن الميزات القائمة على معالجة الإشارات لا تعمل على تحسين الأداء وأن إشارات المعالجة المسبقة بكفاءة تحقق أداءً متطورًا مقارنةً بتقنيات استخراج الميزات الأخرى. بالإضافة إلى ذلك ، وجدنا أن النماذج المبوبة LSTMs و GRUs أظهرت نتائج جيدة. علاوة على ذلك ، يكشف تحليلنا أن آلة تعزيز التدرج LightGBM ، XGBoost تظهر درجات عالية مقارنةً بنماذج التعلم الأخرى. الأهم من ذلك ، من أجل بناء كاشف خطأ ناجح ، يجب أن ينتقل التركيز من النموذج إلى التركيز على البيانات ، أي فهم بيانات تدريب الإدخال ومعالجتها مسبقًا.

كلمات المفاتيح : التفريغ الجزئي , مجموعة التعلم, الذاكرة طويلة المدى قصيرة المدى , اكتشاف الاعطال , معالجة الاشارات .

# **Acknowledgments**

We would first like to thank our thesis supervisors Ms YKHLEF Hadjer and Mr YKHLEF Farid. Their offices were always open whenever we ran into a trouble spot or had a question about our research or writing. They consistently allowed this project to be our own work, but steered us in the right direction whenever they thought we needed it. Special thanks goes to the examiners for the time they spent on carefully reviewing this thesis. In addition, we wish to extend our thanks to all of my professors for sharing their invaluable knowledge. Finally, many thanks goes to all the people who have contributed to the completion of our research work directly or indirectly.

# Contents

|   |           |
|---|-----------|
| <b>Introduction .....</b>   | <b>1</b>  |
| 1. Background and problem definition .....                                  | 1         |
| 2. Thesis contributions .....   | 3         |
| 3. Thesis organization .....  | 3         |
| <b>Chapter 1 : Generalities on Partial Discharge .....</b>                  | <b>4</b>  |
| 1.1 Introduction .....  | 4         |
| 1.2 Partial discharge .....   | 4         |
| 1.3 Electrical signal .....   | 5         |
| 1.4 Development pipeline for partial discharge detection .....              | 6         |
| 1.5 Feature engineering .....   | 7         |
| 1.6 Temporal domain representation vs frequency domain representation ..... | 11        |
| 1.7 Conclusion .....  | 12        |
| <b>Chapter 2 : Machine Learning Fundamentals .....</b>                      | <b>13</b> |
| 2.1 Introduction .....  | 13        |
| 2.2 Generalities on machine learning .....                                  | 13        |
| 2.3 Anomaly detection.....  | 16        |
| 2.4 Common machine learning classifiers.....                                | 17        |
| 2.5 Deep learning for sequential models .....                               | 22        |
| 2.6 Conclusion .....  | 33        |
| <b>Chapter 3 : Experimental Framework .....</b>                             | <b>34</b> |
| 3.1 Introduction .....  | 34        |
| 3.2 Utility libraries .....   | 34        |
| 3.3 Training data-Set .....   | 35        |
| 3.4 Features engineering.....   | 36        |
| 3.5 Learning models .....   | 37        |
| 3.6 Performance evaluation .....  | 39        |
| 3.7 Conclusion .....  | 40        |
| <b>Chapter 4 : Experimental Results and Discussion.....</b>                 | <b>41</b> |
| 4.1 Introduction .....  | 41        |
| 4.2 Experiment set 1: Sequential modeling .....                             | 41        |

|                         |   |           |
|-------------------------|---|-----------|
| 4.3                     | Experiment set 2: Traditional machine learning models ..... | 49        |
| 4.4                     | Summary of experimental findings .....                      | 52        |
| 4.5                     | Chapter summary.....  | 53        |
| <b>Conclusion</b> ..... |   | <b>54</b> |
| 1.                      | Contributions and summary of experimental findings.....     | 54        |
| 2.                      | Limits and future work.....                                 | 54        |
| <b>References</b> ..... |   | <b>56</b> |

## List of Figures

|  |    |
|--|----|
| Figure 1.1: Periodic signal representation.....  | 6  |
| Figure 1.2: Continuous vs Discrete Signals. ....   | 6  |
| Figure 1.3: Development pipeline for partial discharge detection. ....   | 7  |
| Figure 1.4: Max values for anomalous/non-anomalous signals. ....   | 9  |
| Figure 1.5: Min values for PD/non-PD training signals. ....  | 9  |
| Figure 1.6: Average of PD/non-PD training signals.....   | 10 |
| Figure 1.7: Standard-deviation of PD/non-PD training signals. ....   | 10 |
| Figure 1.8:A signal (black) consisting of multiple component signals (blue) with different frequencies (red). .... | 12 |
| Figure 2.1:Confusion matrix [50]......   | 15 |
| Figure 2.2: Cross validation [51]......  | 16 |
| Figure 2.3: ANN architecture.....  | 23 |
| Figure 2.4: Perceptron.....  | 23 |
| Figure 2.5:Gradient descent graph.....   | 26 |
| Figure 2.6: Recurrent neural network. ....   | 26 |
| Figure 2.7: RNN architecture. ....   | 27 |
| Figure 2.8: RNNs types and applications.....   | 27 |
| Figure 2.9: Self-loop LSTM cell. ....  | 28 |
| Figure 2.10: Gate Form. ....   | 28 |
| Figure 2.11: LSTM Architecture.....  | 29 |
| Figure 2.12: Bidirectional-LSTM architecture [74]. ....  | 31 |
| Figure 2.13:GRU architecture diagram.....  | 31 |
| Figure 2.14: Deep sequential model. ....   | 32 |
| Figure 2.15: Attention layer [80]......  | 33 |
| Figure 3.1: Metadata of training set.....  | 36 |
| Figure 3.2: Three-phase signal Smoothed with 1D convolution. ....  | 37 |
| Figure 3.3: Sequential model implementation architecture. ....   | 38 |
| Figure 3.4: training traditional models process.....   | 39 |
| Figure 4.1: LSTM vs Bi-LSTM vs RNN vs GRU Wicker plot representation.....  | 41 |
| Figure 4.2: multiple stacked-layers models bar-plots.....  | 42 |
| Figure 4.3: Attention impact on Bi-LSTM variants. ....   | 44 |
| Figure 4.4: Attention impact on GRU variants. ....   | 44 |
| Figure 4.5: MCC scores for Bi-LSTM2 model trained on multiple feature sets. ....                                   | 46 |
| Figure 4.6: MCC scores for GRU2 models trained on multiple feature sets. ....                                      | 46 |
| Figure 4.7:MCC Score for AdaBoost and Random Forest. ....  | 50 |
| Figure 4.8: MCC Score for LightGBM and XGBoost. ....   | 50 |
| Figure 4.9: MCC Score for all classifiers trained on preprocessing and FFT+Stat. ....                              | 50 |



## List of Tables

|  |    |
|--|----|
| Table 2.1: Activation functions formulas. ....   | 24 |
| Table 3.1: Traditional models parameters set. ....                                       | 39 |
| Table 4.1: MCC score (%) of the gated memory systems. ....                               | 41 |
| Table 4.2: MCC score (%) of Bi-LSTM/GRU for multiple stacked layers.....                 | 42 |
| Table 4.3: MCC score (%) of sequential systems with/without Attention mechanism. ....    | 43 |
| Table 4.4: MCC results (%) of Bi-LSTM/GRU trained for 200 epochs. ....                   | 45 |
| Table 4.5: MCC score (%) of deep systems tested on different sets of features.....       | 46 |
| Table 4.6: Submission scores (%) of four sequential systems with Stat features. ....     | 47 |
| Table 4.7: Submission scores (%) of four sequential systems with FFT features.....       | 48 |
| Table 4.8: Submission scores (%) of four sequential systems with Stat+FFT features. .... | 48 |
| Table 4.9: Submission scores (%) of combined systems. ....                               | 49 |
| Table 4.10: MCC scores (%) of Boosting machines. ....                                    | 49 |
| Table 4.11: Submission scores (%) of AdaBoost with four different feature sets.....      | 51 |
| Table 4.12: Submission scores (%) of Random Forest with four different feature sets..... | 52 |
| Table 4.13 :Submission scores (%) of LightGBM with four different feature sets. ....     | 52 |
| Table 4.14: Submission scores (%) of XGBoost with four different feature sets.....       | 52 |
| Table 4.15: Top 3 submitted scores.....  | 52 |

## **Notation and Acronyms**

|                   |   |
|-------------------|---|
| <b>PD :</b>       | <b>Partial Discharge</b>                |
| <b>RNN :</b>      | <b>Recurrent neural networks</b>        |
| <b>LSTM :</b>     | <b>Long Short-Term Memory</b>           |
| <b>GRU :</b>      | <b>Gated Recurrent Unit</b>             |
| <b>FFT:</b>       | <b>Fast Fourier Transform</b>           |
| <b>PSD:</b>       | <b>Power Spectral Density</b>           |
| <b>MCC :</b>      | <b>Matthews Correlation Coefficient</b> |
| <b>GD :</b>       | <b>Gradient Descent</b>                 |
| <b>BPTT:</b>      | <b>Back Propagation Through Time</b>    |
| <b>GBM:</b>       | <b>Gradient Boosting Machine</b>        |
| <b>ANN :</b>      | <b>Artificial Neural Network</b>        |
| <b>XGBOOST :</b>  | <b>Extreme Gradient Boosting</b>        |
| <b>LightGBM :</b> | <b>Light Gradient Boosting Machine</b>  |
| <b>Adam :</b>     | <b>Adaptive Moment Estimation</b>       |

# Introduction

## 1. Background and problem definition

Researchers have been paying close attention to the rise of Industry 5.0 [1]. It is concerned with the fifth industrial revolution in manufacturing, in which technologies such as automation and artificial intelligence are completely altering traditional manufacturing processes. The main goal of Industry 5.0 is to strike a balance whereby machine-human interaction can offer the highest benefits. Industrial machinery and heavy equipment manufacturing is becoming more complex with the increasing requirements of embedded software and electronics content. Many manufacturers are challenged with these new and ever-changing needs. Industrial equipment generates various types of time-series data, including power-line signals, heart rate monitoring (EKG), brain monitoring (EEG), sensor data like Weather data and Rainfall measurements, Temperature readings, and even sounds signals. As such data continues to increase, analyzing it and identifying patterns from it can become a challenging task, which requires an adequate treatment [2]. This problem has led to the introduction of machine learning to extract information from data that would otherwise be impossible to obtain even by human experts [3]. It has successfully been applied to a number of problems, including speech enhancement [4], acoustic sound [5], many other issues or even fault detection in power line signals [6] which is our interest.

Anomaly detection, also known as fault detection, aims to discover unexpected events or rare items in data[7]. It is popular in many industrial applications and is an important research area in data mining [8]. Accurate anomaly detection can trigger prompt troubleshooting, help to avoid loss in revenue, and maintain the reputation and branding for a company. When anomalies are detected, alerts will be sent to the operators to make timely decisions related to incidents.

One of the most common challenge is to detect partial discharge in the overhead power lines. *So how can we define a partial discharge ?* The overhead power lines transfer power from one region to another over hundreds of kilometers. These distances make it difficult and expensive to manually inspect for any damages caused to the power lines. These damages lead to a phenomenon called **Partial Discharge (PD)** in the insulators of the power line [9]. Basically, partial discharge is an electrical discharge that does not bridge the electrodes between the insulation systems [9].

In fact, the presence of partial discharge can be indicative of anomalies in many electrical systems and can cause further degradation of the insulation. The high-voltage discharges deteriorate the insulation materials and can have impacts on the entire system. The PD will slowly damage the power line if it is not detected and repaired promptly, causing power outages or even a fire. Their detection is, therefore, of utmost importance to assess the condition of electrical components and has

been a long-standing challenge [10]. As such, the literature is extremely vast. PD detection has been studied in many systems such as in transformers [11], gas-insulated high-voltage switchgear [12], power plants [13], and power lines [14].

The overarching goal of this thesis is to find or construct a model for anomaly detection and evaluate its applicability in detecting Partial Discharge in power lines signals. Anomaly detection falls into two main categories: Unsupervised and Supervised Anomaly Detection. Unsupervised anomaly detection techniques do not need training data. Alternatively, it is the most flexible setup which does not require any labels, whereas Supervised Anomaly Detection, in which supervised methods (also known as classification methods) requires a labeled training set containing both normal and anomalous samples to construct the predictive models. Since our training set comes labeled, we will focus only on Supervised Anomaly Detection.

The general framework for building a partial discharge detector is usually categorized into two major steps. (1) Feature extraction: a representation of power signals is obtained via multiple signal processing-based methods or basic statistics like the mean and the standard deviation. This representation is then fed to the model for the training process. (2) Machine Learning: the extracted features are employed to achieve recognition using a classification model. We can distinguish two types of modeling namely the sequential models and traditional machine learning techniques. Sequence Modeling is a technique in machine learning that is used for analyzing sequence data. It is the task of predicting what data is coming next. In sequence modeling, the current output is dependent on the previous input and the length of the input is not fixed. Sequence models can be used in different applications such as image captioning, smart replies on chat tools etc. Multiple traditional machine learning models can be used for anomaly detection, since they provide a simple learning process. (ML) methods contributed highly in the advancement of prediction systems providing better performance and cost-effective solutions. Due to the vast benefits and potential of ML. These two categories of learning models have a different implementation and they are distinct from each other in some points: Machine learning programs tend to be less complex than sequential modes and can often run on conventional computers, but deep learning systems require far more powerful hardware and resources. Traditional systems can be set up and operate quickly but may be limited in the power of their results. Whereas sequential systems take more time to set up but can generate results instantaneously.

Many developers have tackled this challenge. Ming Dong [15] developed a unique method based on Seasonal and Trend decomposition using Loess (STL) and Support Vector Machine (SVM) to recognize PD activities on insulated overhead conductors. Different SVM kernels were tested and compared. Also Gabriel Michau and Chi-Ching Hsu proposed a novel end-to-end framework based on convolutional neural networks [16]. They implemented a framework with two contributions: First, it does not require any feature extraction and enables robust partial discharge detection.

Second, it devises the pulse activation map. It provides interpretability of the results for the domain experts with the identification of the pulses that led to the detection of the PDs. There exists many other challengers who have implemented different strategies in the VSB Power line challenge that main to detect the presence of the PD [17].

## 2. Thesis contributions

The main goal of our work consists of devising a monitoring system capable of detecting partial discharge patterns in signals acquired from power lines. To accomplish this task, we formulate the partial discharge detection as a binary classification problem. Given an input signal, the output of the detector is restricted to two classes: 1 or 0; if the output is 1, it indicates the presence of a partial discharge, 0 indicates no partial discharge. This problem can be tackled using a classification technique that is able to learn a mapping from the input signal to the output, its associated class. In our work, we have devised several detectors using various well-known ensemble learning technique, namely: AdaBoost, Random Forest and two variants of the successful Gradient Boosting Machine LightGBM and XGBoost. In addition, since the data of interest comprises signals that evolve over time, sequential models like RNNs and LSTMs can be also used to design fault detectors. We have specifically explored four different architectures: RNN, LSTM, GRU and Bi-LSTM.

The contributions of this work are highlighted as follows:

1. We extract a variety of features using basic statistical measures and multiple signal processing techniques.
2. We explore several memory unit-based systems LSTMs, GRUs, Bi-LSTMs, while varying some hyper-parameters like the number of stacked layers, the attention mechanism, and analyze the behavior of these architectures.
3. We invoke diverse ensemble learning algorithms and compare their detection scores.

## 3. Thesis organization

The thesis is organized as follows. In **Chapter 1**, we cover some fundamentals behind partial discharge. Specifically, we describe various representations of power lines signals and the most frequently used feature extraction techniques in literature. In **Chapter 2**, we review some relevant classification concepts, providing a brief description of machine learning models, state-of-the-art deep learning architectures along with the evaluation metrics invoked in this work. We provide in **Chapter 3**, a detailed description of the experimental setup, hyper-parameters tuning and evaluation procedure that we have used. Finally in the last chapter, **Chapter 4**, we lay out the obtained results through performance tables and statistics-based plots and discuss these findings. To conclude we summarized the contributions of this thesis, the lines of limitations and future work.

# Chapter 1 : Generalities on Partial Discharge

## 1.1 Introduction

Many important services in our society, including healthcare, transportation and security, require a robust, reliable and undisrupted supply of electricity. This requires on the one hand a reliable and redundant infrastructure, and on the other hand, the ability to maintain the performance of the infrastructure. As such, the ability to detect faults. Many components of the power generation and distribution network can be directly monitored with specific sensors. However, it is not possible nor cost efficient for all the components and all the fault types. Therefore, for some of the components, the monitoring can only be performed indirectly through the behavior of the electrical current. For example, insulation damages in power systems, such as generators or defects in medium-voltage cables. Medium voltage overhead power lines run for hundreds of miles to supply power to cities. These great distances make it expensive to manually inspect the lines for damage that does not immediately lead to a power outage, such as a tree branch hitting the line or a flaw in the insulator. An electrical discharge caused by these modes of damage, known as partial discharge, occurs when the electrodes between the insulation systems are not fully bridged. In the long run, partial discharges will cause a power outage or start a fire if left unrepaired. In this chapter, we introduce a few fundamental concepts behind partial discharge and power line signal processing that will be required to perform our work.

## 1.2 Partial discharge

The formal definition is “A localized electrical discharge that only partially bridges the insulation between conductors and which can or cannot occur adjacent to a conductor” [18] [19]. In fact, the presence of the partial discharge (PD) can be indicative of anomalies in many electrical systems and can cause further degradation of the insulation. The high-voltage discharges deteriorate the insulation materials and can have impacts on the entire system. Their detection is, therefore, of utmost importance to assess the condition of electrical components and has been a long-standing challenge. As such, the literature is extremely vast [20]. PD detection has been studied in many systems such as in transformers, gas-insulated high-voltage switchgear, power plants, and power lines [21]. The main challenge lies in the detection of extremely short and temporally localized events: their wavelength is at the microsecond scale. It requires, therefore, extremely high-frequency data (several tens of MHz) [22].

### 1.3 Electrical signal

An electrical signal is a function that transmits data about a phenomenon. Any number that may change over time or space can be used as a signal to communicate between observers [23]. Audio, video, voice, sonar, and radar are all examples of signals. The information contained in a signal is often representative of another physical phenomenon or a result of calculations (or measurements). As a rule, a signal varies continuously because the information is in motion or undergoes slow or rapid variations or disturbances. The signal can easily be measured with measuring devices such as voltmeters and oscilloscopes, for the most common, sometimes vectors scopes or spectral analyzers [24].

#### 1.3.1 Disruptive elements

Noise in electronics represents any kind of disturbing signal that is not derived from or associated with the input signal of a system [25]. The greater the noise, the less information there is. We can distinguish three main types of noise in case of electric signals:

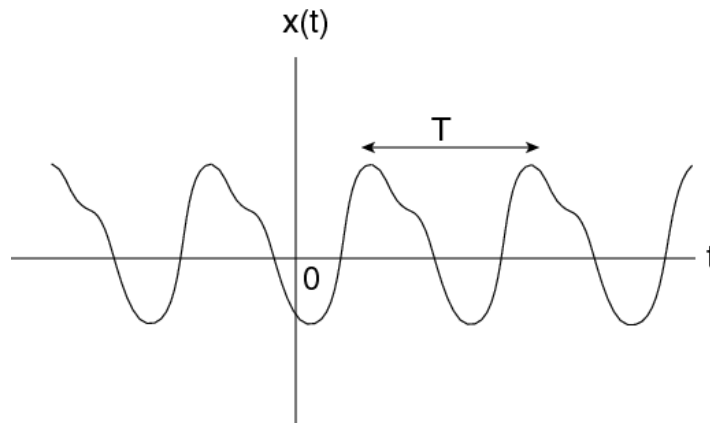
**Electromagnetic noise** is produced mainly by sparks from electric motors, lightning storms, neon signs, etc. It can be reduced by shielding cables or circuits or by electronic devices such as circuits with capacitors or VDR varistors. It can be reduced by shielding cables or circuits or by electronic devices such as circuits with capacitors or VDR varistors.

**Ripple noise** produced by power supplies that convert AC voltage to DC voltage is really noise because it is independent of the wanted signal and is superimposed on it. It can easily be reduced to negligible quantities by electronic circuits such as Zener stabilization, regulation, etc.

**Thermal noise** is due to the random displacement in different directions of the valence electrons from one atom to another contained in an electrical material. This displacement is due to the temperature and its consequence, which is the agitation of the atoms between them.

#### 1.3.2 Periodic signal

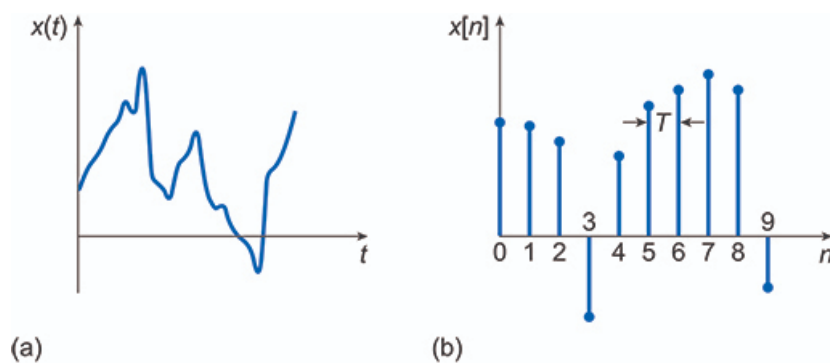
We call a signal periodic if it contains a pattern that repeats after a certain amount of time [26]. The time it takes for this signal to repeat itself is called the **period  $T$** , and the distance this period travels is called the **wavelength**. The **frequency** of a periodic phenomenon corresponds to the number of repetitions of the latter during one second. The legal unit of frequency is the hertz, symbol Hz. It is also possible to use derived units such as millihertz, centihertz, decihertz. The period  $T$  and the frequency  $f$  are linked by the relation:  $f = 1/T$ . The frequency is therefore simply the inverse of the period [23] [27]. Figure 1.1 illustrates an example of a periodic signal.



**Figure 1.1: Periodic signal representation.**

### 1.3.3 Signals with continuous and discrete support

There are two kinds of signals; signals with continuous support, denoted  $x(t)$  and signals with discrete support, denoted  $x(n)$  [28]. A continuous signal is defined for any value  $t$  of the support. It has an independent variable which is continuous in nature, i.e. it is present at each time-step within its domain, as shown in Figure 1.2. In contrast, the discrete signal is only known for certain values of  $t$ : it is a list of support-value pairs. In nature, the vast majority of signals are analog (continuous); for example, electrical signals in our body, human speech, any other sounds we hear, light during the day, atmospheric pressure, etc. Digitizing these analog signals for analysis and visualization on a computer turns them into discrete signals.



**Figure 1.2: Continuous vs Discrete Signals.**

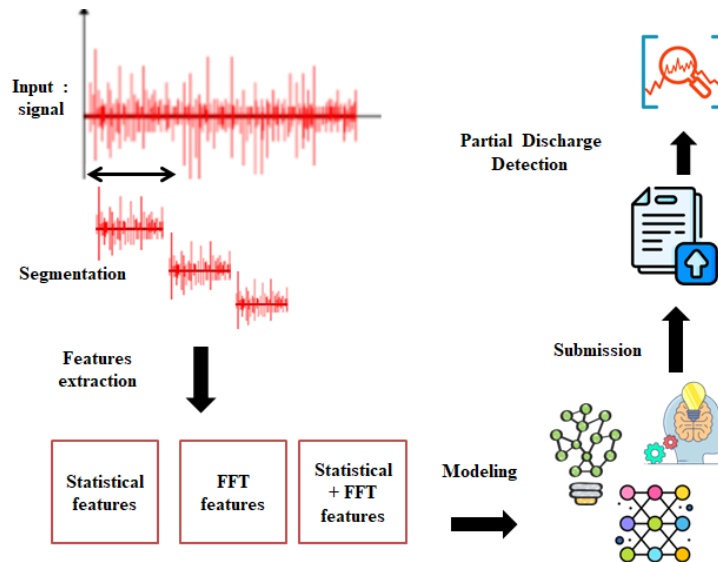
## 1.4 Development pipeline for partial discharge detection

The development process consists of two major steps: feature extraction, and machine learning. In the first step we extract a variety of features that can define the partial discharge anomaly by using different techniques. Then, we feed the extracted features to a learning model. In this step, machine learning and deep learning models are used, both traditional machine learning



classifiers and deep sequential models. The main goal of our work is to thoroughly explore and study different models by numerous parameters in order to obtain detectors with high detection rates.

Then at the end of our investigations, we have submitted our systems to the VSB competition in order to get the final scores. Figure 1.3 summarizes this pipeline.



**Figure 1.3: Development pipeline for partial discharge detection.**

## 1.5 Feature Engineering

Anomaly detection operates in two stages: developing the right features, and then feeding these features into a statistical system that detects anomalies in the features [18]. Most literature on anomaly detection focuses on the second part. Our goal is to illustrate the importance of the first part and to explain what feature engineering is. Feature engineering involves extracting features from raw data, with two principal goals of improving the performance of machine learning algorithms: first, providing the machine learning algorithm with the right input dataset, and second, enhancing the performance of machine learning models.

### 1.5.1 Framing

Framing, also known as frame blocking, is a fundamental signal processing technique that allows the decomposition of the original signal into a series of overlapping blocks, often called frames with the same length [29]. Overlapping the frames helps avoiding information loss in between adjacent frames. The main purpose of this technique is to capture a chunk of a signal that contains an anomaly.

### 1.5.2 Fourier transform

The Fourier transform is a mathematical method of transforming a signal from a functional representation to a Fourier representation. This is done by summarizing its sinusoidal or complex exponential components. It allows the passage from the temporal representation that shows the way the overall signal waves amplitude changes over time to the frequency representation that shows how much of the signal lies within each given frequency band over a range of frequencies [30]. When the Fourier transform is applied to a signal expressed as a function of time, it provides a complex value whose imaginary part represents the phase off-set of the pure sinusoidal component and the real part value represents the corresponding frequency component. Fourier Transform is useful in signal processing because signals are usually defined over time and Fourier Transform is useful for further analysis of time series [31].

The ordinary Fourier Transform is for a continuous function. The continuous Fourier Transform is difficult to use in real time and to be implemented on signals with continuous support. For this reason, the **Discrete Fourier Transform (DFT)** was invented [31]. The discrete Fourier transform (DFT) converts a finite sequence of equally-spaced samples of a function into a same-length sequence of equally-spaced samples of the discrete-time Fourier transform (DTFT) [31].

The **Fast Fourier Transform (FFT)** is an algorithm that increases the computation speed of the DFT of a sequence by simplifying its complexity. Fourier analysis converts a signal from its original domain (time or space) into the frequency domain and vice versa. Each frame having  $N$  samples is converted into the frequency domain [32]. Fast Fourier transform is a fast algorithm to apply DFT on the given set of  $N$  samples. FFT manages to minimize the difficulty of computing the DFT from  $O(N^2)$  which arises if the concept of DFT is simply applied to  $O(N \log N)$ , where  $N$  is the data size.

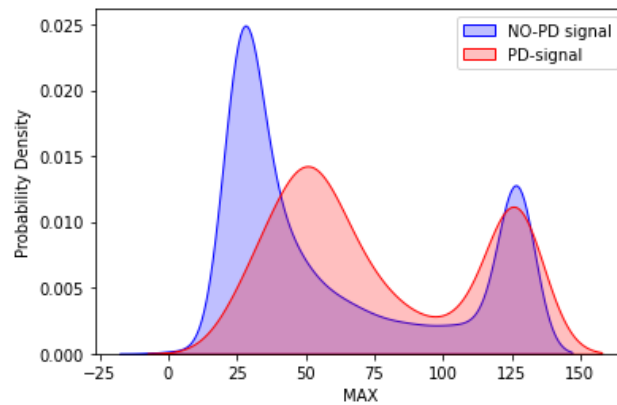
**Power spectral density (PSD)** is a mathematical quantity that defines the spectral content of a signal. The PSD describes how a signal's power is distributed in frequency [33]. Traditionally, PSD has been used to quantify vibration characteristics of a system. It also has many electronics applications to quantify noise characteristics [34] Similar to the FFT, it describes the frequency spectrum of a signal. But in addition to the FFT it also takes the power distribution at each frequency (bin) into account. The surface below the peaks corresponds with the power distribution at that frequency. Calculation of the Power Spectral density is a bit easier using **Welch** function which not only return a vector of amplitudes, but also a vector containing the tick-values of the frequency-axis [35].

### 1.5.3 Feature extraction

We can distinguish two primary categories of feature extraction techniques: statistics-based and signal processing-based features.

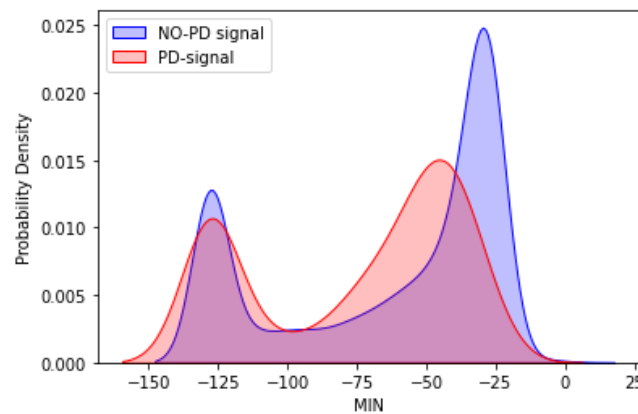
**A. Statistical Features:** The main idea of these types of features is to frame our signal into chunks and extract from each segment a number of features (this step is applied only on sequential models). In general, these features are determined by directly analyzing the temporal waveform. They are usually quite straightforward to compute. We illustrate some these features as follows:

**Maximum:** it defines the highest value in the current signal. The plot below (Figure1.4) is obtain over analyzing the max values on the training set. If the maximum of signal is less than 50 there's more probability of no partial discharge.



**Figure 1.4: Max values for anomalous/non-anomalous signals.**

**Minimum:** it describes the signal's lowest point. If the minimum of signal is greater than -50 there's more probability of no partial discharge as the plot below shows.

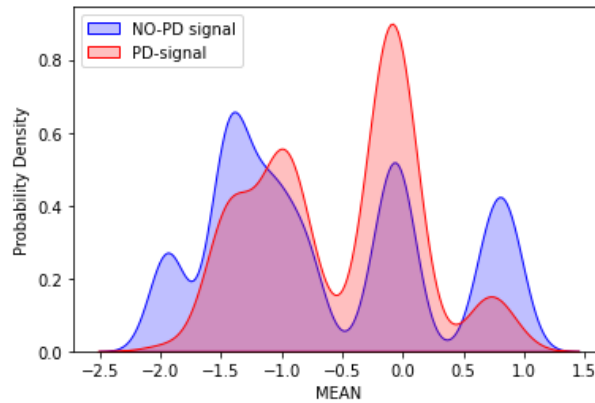


**Figure 1.5: Min values for PD/non-PD training signals.**

**Mean ( $\bar{X}$ ):** it defines as the average of the values of the signal.

$$\bar{X} = \frac{1}{N} \sum_{i=1}^N x_i, \quad (1.1)$$

where  $x_i$  is the magnitude of the signal at a discrete time instance and  $N$  is the length of the signal. The average of the training set of the VSB data base shows that If the mean of the signal is between less than -2 or more than 0.5 it is more likely that there is no partial discharge.

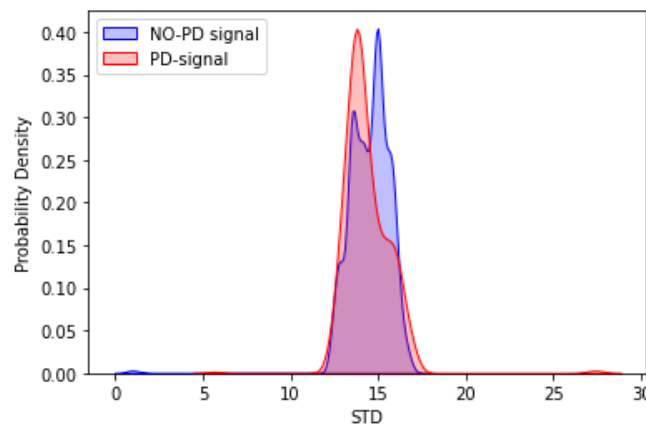


**Figure 1.6: Average of PD/non-PD training signals.**

**Standard-Deviation ( $\sigma$ ):** it is used to represent the dispersion of signal values in a statistical sample.

$$\sigma = \sqrt{\frac{\sum_{i=1}^N (x_i - \bar{X})^2}{N}}, \quad (1.2)$$

where  $\bar{X}$  denotes the mean,  $x_i$  is the magnitude of the signal at a discrete time instance and  $N$  is the length of the signal. Note that it is more probable that STD is greater than 15 if there is no partial discharge. From both mean and STD it can be said that the signal is spread more if there is no partial discharge. As shown in Figure 1.7.



**Figure 1.7: Standard-deviation of PD/non-PD training signals.**

**Bandwidth:** it is defined by  $\bar{X} - \sigma$ .

**Percentile:** a percentile of a vector  $V$  defines the value  $q/100$  of the way from the minimum to the maximum in a sorted copy of  $V$ ; this function is the same as the median if  $q=50$ , the same as the minimum if  $q=0$  and the same as the maximum if  $q=100$ .

**B. FFT features:** The FFT features are obtained through transforming the time-based signal to the frequency domain using the **Fourier Transform**. Other features can also be obtained such as standard deviation, the maximum and minimum at each frequency. It is also considered the power density distribution according to variate frequencies. It is concluded that the exact extraction of features has become the most crucial aspect in the current anomaly detection methods [19].

### C. Other features :

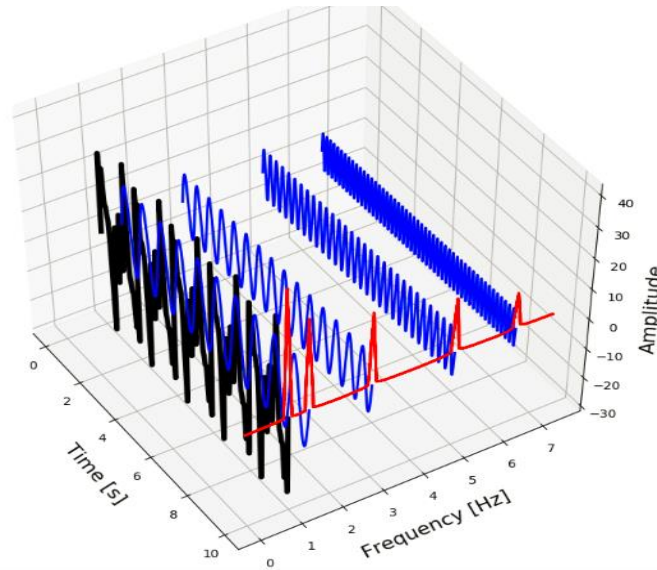
**Empirical Mode Decomposition (EMD)**, proposed by Huang et al., is a novel self-adapting method especially for non-linear analysis and processing non-stationary signals [36].

**Sample Entropy (SamEn)**, proposed by Richman, was used to evaluate the complexity of a time series [36].

## 1.6 Temporal domain representation vs frequency domain representation

Electrical signals have representations in the time and frequency domains [37]. In the time domain, the voltage or current is expressed in terms of time, and signals measured on an oscilloscope are displayed in the time domain [38]. The time-domain representation  $S(t)$  reveals information about the actual presence of the signal, its start and end times, its strength and temporal evolution, and it indicates how the signal energy is distributed along the T axis [39].

The signals can also be represented by the amplitude and a phase as a function of frequency. The frequency domain is an analysis of signals or mathematical functions with reference to frequency instead of time. Moreover, you can convert a designated signal between the time or frequency domain with a pair of operators called the **Fourier transform**



**Figure 1.8:**A signal (black) consisting of multiple component signals (blue) with different frequencies (red).

## 1.7 Conclusion

In this chapter, we have reviewed the basics of Partial discharge representation and some signal processing notions that are necessary for the comprehension of this work. We have also presented several types of features that exist in literature. These features are required to be an input for the learning stage. In The next chapter we will give an overview of some learning models that have been frequently used for addressing partial discharge detection.

## Chapter 2 : Machine Learning Fundamentals

### 2.1 Introduction

In the previous chapter we have discussed the notions of power line signals acquisition and we have also defined the fundamental concept and notions behind the partial discharge anomaly and listed their most substantial feature, including numerous representations and preprocessing techniques required to prepare the power line signal for the machine learning task. Machine learning is the field of study that gives computers the ability to learn without been explicitly programed. In the other hand, deep learning is a subset of machine learning. It consists of a set of models known as neural networks that unlike the machine learning models are hungry which means they need a large amount of data to perform well. Different categories of machine learning approaches have been introduced in the literature. They fall into three primary categories: Supervised Learning, Unsupervised Learning, and Semi-Supervised Learning. We undertake supervised learning approach to address scene classification problem. ML has successfully evolved into deep learning (DL), which addresses complex and large-scale problems with robust, adaptable, and efficient solutions.

### 2.2 Generalities on machine learning

Artificial intelligence (AI) is a scientific discipline that aims to create intelligent machines [40]. Machine learning is a form of (AI) that allows a system to learn from data and not from explicit programming [41]. Particularly, it enables computers to perform various tasks by learning from past experience rather than being **explicitly programmed**. It can be defined as the study of the construction of computer programs that automatically improve and/or adapt their performance through experience; it can be thought of as “programming by example”[40]. A machine learning model is the output generated when a machine learning model is trained on data A trained model takes data samples as input and produces the output result [42]. Depending on the nature of the problem being addressed, there are different approaches that vary depending on the type and volume of the data .These approaches can be classified into three primary categories: **Supervised learning, unsupervised learning and semi-supervised learning** [42]. Supervised learning entails learning a mapping between a set of input variables X and an output variable Y and applying this mapping to predict the outputs for unseen data. Supervised learning creates a knowledge base that enables the classification of new patterns. Supervised learning is the most important methodology in machine learning and it also has a central importance in anomaly detection tasks [43]. Unsupervised methods receive unlabeled input training data. However, the issue with unlabeled data is that we do not have a

correct result to match, which means there are no error or reward signals to evaluate a potential solution. Therefore, the learning algorithm will try to discover persistent patterns and find hidden structures to link results that are close to each other in order to group them into classes using clustering algorithms [44][45]. Semi-Supervised Learning (SSL) is half way between supervised and unsupervised learning [46]. The main objective of SSL is to overcome the drawbacks of both supervised and unsupervised learning. It can learn with a small amount of training data to label the unknown (or) test data. SSL builds a model with few labeled patterns as training data and treats the rest of the patterns as test data [47].

### 2.2.1 Classification

The classification problem is one of the main problems in the machine learning area. It entails categorizing examples into a discrete set of classes [48]. A classification algorithm falls under the category of Supervised Learning, in which the input data is labeled before processing. This process involves predicting class labels for a given set of data points (called samples or instances) using a feature vector  $x \in X$  and its class label  $y \in Y$  [48]. An input-output association  $T$  is taken into account by the classification algorithms  $\{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$  where  $x \in X$  and  $y \in Y$ , and learn a mapping function  $f$  from a feature vector  $x \in X$ .

**Binary classification** is the term used to describe the problem with two classes. It can be defined as follows: Given a set of  $m$  examples  $(x_i, y_i), i = 1, 2, \dots, m$  (the training data set of vectors  $x \in \mathbb{R}$ ), the output is a function  $C: \mathbb{R} \rightarrow \{0, 1\}$ , where  $C$  is a classifier and  $C(x)$  represents the predicted output of sample  $x$ . **Binary classification**, is used in many different data science applications, such as: Medical Diagnosis, Email Analysis, anomaly classification[49].

In our work, we formulate the problem of detecting partial discharge within power line signals as a **Binary classification** problem.

### 2.2.2 Performance measures

During the training phase, the evaluation measure is a critical component in getting the best classifier. Thus, selecting an appropriate assessment measure is of paramount importance for differentiating and choosing the best classifier [50]. In the case of classification, there are numerous forms of performance evaluation metrics such as confusion matrix, accuracy, and others.

However, most of these metrics are not useful when the dataset of the task suffers from the imbalanced class problem [50]. This issue arises when one class has more instances than another, which frequently occurs in real life scenarios. As a result, accuracy gives low performance statistic. If class A appears in 90% of our samples and class B appears in 10%, we can simply attain 90% accuracy by building a model that only predicts class A, one metric that helps with this problem is Matthew's Correlation Coefficient (MCC), which was introduced in the binary setting by Matthews



in 1975 [50]. Before we present the formula of MCC, we first introduce the concept of a confusion matrix. As shown in Figure 2.1, a confusion matrix has 4 cells, created by a combination of the predicted values against the real values. Two of those cells represent correct predictions (True Positives and True Negatives), and the other represent incorrect predictions (False Positives and False Negatives).

|        |               | Predicted                             |                                      |
|--------|---------------|---------------------------------------|--------------------------------------|
|        |               | Negative (N)<br>-                     | Positive (P)<br>+                    |
| Actual | Negative<br>- | True Negatives (TN)                   | False Positives (FP)<br>Type I error |
|        | Positive<br>+ | False Negatives (FN)<br>Type II error | True Positives (TP)                  |

**Figure 2.1: Confusion matrix [50].**

Matthew's correlation coefficient is calculated as follows:

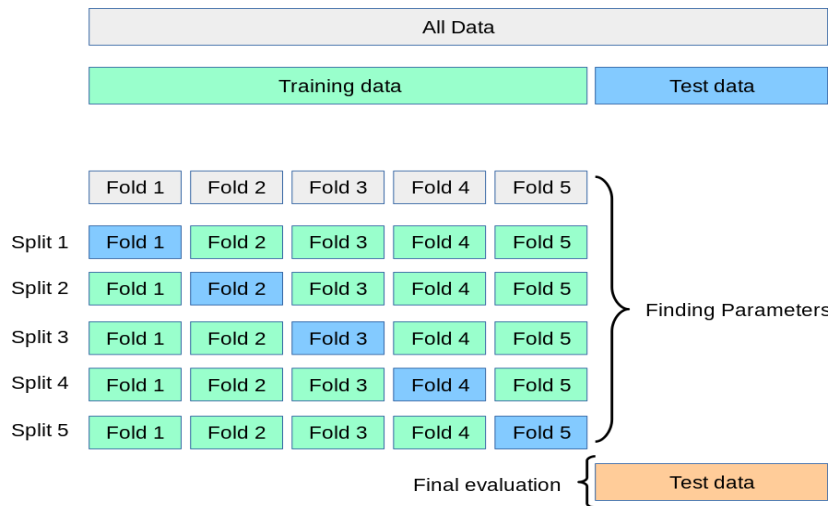
$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (2.1)$$

The MCC measures the agreement between the predictions and the true class labels. It takes values between -1 and 1. A score of 1 indicates perfect agreement; whereas a score of -1 indicates a worst prediction

### 2.2.3 Cross validation

Cross-validation is a statistical method, also known as resampling procedure, used to estimate the performance of a machine learning model on a limited data sample. It is commonly used to compare and select a model for a given problem. Several resampling approaches have been introduced in the literature; we can cite 5x2 cross validation, generally k-fold cross validation [51].

K-Fold Cross-validation is an iterative approach. During iteration  $i$ , it randomly divides the set of observations, usually a development set, into  $K$  groups or folds, of approximately equal size. Fold  $i$  is treated as a validation set, and the remaining  $K - 1$  folds assigned as a training set. This procedure is repeated  $K$  times, as shown in Figure 2.2.



**Figure 2.2: Cross validation [51].**

## 2.3 Anomaly detection

Anomaly detection, also known as outlier detection is the process of identifying extreme points or observations that are significantly deviating from the remaining data. It is a vast area of data analytics [52]. The approach to Anomaly Detection depends on which problem area use case is focusing on, in our case Time Series Anomaly Detection is our interest. This latter is concerned with Time Series data, or data that evolves over time. For example, with our personal computer, CPU usage, network usage, and memory usage all increase over time. Anomaly detection falls into two main categories [52].

### 2.3.1 Unsupervised Anomaly Detection

These techniques do not need training data. Alternatively, Unsupervised Anomaly Detection is the most flexible setup which does not require any labels. Furthermore, there is also no distinction between training and a test dataset. The idea is that an unsupervised anomaly detection algorithm scores the data solely based on intrinsic properties of the dataset. Typically, distances or densities are used to give an estimation of what is normal and what is an outlier [52].

### 2.3.2 Supervised Anomaly Detection

Supervised methods (also known as classification methods) required a labeled training set containing both normal and anomalous samples to construct the predictive model. Theoretically, supervised methods provide better detection rate than semi-supervised and unsupervised methods, since they have access to more information. However, there exist some technical issues, which make these methods seem not accurate as they are supposed to be. The first issue is the shortage of a training data set that covers all areas. Moreover, obtaining accurate labels is a challenge and the

training sets usually contain some noises that result in higher false alarm rates. The most common supervised algorithms used for anomaly detection are, Supervised Neural Networks, Support Vector Machines (SVM), k-Nearest Neighbors, Bayesian Networks and Decision Tree and Ensembles like Random Forest or Gradient Boosting Machines and Random Forest [53]. For the remainder of this manuscript, we will focus only on Supervised Anomaly Detection, since our training set comes labeled.

### 2.3.3 Supervised Learning for Partial Discharge Detection

To solve the problem of partial discharge detection, two ML approaches have been employed: “**sequential modeling**” and “**traditional modeling**”.

- **Sequential modeling:** Sequence models are machine learning models that input or output data sequences, such as text streams, audio snippets, video clips and time series data. The crucial element to remember about sequence models is that the data we are working with rely on one another due to their sequential order. For time series prediction and anomaly classification, sequence models that are popular are: Recurrent neural network (RNN), Long Short Time Memory (LSTM), Bidirectional Long Short Time Memory (Bi-LSTM) etc.
- **Traditional modeling:** Multiple machine learning algorithms can be used for anomaly detection depending on the dataset size and the type of the problem. Most common models are Random Forest and boosting models.

In the following two sections, we present some relevant classifiers and their concepts that are necessary for understanding the ideas developed in this work.

## 2.4 Common machine learning classifiers

Ensemble learning, also known as multiple classifiers and committee-based learning, imitates our second nature to seek several opinions before making a crucial decision [54]. It refers to the process of creating a collection (also called team, committee, ensemble, and pool) of learning models whose predictions are merged together to produce the final decision. Numerous experimental and theoretical studies have demonstrated that a combination of multiple learning models reaches higher prediction performance and usually generalizes better than a single classifier [55].

In what follows, we review some major ensemble learning approaches that have been frequently invoked to address the anomaly detection problem. We start with, Random Forest, then, we present AdaBoost and two variants of the recently acknowledged Gradient Boosting Machine, namely XGBoost and LightGBM.

## Random Forest

Random Forest is an ensemble classifier that consists of many decision trees and the final classification of the test sample is decided by majority votes of every individual decision tree [54] [56]. A decision tree is a classifier expressed as a recursive partition of the instance space. It has three types of nodes: a node called “root” and a node with outgoing edge is called an internal node and leaf node. Each node corresponds to a certain property and a leaf node which makes a prediction [56]. Let us consider a binary classification problem formalized as  $\Gamma = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$  where  $x_i \in X$  and  $y_i \in Y = \{0,1\}$ . The classification of training samples begins at the root node which takes  $\Gamma$  as an input and asks for the value of a particular feature of the samples that can split  $\Gamma$  into different possible subsets. Different links from the root node correspond to the different possible subsets of  $\Gamma$  and based on the answer we follow the appropriate link to a descendent node. The links must be distinct i.e. one and only one link will be followed. The next step is to ask for the value of a particular feature that can split the chosen subset of  $\Gamma$  into other different subsets. We continue this way until we reach a leaf node, which has no further questions. Each leaf node bears a class label and the test samples are assigned to the class of the leaf node reached [57]. The way  $\Gamma$  is split is based on two important concepts that form our objective function that we are optimizing for to improve the performance of our model. These concepts are: entropy and information gain.

The entropy measures the impurity of the node to find the best value of the feature  $x_i$  that allows splitting  $\Gamma$  into different subsets. These subsets should minimize their entropies. This process is repeated recursively until we reach a leaf. Impurity means that each subset of features represents one type of class, in this case the *entropy* = 0 and the information gain reaches its maximum value of 1. However, some percentage of impurity is tolerated in order to stop further division to reduce the training time. The entropy  $H$  and information gain are computed using the following mathematical formulas.

$$H(x) = \sum_{i=1}^m -x_i \log_2(x_i) \quad (2.2)$$

$$\text{Information gain} = H(\text{parent}) - (\text{weighted average}).H(\text{children})$$

## AdaBoost

AdaBoost, abbreviation of Adaptive boosting, is the first practical boosting classifier [54]. It was initially proposed by Freund and Schapire as an ensemble method for improving the performance of a weak learner. It combines multiple base classifiers, generally decision trees, to produce a strong classifier that achieves accurate classification. The main idea behind this algorithm is to give more focus to patterns that are harder to classify. The amount of focus is quantified by a weight that is assigned to every pattern in the training set. At stage  $t$ , every training sample  $x_i$  receives a weight  $w_i(t)$  that indicates its probability of being selected to train a new weak classifier. The first classifier is built by fixing these weights to  $1/m$ , where  $m$  indicates the number of training samples i.e. all patterns initially have the same importance. In each iteration, the weight of each

misclassified instance is increased while the weight of correctly classified instances is decreased i.e. if a training sample is accurately identified, it has a lower chance of being employed in the next stage, but if a sample is misclassified, the likelihood of it being reselected increases. As a result, the succeeding classifiers concentrate on difficult-to-classify cases. AdaBoost assigns to the new trained classifier a weighting coefficient  $\alpha_i$ : accurate members receive higher weights. This process continues until the desired number of base learners or the overall accuracy has been reached. The final classification decision of a test sample is based on the weighted linear combination of these weak classifiers. AdaBoost algorithm is given by:

| <b>Training phase</b>       |  |
|-----------------------------|--|
| 1: <b>Input</b>             | $I$ : a weak learner.<br>$T$ : number of iterations.<br>$\Gamma$ : a set of $m$ labeled training samples.                    |
| 2: <b>Initialization</b>    | $t=1$ ;<br>$w_t^{(1)} = 1/m, i=1\dots m$ ;<br>$\Omega = \phi$ ;  |
| 3: <b>Repeat</b>            |  |
| 4:                          | -Learn hypothesis $h(t)$ from $\Gamma$ using $I$   |
| 5:                          | $\epsilon_t = \sum_{x_i, y_i \in \Gamma} W_i^t \times \mathbb{I}(h_t(x_i) \neq y_i)$   |
| 6:                          | <b>if</b> $\epsilon_t > 0.5$   |
| 7:                          | $T=t-1$  |
| 8:                          | <b>Break;</b>  |
| 9:                          | <b>End if</b>  |
| 10:                         | $\beta_t = \epsilon_t / (1 - \epsilon_t)$  |
| 11:                         | $w_i^{t+1} = \frac{w_i^t}{Z^t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$ |
|                             | Where $Z^t$ denotes a normalization which enable $w_i^{t+1}$ to be a distribution i.e. $\sum_{i=1}^m W_i^{t+1} = 1$ ;        |
| 12:                         | $\Omega = \Omega \cup \{h_t\}$ ;   |
| 13:                         | $\alpha_t = \log 1 / \beta_t$ ;  |
| 14:                         | $t=t+1$ ;  |
| 15:                         | <b>Until</b> $t \geq T$  |
| 16: <b>Output:</b>          | The ensemble members $h_1 \dots h_t$ and their voting weight $\alpha_1 \dots \alpha_t$                                       |
| <b>Classification phase</b> |  |
| 17: <b>Input</b>            | $x$ : a feature vector characterizing a pattern.   |
| 18: <b>Output</b>           | $\Omega(x) = \operatorname{argmax}_{c_i \in \mathcal{Y}} \sum_{j=1}^T \alpha_j \times \mathbb{I}(h_j(x) = c_t)$              |

### 2.4.1 Gradient boosting machine

The main goal of “boosting” is the conversion of a set of weak learners to a **strong and robust classifier** [58]. It can be used for regression and classification problems. To make a strong learner, the predictions of a number of independent weak learners have to be combined. This is achieved by taking the majority vote or calculating average of every weak learner's prediction as the final prediction. Iteratively add a weak learner to the ensemble until it offers the right classification [59]. In a gradient boosting machine, each training model is based on previously trained models. It includes a learning procedure with the goal of building base models that are maximally correlated. The steps of the gradient boosting are the following:

First we have the data input where  $x_i$  refers to training dataset and  $y_i$  refers to target and  $L(y_i, F(x))$  is the loss function Likelihood which is a function of the predicted probability  $p$  which can be convert to a function of predicted  $\log(\text{odds})$  knowing that :

$$P = \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}} \quad (2.3)$$

Then the loss function is:

$$L = y_i \log(\text{odds}) + \log(1 + e^{\log(\text{odds})}). \quad (2.4)$$

The first step consists in initialize model with a constant value:

$$F_0(x) = \text{argmin} \sum_{i=1}^n L(y_i, \delta) \quad , \quad (2.5)$$

where  $y_i$  is the observed values,  $L$  is the loss function, and  $\delta$  is the value for  $\log(\text{odds})$ . This is the summation of the Loss Function for each observed value, and argmin over  $\delta$  means that we need to find a  $\log(\text{odds})$  value that minimizes this sum. Then take the derivative of each loss function. The second step consists in calculating the pseudo residual with the following formula :

$$\text{Residual} = - \left[ \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)} \quad (2.6)$$

Thereafter, Fit a regression tree to the residual values and create terminal regions. For each leaf in the new tree, we calculate  $\delta$ , the output value of each leaf is the value for  $\delta$  that minimizes the summation where :

$$\delta = \frac{\sum \text{Residual}}{\sum [p*(1-p)]} \quad (2.7)$$

As the last making a new prediction for each simple where the new prediction  $F_m(x)$  is based on the last prediction using the following formula:

$$F_m(x) = \text{Old tree} + \text{Learning rate} * \text{New tree} \quad (2.8)$$

The learning rate determines the contribution of each tree on the final outcome and controls how quickly the algorithm learns. Similarly, we substitute and find the new *log* (odds) for each instance and hence find the probability. Using the new probability, we will calculate the new residuals. This process repeats until we have made the maximum number of trees specified or the residuals get super small. When training a gradient boosting model, selecting the appropriate number of models (i.e., iterations) is important. Setting it too high can result in overfitting, while setting it too low can result in under fitting [54].

Gradient boosting algorithm is given by:

|   |
|---|
| <p><b>Input</b> : Data <math>\{x_i, y_i\}, i=1..,n</math> and differentiable Loss Function <math>L(y_i, F(x))</math></p> <p>1: initialize model with a constant value :<br/> <math display="block">F_0(x) = \operatorname{argmin} \sum_{i=1}^n L(y_i, \delta).</math></p> <p>2: For <math>m=1</math> to <math>M</math> :</p> <p>3: Compute<br/> <math display="block">r_{im} = - \left[ \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)} \text{ for } i=1 \dots n.</math></p> <p>4: Fit a regression tree to the <math>r_{im}</math> values and create a terminal region.</p> <p>5: For <math>j=1..j_m</math> compute<br/> <math display="block">\gamma_{jm} = \operatorname{argmin} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)</math></p> <p>6: Update <math>F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{j_m} \gamma_j I(x \in R_{jm})</math></p> <p><b>Output</b> : <math>F_m(x)</math></p> |
|---|

## LightGBM

LightGBM is a specific instance of the Gradient Boosting Decision Tree algorithm. It is a very simplistic algorithm yet became one of the most successful nonlinear algorithms due to its superb performance and flexibility [60]. LightGBM uses a new technique: Gradient-based one side sampling (GOSS). It is used to make the model perform better and give it a competitive advantage over alternative Gradient Boosting Decision Tree (GBDT) frameworks. This technique allows us to focus on instances with a higher gradient which will accelerate the learning process (i.e. the under-trained instances) will contribute more to the information gain. GOSS keeps those instances with large gradients (e.g. larger than a predefined threshold) and just drops those occurrences with a small gradient to retain the accuracy of information gain estimation. LightGBM is an excellent choice for faster training, adequate efficiency, optimal memory, satisfactory accuracy, parallelism, and large-scale data processing features are achieved [61].

## **XGBoost**

The XGBoost (Extreme Gradient Boosting) algorithm is a machine learning algorithm for classifying and predicting data. XGBoost is another implementation of gradient boosted trees designed for high speed and accuracy, which has been proven to be very effective and widely used in machine learning competitions. XGBoost has high predictive power and is nearly ten times faster than other gradient boosting techniques since it supports distributed platforms such as Apache Hadoop and can be distributed across multiple machines. The algorithm also includes various regularization parameters that can reduce overfitting and improve the overall performance by creating ensembles that are simpler and more generative. Therefore, this technique is also known as the regularized boosting technique [62] [63].

## **2.5 Deep learning for sequential models**

### **2.5.1 Overview of deep learning**

Deep learning is a part of machine learning that deals with artificial neural networks (ANNs). These latter are algorithms inspired by the structure and the function of the brain. It drives many applications and services that improve automation, performing analytical and physical tasks without human intervention, such as digital assistants, voice-enabled TV remotes, and credit card fraud detection as well as emerging technologies (such as self-driving cars). In recent years, deep learning has seen tremendous growth in its popularity and usefulness, largely as the result of more powerful computers and many processing units like Graphics Processing Unit (GPU), Neural Processing Unit (NPU) and TPU (Tensor Processing Unit) [64].

### **Artificial neural network**

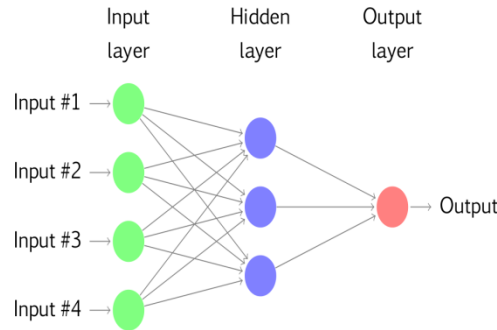
An artificial neural network (ANN), also known as neural network, or feed-forward neural network, is a computational model that is capable of processing information in order to tackle tasks such as classification and regression [65]. One of the reasons for the familiarity it has gained is the ability of its network's self-learning capability to solve complex problems which are difficult or unfulfillable for humans and even bring out better results than statistical methods.

### **ANN architecture**

ANN is a group of several neurons (nodes) at each layer, refer to Figure 2.3. A neural network is based on an **input layer**, an **output layer** and a **hidden layer**. The input layer accepts signals and data from the outside world. However, the output layer realizes the output of the system processing results. Moving to the hidden layer that is located between the input and the output layers, it cannot be observed from outside the device because of the number of neurons in both layers surrounding it that are often fixed. The number of hidden layers and the number of neurons in each layer determine the complexity of the neural network. A single neuron generally has multiple inputs. After some



calculations (i.e., weighted addition), the output will be used as the input of subsequent neurons. In a neural network, every connection between two neurons represents a weight to reflect the strength of the connection. Finally, neurons are combined layer by layer to form a neural network [66].



**Figure 2.3: ANN architecture [66].**

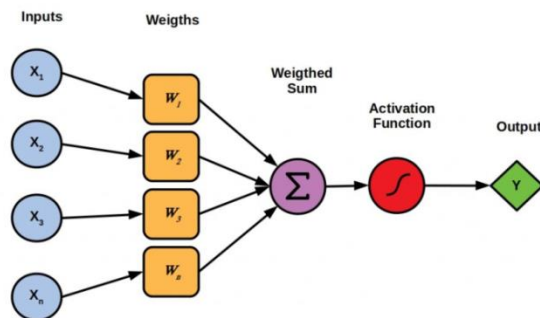
A **perceptron** consists of a single node. It is formally defined as a function  $f_j$  of  $N$  input  $x = (x_1, x_2, \dots, x_N)$  weighted by a vector of connection weights  $w_j = (w_{j1}, w_{j2}, \dots, w_{jN})$  completed by a neuron bias  $b_j$ , and coupled with an **activation function**  $\varphi$ .

$$y_j = f_j(x) = \varphi((w_j, x) + b_j) \quad (2.9)$$

The NN equation is defined by the formula:

$$y_j = \varphi \left[ \sum_{i=1}^N (w_{ij} x_i) + b_j \right], \quad (2.10)$$

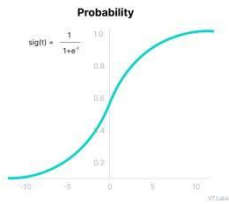
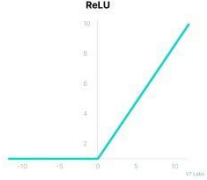
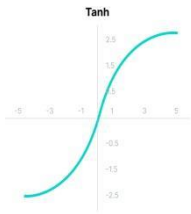
where  $x_i$  is the inputs,  $w_{ij}$  is the weight of connection,  $b_j$  is the bias.



**Figure 2.4: perceptron [66].**

Several activation functions have been introduced in the literature. The most widely used activation functions are shown in Table 2.1.

Table 2.1: Activation functions formulas.

| <i>Activation Function</i>              | <i>Formula</i>                                   | <b>2D plot</b>   |
|---|--|--|
| <i>Sigmoid (logistic)</i>               | $\varphi(x) = \frac{1}{1 + \exp(-x)}$            |   |
| <i>The Rectified Linear Unit (ReLU)</i> | $\varphi(x) = \text{Max}(0, x)$                  |   |
| <i>Hyperbolic tangent (tanh)</i>        | $\varphi(x) = \frac{\exp(2x) - 1}{\exp(2x) + 1}$ |  |

For the output layer, the activation function is generally different from the one used on the hidden layers. In the case of binary classification which coincides with our goals, the output predicts  $P(Y = 1|X)$ . Because this value is in the range  $[0, 1]$ , the sigmoid activation function is commonly used.

### 2.5.2 Training a sequential model

**Backpropagation** refers to the method of calculating the gradient of neural network parameters. In short, the method traverses the network in reverse order, from the output to the input layer. Backpropagation uses the chain rule to calculate the derivative of the loss function  $L$  with respect to each parameter in the network. The **Loss function**  $L(\hat{y}, y)$  penalizes the distance between the output  $\hat{y}$  and the target  $y$ . In general, it computes the difference between the algorithm's current output and its expected output. It is an evaluation of the algorithm's ability to model data. Many loss functions have been introduced in the literature depending on the task at hand: Classification or Regression. We can cite: Mean Squared Error (MSE) for regression and Binary Cross Entropy (BCE) for classification [67]. For sequence models, a variant of gradient descent namely **backpropagation through time (BPTT)** is commonly used. The **BPTT** is the application of Backpropagation training algorithm which is applied to the sequence data like the time series. It is widely used to train LSTMs and other recurrent models.

In deep learning and machine learning, **Gradient Descent (GD)** optimization plays an important role. Several new variant algorithms have been developed in recent years to further enhance efficiency. Optimization aims to define the key parameters that make the solution easier. Many machine learning algorithms have issues with this. The Gradient algorithm is an example of an optimization algorithm that classifies weights by minimizing the cost function. In terms of performance algorithms, GD is by far the most popular and most commonly used method of optimizing neural networks. GD works to determine a distinct role at a local minimum. The purpose of it is to determine the parameter ( $W, b$ ) values (coefficients) of a function which significantly decrease the **loss function**. The theory is to take repeated steps in the opposite direction of the gradient at the current stage (or an approximate gradient) of the function because this is considered the steepest descent direction [68]. The main steps are defined as follows:

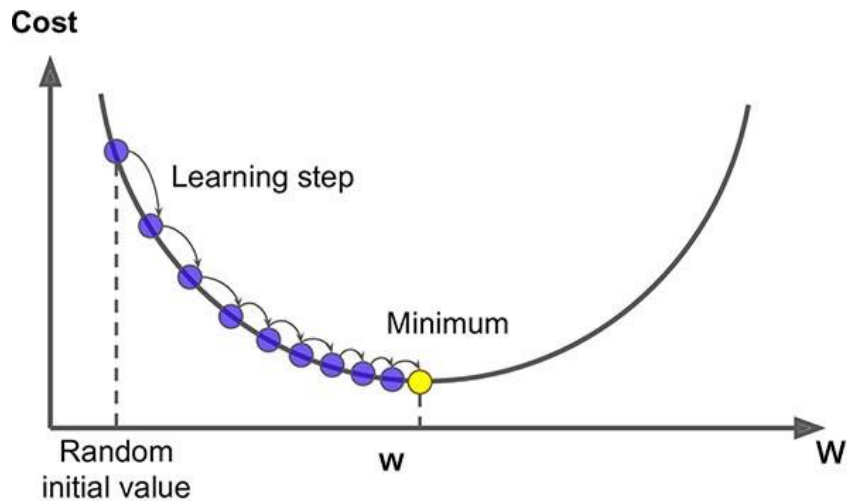
We start from a random initial point, and then measure the value of the gradient at that point. *And how is a gradient measured in mathematics?* By calculating the derivative of the loss function for

both parameters  $\frac{dL}{db}$ ,  $\frac{dL}{dW}$ . Therefore, the second step consist to update the optimization parameters

( $W, b$ ) so that we can minimize our derivative using this formula:  $W_{t+1} = W_t - \alpha \frac{dL}{dW}$ , where  $W_{t+1}$  is

the updated weight at instant  $t+1$ .  $W_t$  is the old one, and  $\alpha$  is the learning rate .Finally, we repeat this process until we get the parameter values that decrease the loss function in order to achieve a global minimum. The goal of this algorithm is to adjust the value of all weights in order to minimize the loss function and get the neural network's prediction result for the input sample closer and closer to the real value as shown in Figure 2.5. There are also several variants dependent on the GD approach that can be used to maximize the algorithm's performance, such as Batch Gradient Descent (BGD), Stochastic Gradient Descent (SGD) and Adaptive Moment Estimation (Adam).

**Adam optimizer** algorithm was described in the 2014 [69]. Adam is a method of GD optimization that measures adaptable learning rates for each parameter. Adam is one of the most common step-size strategies in the field of neural networks. The name was taken from Adaptive Moments [70]. The upgrade operation considers the smooth gradient variant and provides a better final result [71] [72]. Adam lowers computing costs, needs less execution memory, and is invariant to gradient diagonal rescaling. The results of the Adam optimizer are generally better than every other optimization algorithms, have faster computation time, and require fewer parameters for tuning. Because of all that, Adam is recommended as the default optimizer for most of the applications. Choosing the Adam optimizer for our application might give as the best probability of getting the best results.



**Figure 2.5: Gradient descent graph.**

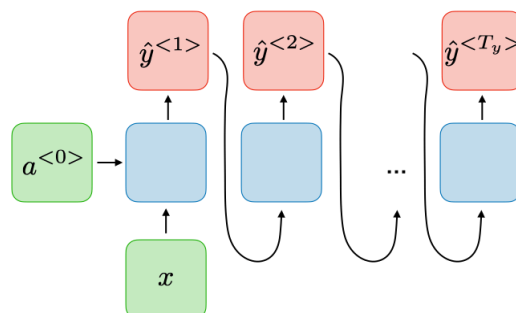
The general algorithm is given by these main steps

1. Present the input pattern and propagate it through the network to get the output.
2. Calculate the error by comparing the predicted output with the expected output using a loss function.
3. Compute the derivatives of the error based on the weights of the network using BPTT.
4. Adjust the weights so that the error is minimum using the Gradient **Descent** algorithm.

### 2.5.3 Common sequential models

#### Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a type of neural networks that are beneficial to use with sequential data [73]. The structure of RNN is similar to that of the standard neural network, with a distinction that RNNs allow their neurons to share their outputs with previous layer neurons. The general form of RNNs is depicted in the following diagram (Figure 2.6):



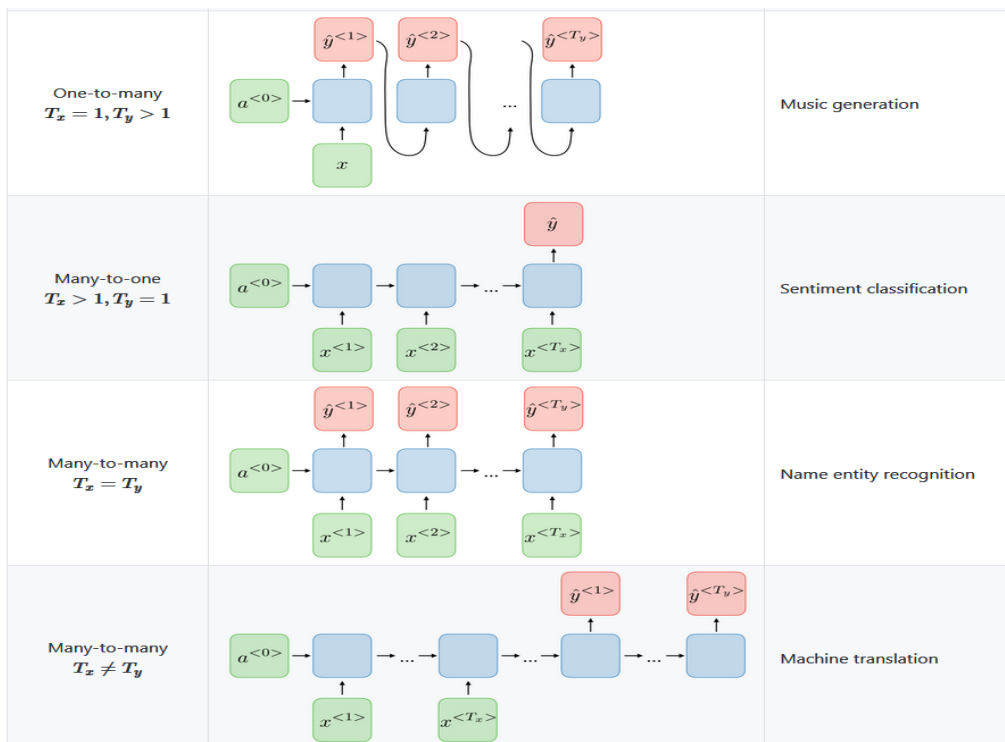
**Figure 2.6: Recurrent neural network [73].**

As we can see in the preceding diagram, data from previous time points goes into the training of the current time point. The recurrent architecture makes the models work well with time series and sequential data. In RNNs, the information is passed from the current state to the next, as shown in the unrolled version of the network in (Figure 2.7). RNN also learns with training data. From there on, it does not process data on inputted data alone. Instead, it uses data from past inputs to make decisions too. (The same task is performed on each element of the sequence, and an output depends on the output of the previous calculations). Put simply, this architecture is built for having a 'memory'.



**Figure 2.7: RNN architecture [73].**

RNNs can be categorized into four architectures: many-to-one, one-to-many, many-to-many (Unequal Unit Size) and many-to-many (equal Unit Size) based on their input and output. Figure 2.8 shows these categories attaching with some application examples.

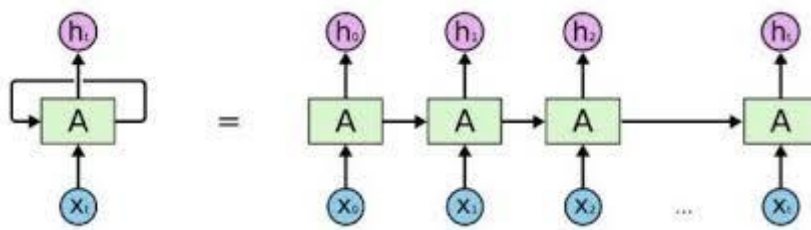


**Figure 2.8: RNNs types and applications [73].**

From the perspective of hidden layers, the most commonly used RNN architectures include the basic RNNs, and the bidirectional ones, LSTM and GRU. We will focus on these four architectures of RNNs, and will start by briefly presenting these architectures.

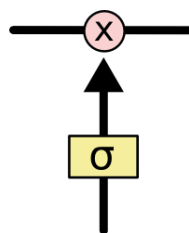
### Long Short-Term Memory

The Long Short-Term Memory (LSTM) unit was initially proposed in order to mitigate vanishing and exploding gradient problems [74]. Since then, a number of minor modifications to the original LSTM unit have been made [74]. The core idea of LSTMs is an introduction of memory cells with a self-loop connection with a constant value 1 (Figure 2.9). Because a memory cell only runs through linear operations, it can store a bit of information for an arbitrary long period without suffering too much from vanishing gradients [75]. The vanishing gradients problem refers to the large decrease in the norm of the gradient during backpropagation. Such events are due to the long term components going exponentially fast to norm 0, making it impossible for the model to learn correlation between temporally distant events [75].



**Figure 2.9: Self-loop LSTM cell [74].**

LSTMs are far more sophisticated. The hidden state, as defined in LSTM, is decomposed as  $h_t = h_t; C_t$ , where  $h_t$  is usually referred to as the hidden state and  $C_t$  are called memory cells [76]. LSTMs deal with both Long Term Memory (LTM) and Short Term Memory (STM). The LSTM may delete or add information to the cell state, which is carefully controlled by structures called gates. The notion of gates is used to make computations easy and effective. A sigmoid neural net layer and a pointwise multiplication procedure are used to create them. Figure 2.10 illustrates the gate notion [74].



**Figure 2.10: Gate Form [74].**

Each unit in LSTM is made up of four functional gate units:

**Forget gate** " $f_t$ ": to filter the valuable information and to ignore the unhelpful ones from the previous cell

**Input gate** " $i_t$ ": controlling what information is stored in the current candidate cell.

**Candidate gate** " $\tilde{C}_t$ ": calculate the current candidate state by using  $\tanh$  layer to create a vector of new candidate values.

**Output gate** " $\tilde{O}_t$ ": controlling what information to release from the current candidate cell.

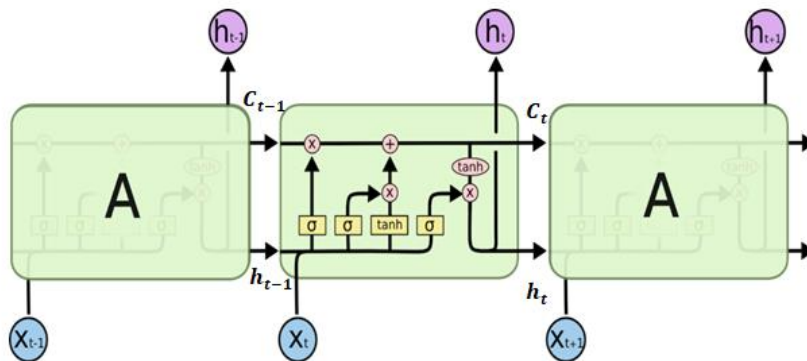


Figure 2.11: LSTM Architecture [74].

LSTM computational unit is defined as follows:

**Function**  $\text{LSTM-cell}_t(h_{t-1}, C_{t-1}, x_t)$  **returns**  $(h_t, C_t)$

**Input**

$h_{t-1}$ : The hidden vector at time step  $t-1$ .

$C_{t-1}$ : The output vector of time step  $t-1$ .

$x_t$ : is The input vector at time step  $t$ .

**Local variable**

$f_t, i_t, \tilde{C}_t, \tilde{O}_t$ : Gates.

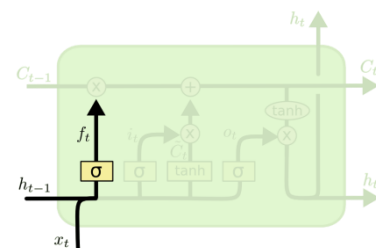
$W_f, W_i, W_c, W_o$ : Weights.

$b_f, b_i, b_c, b_o$ : Bias.

**begin**

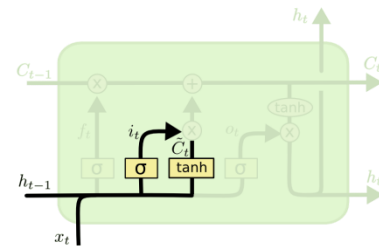
- Firstly select which information from the cell state will be discarded by a sigmoid function

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$



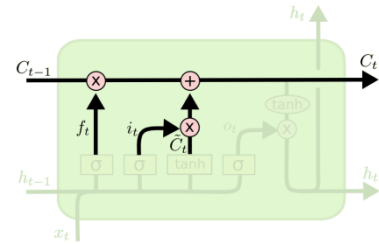
- Using a sigmoid layer and the  $\tanh$  function to choose which values to update.

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \end{aligned}$$



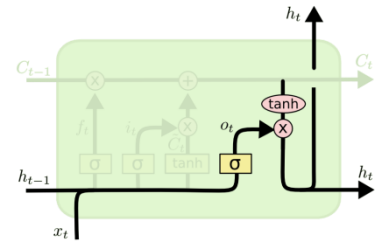
- Applying this formula to update the old cell state ( $C_{t-1}$ ).

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t$$



- Run the sigmoid layer to define the output. Then run it through the tanh function using these two equations.

$$\begin{aligned} \tilde{O}_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= \tilde{O}_t \times \tanh(C_t) \end{aligned}$$



**return** ( $h_t, C_t$ )  
**End.**

### Bidirectional-Long Short-Term Memory

Bi-LSTM is a combination of two LSTM with their own parameters: one encodes the input sequence in the forward direction, the other encodes the input sequence in the backward direction. It returns two hidden vectors  $\vec{h}_t \oplus \bar{h}_t$  at time step  $t$ . We represent the hidden states  $h_t$  of Bi-LSTM by making a concatenation of two hidden states, which captures both previous and future information in the sequence [74].

$$h_t = \vec{h}_t \oplus \bar{h}_t, \quad (2.11)$$

where  $\oplus$  is the concatenation operation.



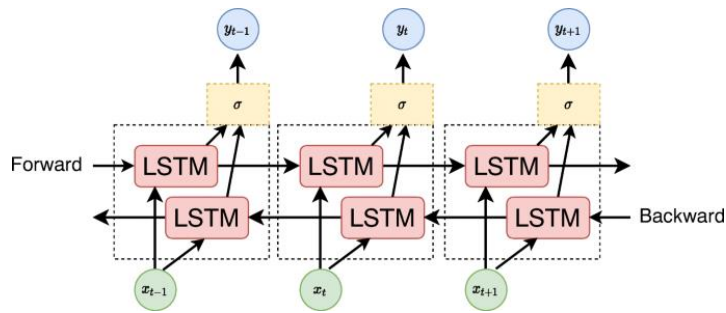


Figure 2.12: Bidirectional-LSTM architecture [74].

### Gated Recurrent Units

A Gated Recurrent Unit (GRU) is a lightweight version of an LSTM. Unlike LSTMs, GRUs do not have an explicit cell structure, and they are designed to adaptively reset or update the memory content. A GRU has a reset gate  $r_t$  and an update gate  $z_t$ . Because a GRU does not have a memory cell, the context is fully exposed at each time step, and the new context is determined by performing leaky integration between the previous context and the new context. Figure 2.13 below shows a graphical view of a GRU [75].

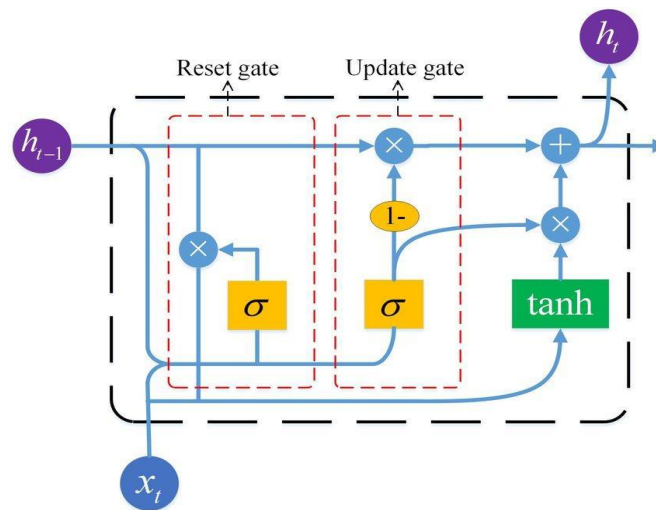
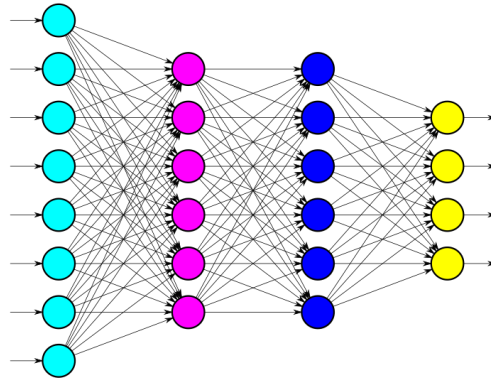


Figure 2.13: GRU architecture diagram [75].

### 2.5.4 Deep sequential models

Stacking more than one sequence model qualifies it as a deep learner. Deep sequential models are deep learning techniques used when the input is a sequence of data. Sequences are made up of data points that can be arranged so that observations at one point in the sequence provide meaningful information about observations at other places in the sequence.



**Figure 2.14: Deep sequential model.**

### 2.5.5 Attention mechanism

As a result of its ability to focus on the effective parts of features adaptively, the attention mechanism has demonstrated success in many tasks, such as image classification, anomaly detection, neural machine translation, multimedia recommendation, and others [77]. A particular region of an image, or video or signal can be selected adaptively by the proposed model's attention mechanism, and only those regions are processed at high resolution [77]. Recently the attention mechanism has also been widely applied in time series analysis [78]. Attention mechanisms allow for a more direct dependence between the states of the model at different points in time.

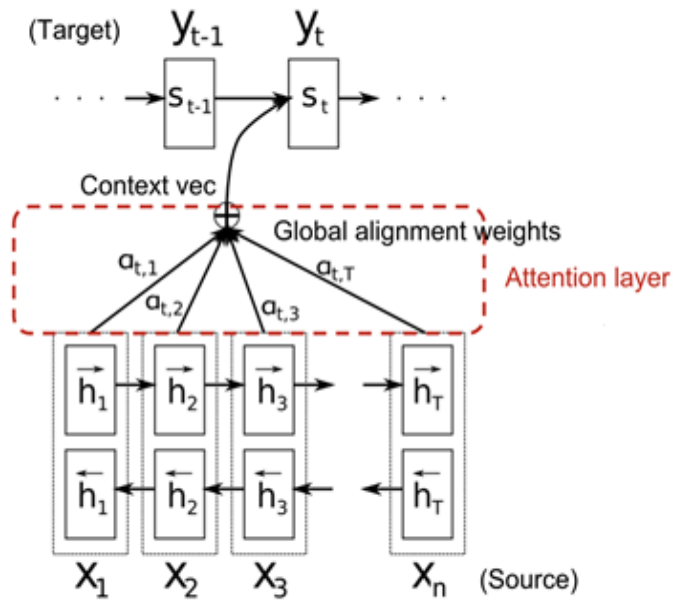
Following the definition, given a model which produces a hidden state  $h_t$  at each time step, attention-based models compute a “context” vector  $C_t$  as the weighted mean of the state sequence  $h_j$  by [79]:

$$C_t = \sum_{j=1}^T \alpha_{tj} h_j, \quad (2.12)$$

where  $T$  : is the total number of time steps in the input sequence, and  $\alpha_{tj}$  : is a weight computed at each time step  $t$  for each state  $h_j$ . These context vectors are then used to compute a new state sequence  $s$ , where  $s_t$  depends on  $s_{t-1}$ ,  $C_t$  and the model output at  $t-1$ . The weightings  $\alpha_{tj}$  are then computed by formula [79]:

$$e_{tj} = a(s_{t-1}, h_j), \alpha_{tj} = \frac{e_{tj}}{\sum_{K=1}^T \exp(e_{tK})}, \quad (2.13)$$

where  $a$  is a training function which can be thought of as computing a scalar importance value for  $h_j$  given the value of  $h_j$  and the previous state  $s_{t-1}$  [80]. This approach provides more direct access to the whole state sequence  $h$  for the new state sequences.



**Figure 2.15: Attention layer [80].**

## 2.6 Conclusion

Across this Chapter, we have reviewed the crucial concepts of classification. The main goal of our work is to conduct Machine Learning experiments for partial discharge detection in power line signals. First, we presented what classification is, and then we highlighted the traditional and sequential models as the main models required for our work. We have also introduced famous measures to evaluate the efficiency of learning models. In the next Chapter, we will introduce the experimental setup and describe the general environment deployed that we have used for our experiments.

## Chapter 3 : Experimental Framework

### 3.1 Introduction

In this chapter, we present the framework that we have used to conduct our tests. We first define all the libraries and the development set that we have used; as well the VSB dataset. Then, we define the feature extraction setup we have used to provide a numerical representation of our signals. Next, we describe the learning models invoked for training our detectors; namely: RNN, GRU, LSTM, Bi-LSTM, AdaBoost, XGBosst, LightBGM and Random Forest. Finally, we present the evaluation procedure that we have followed for assessing the performance of our detectors.

### 3.2 Utility libraries

We have conducted our experiments using Python 3.7 on Kaggle [81]. Kaggle is a web platform organizing data science competitions owned by Google. On this platform, companies propose data science problems and offer a prize to the data scientists with the best performance. In addition to these contests, Kaggle also allows users to publish and search datasets, which they can use for their machine learning projects. Kaggle provides free access to NVidia K80 GPUs in kernels. This benchmark shows that enabling a GPU to your Kernel results in a 12.5X speedup during training of a deep learning model.

**Python:** Another bright spot for Kaggle is the availability of all the necessary python libraries used for signal processing, deep and machine learning experiments. These libraries do not require any installation or configuration. The following are a few of the most relevant libraries that we have invoked in our work.

**Keras:** Keras is a high-level API written in python that runs on a Tensorflow backend. It is a highly-productive interface for solving machine learning problems, with a focus on modern deep learning. Its simplicity helps users develop a deep learning model quickly and provides a ton of flexibility while still being a high-level API [82].

**Tensorflow:** Tensorflow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization to conduct machine learning and deep neural networks research [83].

**Scikit-learn:** Scikit-learn is an open source machine learning library that supports both supervised and unsupervised learning [84]. It also provides various tools for model fitting, data preprocessing, model selection, model evaluation and many other utilities.

### 3.3 Training data-Set

The VSB data-base is proposed by the cooperation of both VSB – Technical University of Ostrava and ENET center. Since the signal data originate from the real environment, rather than a laboratory, they contain a lot of background noise. The **VSB dataset** contains two sets of measurements: training and test sets. The training set contains 8712 samples with 3 labels: the measurement ID, the phase, and whether the power line insulation was damaged at the time of recording. These signals are three-phase signals acquired 2,903 times. Damaged power lines should contain partial discharge, however no additional information is provided on the partial discharge types, shapes, or location. In this set, 575 samples are labeled as damaged power lines and 8186 with no PD. The test set contains 20337 signals with two labels: signal-ID, the measurement ID and the phase. No ground truth is provided with respect to the presence of PD. However, the assessment of the anomaly score on the test set can be obtained through the challenge website upon submission. Further details can be found in Section 4.2.5. Figure 3.1 shows a sample of the metadata file. The metadata contains the following information about the signals:

**Signal id** — a unique integer used to identify each signal. A signal id of ‘0’ corresponds to column ‘0’ in the signal data.

**Phase id** — 3 conductors are used to transfer the power from one region to another. Each conductor carries a signal. The phase of the signal in each conductor is different. The phase id mainly refers to the conductor in which the signal is being carried. Each signal id is associated with a unique phase id of 0, 1, or 2.

**Measurement id** — since the signals are measured using a meter, each signal is associated with a measurement id. Each measurement id is associated with 3 signals corresponding to the 3 phases of the signal.

**Target** — this field provides information on whether a given signal has a partial discharge pattern in it or not. A value of 0 corresponds to the partial discharge pattern not present and a value of 1 corresponds to the partial discharge pattern present for the respective signal id. This field is present only for the training data and has to be determined for the test data.

| signal_id | id_measurement | phase | target |     |
|-----------|----------------|-------|--------|-----|
| 0         | 0              | 0     | 0      | 0   |
| 1         | 1              | 0     | 1      | 0   |
| 2         | 2              | 0     | 2      | 0   |
| 3         | 3              | 1     | 0      | 1   |
| 4         | 4              | 1     | 1      | 1   |
| ...       | ...            | ...   | ...    | ... |
| 8707      | 8707           | 2902  | 1      | 0   |
| 8708      | 8708           | 2902  | 2      | 0   |
| 8709      | 8709           | 2903  | 0      | 0   |
| 8710      | 8710           | 2903  | 1      | 0   |
| 8711      | 8711           | 2903  | 2      | 0   |

8712 rows × 4 columns

**Figure 3.1: Metadata of training set.**

### 3.4 Features engineering

We have trained our detectors on a set of features that can define and describe the presence of the partial discharge anomaly. In this section, we present the feature extraction setup used in the development of our systems. We have extracted two sets of features: Statistics-based and signal processing-based feature sets, which we denote, respectively, Stat and FFT. The following subsections describe the process of extracting these features.

#### Preprocessing features

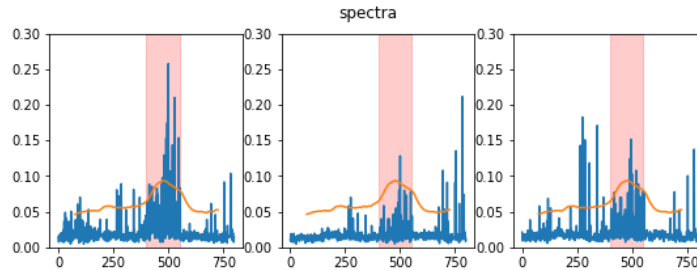
We have trained traditional models on preprocessed features to have a robust and efficient model. Each signal is preprocessed to identify the peaks and calculate the features. The following step describes the process:

First Flatten the trace using the flatiron function, then identify local maxima using the local-maxima-1d-window function to extract peaks once all peaks have been identified peaks caused by noise in a signal must be removed, this function is used using the get-peak function. Once the noisy peaks have been removed, features are computed for each of the remaining peaks. As a result we have 9 preprocessed features.

#### Statistical features “Stat”

For each signal in the training set  $x(t), t = 1 \dots 800000$ , we calculate the mean and 6 percentile values  $[100, 99, 95, 0, 1, 5]$  for every chunk of 1000 points, resulting in a 2D array (A) of shape (8712, 800). Recall that the training set is made of 8712 signals. Each entry of the previous array  $A_{ij}$  corresponds to the mean of the  $j^{th}$  chunk of signal  $i$ . In addition, we have a percentile N-

Dim array (P) of shape (8712, 800, 6), each entry  $P_{ij}^k$  corresponds to the  $k^{th}$  percentile value of the  $j^{th}$  chunk of signal i. When  $k=1$  the entry  $P_{ij}^k$  corresponds to the maximum value (100%) and for  $P_{ij}^4$ , this cell represents the minimum value. Then, we have employed 1D convolution on P in order to smooth its values. As shown in Figure 3.2.



**Figure 3.2: Three-phase signal Smoothed with 1D convolution.**

Then we have selected **3 main-features**. First we have extracted the max value from P, also we have defined the peak interval; from that interval we extract the mean and the max values. These features could be helpful in detecting partial discharge to have at the end of this process an array of 57 features for each 3-phases signal (2904, 57).

### Features "FFT"

The appellation "FFT" does not refer just to the Fast Fourier features, but also includes the PSD features and other common features. We have used this name just for concise notation. We have extracted **19 FFT features**; specifically:

- 8 features extracted using FFT.
- 8 features extracted using Welch power density function.
- **STD, Mean, Band-Width** of the training signals.

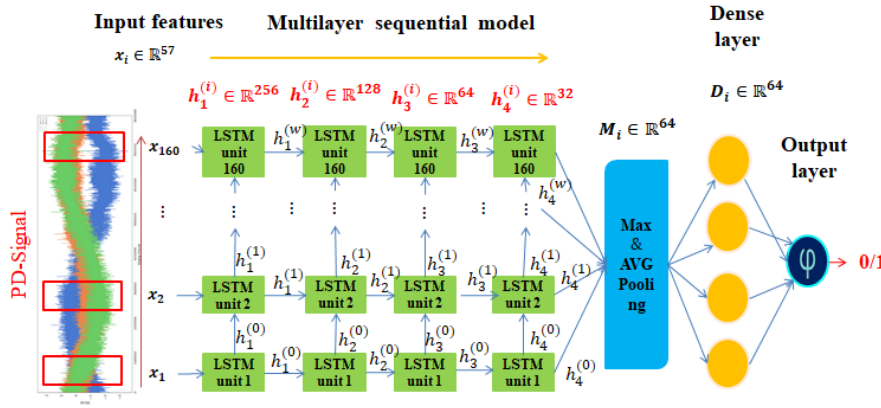
## 3.5 Learning models

In this section, we introduce the main learning models and hyperparameters that we have used to train our deep models and traditional models. First, we present the basic architectures that we have invoked in our implementation.

### 3.5.1 Sequential models

The original LSTM model consists of a single hidden LSTM layer followed by a standard output layer, the Stacked LSTM is an extension to this model that has multiple hidden LSTM layers where each layer contains multiple memory cells, the number of cells is defined mathematically as the power of 2, the most common number of cells used is 128 and 64, too many can over fit and too few can under fit. The layers are organized in a decreasing manner according to the number of cells. The architecture below corresponds to the LSTM model that we have implemented with four stacked

layers. As shown in Figure 3.3, the model takes an input of features of shape (160, N) the input shape varies according to the feature type;  $N \in \{57, 19, 76\}$ . If  $N=57$ , then the model is trained on the Stat features, while if  $N=19$ , then it takes as input FFT features, finally, if  $N=76$ , the model is trained on both feature sets.



**Figure 3.3: Sequential model implementation architecture.**

In order to perform learning, we have used the common error estimation function **Binary Cross Entropy / Log Loss**: this method compares the predicted probabilities to the actual class output, which can be either 0 or 1. It then computes a penalty for each probability based on how near or far it was from the expected value. The function is given by the following formula: where  $N$  is the number of samples represents the predicted probability and  $p(y_i)$  is the output.

$$LL = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) + (1 - y_i) \log(p(1 - y_i)). \quad (3.1)$$

We have used a variant of gradient descent, the **Adam optimizer** in order to minimize the loss function. Finally, we perform thresholding on the output probability vector to produce the oracle  $Y = \{y^1, \dots, y^i, \dots\}$  outputs, with  $y^i = 1$  if the anomaly is present in signal  $i$ , and  $y^i = 0$  otherwise.

### 3.5.2 Traditional models

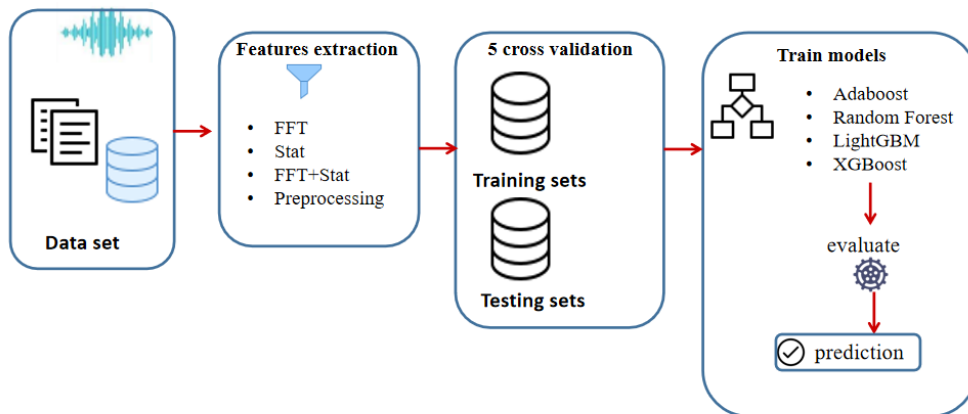
We have trained our models on various features ( FFT features , Stat features , FFT+ Stat and preprocessed ) using AdaBoost, Random forest, LightGBM and XGBoost classifier while varying some hyper-parameters such as learning rate, number of tree and number of leaves. We have invoked the Scikit-Learn library for implementing these models. Each model takes as input a vector of data with a size of  $2904 \times N$ , with  $N \in \{57, 19, 76, 9\}$ . If  $N=57$ , then the model is trained on the Stat features, while if  $N=19$ , then it takes as input FFT features and if  $N=76$ , the model is trained on both feature sets, finally, if  $n=9$  the model is trained on preprocessing features. The figure 3.4 shows the steps to train our models.



Our classifier is divided into the learning phase and testing phase. In the learning phase, our model is trained to learn from our train set. First we apply cross validation with 5 splits then we fit our model. But in the test phase the model predicts the label of unseen signals from the test set. Table 3.1 gives a summary of these learning algorithms and their hyper-parameter settings.

**Table 3.1: Traditional models parameters set.**

| Classifiers          | Parameters            | values |
|----------------------|-----------------------|--------|
| <b>AdaBoost</b>      | n-estimators          | 10000  |
| <b>Random Forest</b> | n-estimators          | 10000  |
|                      | max_features          | 19     |
|                      | max-depth             | 11     |
| <b>LightGBM</b>      | n-estimators          | 10000  |
|                      | learning rate         | 0.01   |
|                      | num-leaves            | 30     |
|                      | max-depth             | 300    |
|                      | type-boosting         | GDBT   |
|                      | early_stopping_rounds | 100    |
| <b>XGBoost</b>       | n_estimators          | 10000  |
|                      | max_depth             | 5      |
|                      | learning rate         | 0.01   |



**Figure 3.4: training traditional models process**

### 3.6 Performance evaluation

In order to assess the performance of our detectors, we have used one of the common performance metrics, the Matthew Correlation Coefficient; known as MCC score (please refer to section 2.2.2 for more details). In addition, we have performed a stratified 5-cross validation to split the training set into non overlapping training and validation sets. We have used the training set for

learning the models, whereas the validation set was employed for estimating the generalization performance, obtaining at the end 5 trained models and 5 MCC scores. In case of AdaBoost and Random Forest, we report the average over these five measurements, whereas in case of LightGBM, XGBoost, RNN and LSTM variants, we report the model with the best score, i.e. the validation set was used for model selection.

In order to obtain a reliable estimate of the generalization performance and avoid scores affected by random behavior of certain libraries, we have repeated the above steps 10, and report the average and standard deviation of these 10 runs.

### **3.7 Conclusion**

In this chapter, we have described the setup used to devise our fault detectors. We have presented the VSB data set, the features and classifiers used in our development. In the following chapter, we will present the results of these experiments and analyze them in order to derive conclusions.

## Chapter 4 : Experimental Results and Discussion

### 4.1 Introduction

In this Chapter, we present the partial discharge detection systems that we have implemented. We first investigate the sequential modeling of the power line signals. We have used namely: LSTM, Bi-LSTM, RNN and GRU, trained on various sets of features. Then, we examine traditional yet powerful learning algorithms; specifically, ensemble learning. We invoke AdaBoost, Random Forest, two Gradient Boosting Machines Light GBM and XGBoost.

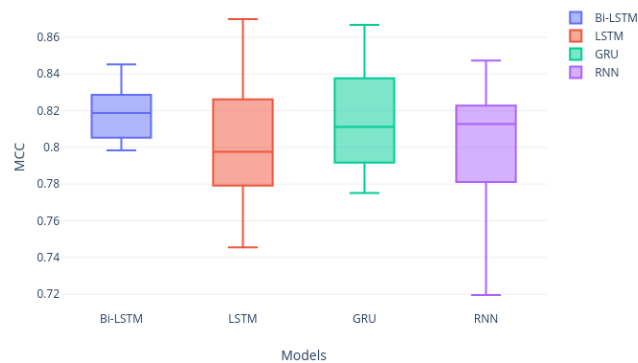
### 4.2 Experiment set 1: Sequential modeling

#### 4.2.1 Experiment 1: LSTM vs Bi-LSTM vs RNN vs GRU

The main goal of this experiment is to analyze the effect of the gating memory mechanism of the sequential models, namely GRU and LSTM. To this end, we have built four detection systems trained only on the statistical feature set using the ADAM optimizer with, LR=0.01, batch size = 128. We have set the number of epochs to 50. Table 4.1 gives the averaged MCC results over 10 runs of each system. We also report in Figure 4.1 the whisker plot of the compared models.

**Table 4.1: MCC score (%) of the gated memory systems.**

| Bi-LSTM         | LSTM            | GRU             | RNN             |
|-----------------|-----------------|-----------------|-----------------|
| 81.87% $\pm$ 2% | 80.06% $\pm$ 4% | 81.46% $\pm$ 3% | 79.98% $\pm$ 4% |



**Figure 4.1: LSTM vs Bi-LSTM vs RNN vs GRU Wicker plot representation.**

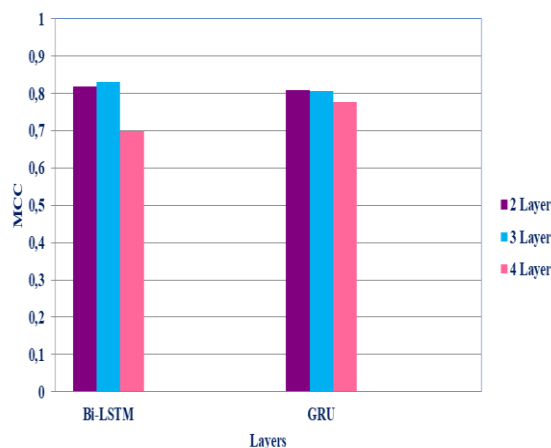
The analysis of the above results indicates that, overall, all four systems yield comparable correlation scores, with some little variations. Specifically, the table reveals that Bi-LSTM performance is in the lead followed by GRU and LSTM, while RNN provides the worst scores. This latter observation is expected since RNNs suffer from the vanishing gradient problem [75], hence, they are not capable of modeling long sequences of data. Therefore, we can conclude that introducing the gating mechanism positively impacts the detection performance. Furthermore, the whisker plot shows that the 10 Bi-LSTM scores have less variations (condensed box) compared to its counterparts. Consequently, Bi-LSTM provides a more stable behavior.

#### 4.2.2 Experiment 2: Impact of the number of stacked layers

Following our previous experiment, we have found that Bi-LSTM and GRU models yield the best predictive score. As a result, we have carried out the remaining investigations using only these models. Recall that the number of stacked layers defines the depth and complexity of sequential architectures. To further study how this hyper parameter affects the detection scores, we have trained several Bi-LSTM and GRU models, while varying the number of stacked layers from 2 to 4. Also, we have set the parameters of the optimizer to the same values as the previous experiment; and have conducted 10 runs of each system using the **statistical feature set**. The averaged-scores over these runs are shown by Table 4.2 and Figure 4.2.

**Table 4.2: MCC score (%) of Bi-LSTM/GRU for multiple stacked layers**

|                 | Bi-LSTM          | GRU              |
|-----------------|------------------|------------------|
| <b>2 Layers</b> | <b>81.87%±2%</b> | 81.46%±3%        |
| <b>3 Layers</b> | <b>82.75%±3%</b> | 80.60%±1%        |
| <b>4 Layers</b> | 70.26%±9%        | <b>77.69%±3%</b> |



**Figure 4.2: multiple stacked-layers models bar-plots.**

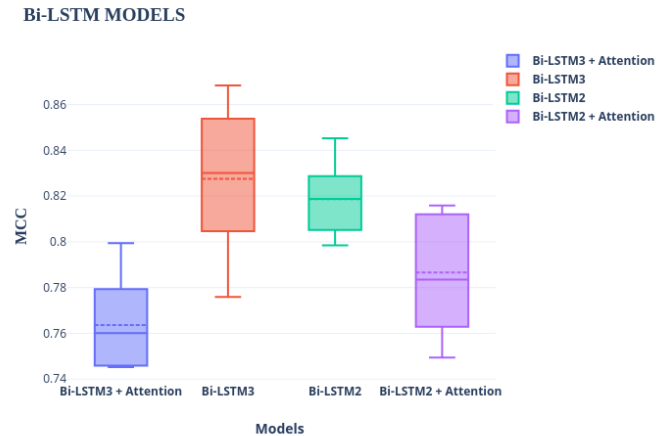
The above plots indicate that systems with 2 and 3 stacked layers surpass 4-layer models for both Bi-LSTM and GRU. This latter observation suggests that deeper networks provide poor performance when compared to a shallower architecture. However, it is widely acknowledged that deep models usually generalize better [85], while requiring more training epochs. Note that in our experiment, we have set the number of epochs to 50. We believe that increasing the training epochs could improve the performance of 4-layer models (Bi-LSTM4 and GRU4).

### 4.2.3 Experiment 3: Impact of the attention mechanism

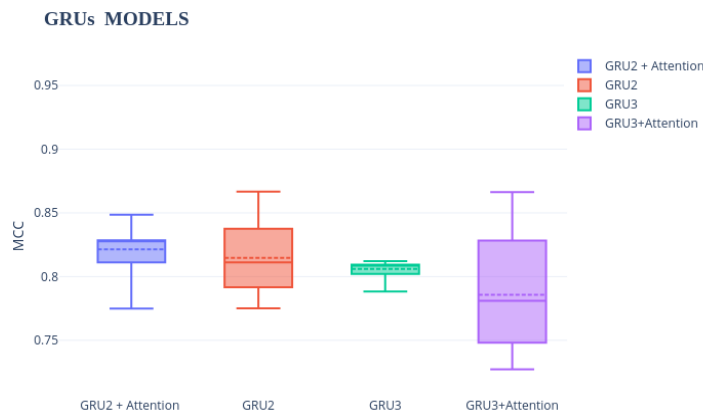
Attention-equipped LSTM models have been extensively used to improve the performance on complex sequence modeling tasks, specifically Natural Language Processing [86]. They provide a weighted focus on a part of the text. Attention was originally introduced as a solution to address the main issue surrounding Seq-to-Seq models [86]. In case of power line signals, the attention mechanism provides the model with an ability to focus on the most effective parts of features that characterize a fault or a partial discharge. Anomaly frames should receive higher weights; hence, the model is able to detect a partial discharge present in one or fewer frames of the signal. To better understand the impact of the attention mechanism, we have compared 4 sequence models, namely GRU2, GRU3, Bi-LSTM2 and Bi-LSTM3, with and without introducing the attention layer using the same parameter settings as in the previous experiment. We report in the Table below and Figures 4.3, 4.4 the results of this study.

**Table 4.3: MCC score (%) of sequential systems with/without Attention mechanism.**

|                  | <b>With Attention</b> | <b>Without Attention</b> |
|------------------|-----------------------|--------------------------|
| <b>Bi-LSTM 3</b> | 76.36% $\pm$ 2%       | 82.75% $\pm$ 3%          |
| <b>Bi-LSTM2</b>  | 78.66% $\pm$ 3%       | 81.87% $\pm$ 2%          |
| <b>GRU2</b>      | 82.86% $\pm$ 1%       | 81.46% $\pm$ 3%          |
| <b>GRU3</b>      | 78.58% $\pm$ 5%       | 80.60% $\pm$ 1%          |



**Figure 4.3: Attention impact on Bi-LSTM variants.**



**Figure 4.4: Attention impact on GRU variants.**

Figure 4.3 reveals that Bi-LSTM3 yields the highest averaged correlation scores followed by Bi-LSTM2, while Bi-LSTM3+Attention produces the worst performance. Therefore, the introduction of Attention does not demonstrate a positive impact on Bi-LSTM. However, as shown by Figure 4.4, attention-equipped GRUs exhibits a different behavior. Specifically, on one hand, GRU2+Attention attain better scores than GRU2 (without Attention); on the other hand, GRU3 surpasses (without attention) GRU3+Attention, by a 2% margin. In addition, the whisker plot shows that the 10 GRU3+Attention scores have more variations (large box) compared to its counterpart GRU3. Consequently, GRU3 without Attention delivers a more stable behavior.

The previous observations suggest that overall the non-attention models achieve better scores than the attention-equipped detectors. This behavior is not expected since attention models are complex but generally very effective and accurate [78]. Many reasons may cause this behavior. It can be related to the hyper parameters used for training such as the batch size, the learning rate, the number of epochs, etc.

To further investigate this issue, we have conducted the following experiment. We have increased the number of training epochs of Bi-LSTM3+Attention, Bi-LSTM2+Attention, GRU3+Attention and GRU2+Attention from 50 to 200; and measured the MCC on the validation fold. The averaged-results are given in Table 4.4.

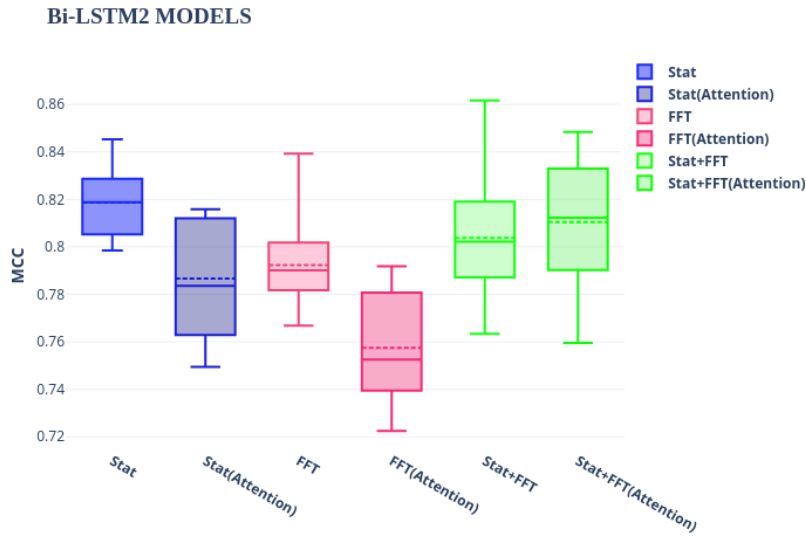
**Table 4.4: MCC results (%) of Bi-LSTM/GRU trained for 200 epochs.**

| <b>Bi-LSTM2</b> | <b>Bi-LSTM3</b> | <b>GRU2</b> | <b>GRU3</b> |
|-----------------|-----------------|-------------|-------------|
| 80.93%±3%       | 81.34%±1%       | 85.21%±1%   | 81.99%±4%   |

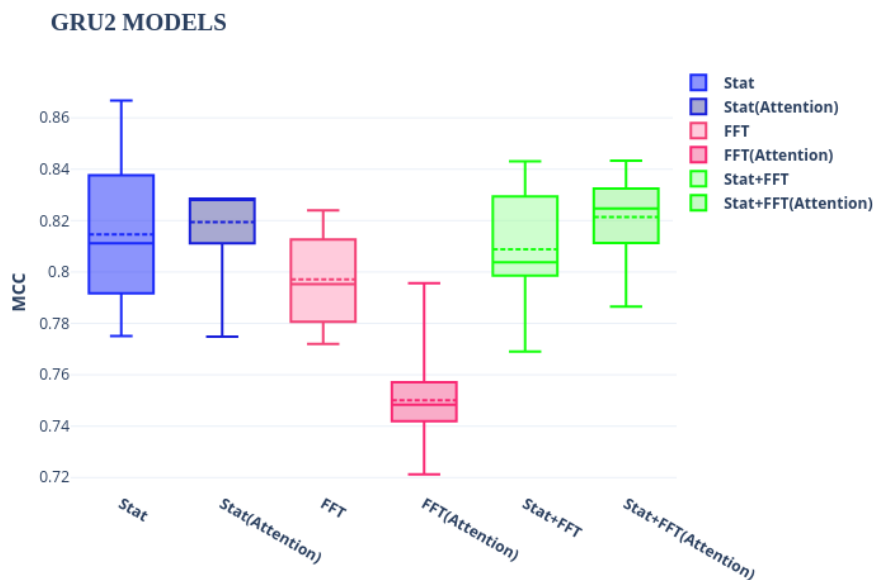
The analysis of the above table indicates the performance of attention models exhibits an improvement as the number of epochs increases. We observe a rise of 3% between epoch 50 and epoch 200. This latter finding coincides with our initial claim regarding the parameters setting for training attention-equipped detectors. We can conclude that attention-based systems need to be trained for longer runs in order to provide state-of-the-art performance.

#### **4.2.4 Experiment 4: Statistical vs FFT features**

Recall that the main goal of this thesis is to design a system for detecting partial discharge in power line signals. In Chapter 1, we have introduced various categories of feature representation, namely statistical and FFT features. Please refer to Section 1.5.2 for additional details on this subject. So far, we have only considered the Statistical feature set to train our detection models. A natural extension to this would be to explore other feature representations. To this end, we have built numerous detection systems based on GRU and LSTM architectures, while varying some hyper parameters and the mechanism used for extracting the features. Specifically, we have trained four models, namely GRU2, GRU2+Attention, Bi-LSTM2 and Bi-LSTM2+Attention, on three sets of features: Statistical, FFT and Statistical + FFT, resulting in 12 systems. Note that we have set the parameters of the optimizer to the same values as the previous experiment; and have conducted 10 runs of each system. The scores of these runs are shown by Figures 4.5, 4.6. Furthermore, the averaged-scores over these runs are reported in Table 4.5.



**Figure 4.5: MCC scores for Bi-LSTM2 model trained on multiple features set.**



**Figure 4.6: MCC scores for GRU models trained on multiple features set.**

**Table 4.5: MCC score (%) of deep systems tested on different set of features.**

|                             | Stat Features | FFT-Features | Stat +FFT Features |
|-----------------------------|---------------|--------------|--------------------|
| <b>Bi-LSTM2</b>             | 81.87%±2%     | 79.23% ±2%   | 80.38%±3%          |
| <b>Bi-LSTM2 + Attention</b> | 78.66%±3%     | 75.74% ±2%   | 81.04%±3%          |
| <b>GRU2</b>                 | 81.46%±3%     | 79.71% ±2%   | 80.88%±2%          |
| <b>GRU2 + Attention</b>     | 82.86%±2%     | 75.01% ±2%   | 82.14%±2%          |

The examination of the above results can be summarized as follows. Systems that were designed using statistical features outperform those trained on the FFT feature set. Moreover, Bi-



LSTM2 and GRU2 trained on Stat features achieve better scores than the ones built using both FFT and Stat features. Therefore, we can conclude that the introduction of the FFT features [does not demonstrate a positive impact] on the performance of non-attention based-models. However, in case of attention-equipped models, incorporating the FFT feature set maintains or even improves the detection scores.

#### 4.2.5 Summary: Evaluation on the test set (private / public score)

In order to evaluate the efficiency of our detection systems, we have measured the MCC scores on the challenge test set. Note that the true class labels of the test signals were not available for download during the development stage of our systems. In order to obtain the MCC scores on the test set, we have proceeded with submitting our models predictions on the test set through the VSB challenge website/evaluation module. This latter provides two types of scores: private and public scores. The private score  $Private_{MCC}$  is calculated with approximately **43%** of the test data; while the public score  $Public_{MCC}$  uses **57%** of the test set. The equation for measuring the final score is given by:

$$Final_{MCC} = Private_{MCC} * 0.43 + Public_{MCC} * 0.57. \quad (4.1)$$

We have designed several systems, namely: Bi-LSTM2+Attention, Bi-LSTM2, GRU2, GRU2+Attention trained on 3 feature sets (stat, FFT, stat + FFT). For each system, we have first resampled the training (development) set following 5-fold cross-validation, resulting in 5 training and 5 validation sets, denoted  $train_i$  and  $Val_i$ , where  $i = 1, \dots, 5$ . Then, we have trained the models on the training fold  $train_i$  for 50 epochs; and have selected the one with the highest score on the  $Val_i$  set. This process was repeated 5 times; and at the end, we obtained 5 best trained models. Note that the output consists of a vector of probabilities assigned to each measurement of our input signals. In order to obtain both private and public scores on the test set and perform a challenge submission, we must compute class labels (labels i.e. 1 or 0). To do so, we have performed thresholding on the probability outputs. It is worth mentioning that we have determined the best threshold using the whole development set. The results of the submissions are given in Tables ( 4.6, 4.7, 4.8)

**Table 4.6: Submission scores (%) of four sequential systems with Stat features.**

|                      | Stat Features  |        |                      |               |
|----------------------|----------------|--------|----------------------|---------------|
|                      | GRU2+Attention | GRU2   | Bi-LSTM2 + Attention | Bi-LSTM2      |
| <b>Private score</b> | 60.41%         | 60.6%  | 58.61%               | 63.43%        |
| <b>Public score</b>  | 63.84%         | 61.6%  | 63.81%               | 61.22%        |
| <b>Final score</b>   | <b>62.37%</b>  | 61.17% | 61.57%               | <b>62.17%</b> |

**Table 4.7: Submission scores (%) of four sequential systems with FFT features.**

|                      | FFT features     |        |                      |          |
|----------------------|------------------|--------|----------------------|----------|
|                      | GRU2 + Attention | GRU2   | Bi-LSTM2 + Attention | Bi-LSTM2 |
| <b>Private score</b> | 63.21%           | 55.19% | 60.34%               | 54.43%   |
| <b>Public score</b>  | 53.60%           | 60.43% | 55.31%               | 52.60%   |
| <b>Final score</b>   | 57.73%           | 58.17% | 57.47%               | 53.39%   |

**Table 4.8: Submission scores (%) of four sequential systems with Stat+FFT features.**

|                      | Stat + FFT features |        |                      |          |
|----------------------|---------------------|--------|----------------------|----------|
|                      | GRU2 + Attention    | GRU2   | Bi-LSTM2 + Attention | Bi-LSTM2 |
| <b>Private score</b> | 50.28%              | 51.01% | 59.92%               | 56.35%   |
| <b>Public score</b>  | 50.07%              | 55.54% | 55.10%               | 54.43%   |
| <b>Final score</b>   | 50.16%              | 53.59% | 57.18%               | 55.25%   |

The analysis of the final scores indicates that GRU2+Attention and Bi-LSTM2 (without Attention) trained on the Stat features outperform the other alternatives by a large margin. This observation confirms our previous results: (1) FFT features deteriorate the generalization performance; (2) GRU performs better when equipped with the attention mechanism, while Bi-LSTM provides higher scores without incorporating the attention layer.

We observe that the best two models GRU2+Attention and Bi-LSTM2, trained on Stat features, provide complementary scores. Therefore, it would be beneficial to create an ensemble that combines the predictions of several models. To this end, we have first selected four models having the highest scores; then, we have tested some pairwise combinations: **GRU2+Attention & Bi-LSTM2**, **GRU2 & Bi-LSTM + Attention**. In order to produce the ensemble outputs, we have averaged the predictions of both learners and performed thresholding; we have set the threshold value TH to 0.5. Note that each learner was trained 5 times since we have followed 5-fold CV, resulting in a committee made of 10 classifiers. The results are summarized in the following table:

**Table 4.9: Submission scores (%) of combined systems.**

|                      | <b>GRU2+Attention &amp; Bi-LSTM2</b> | <b>GRU2 &amp; Bi-LSTM2 + Attention</b> |
|----------------------|--------------------------------------|--|
| <b>Private score</b> | 63.07%                               | 61.94%                                 |
| <b>Public score</b>  | 65.50%                               | 65.29%                                 |
| <b>Final score</b>   | <b>64.45%</b>                        | <b>63.85%</b>                          |

### 4.3 Experiment set 2: Traditional machine learning models

#### 4.3.1 Experiment 1: Comparison AdaBoost, Random Forest, LightGBM and XGBoost

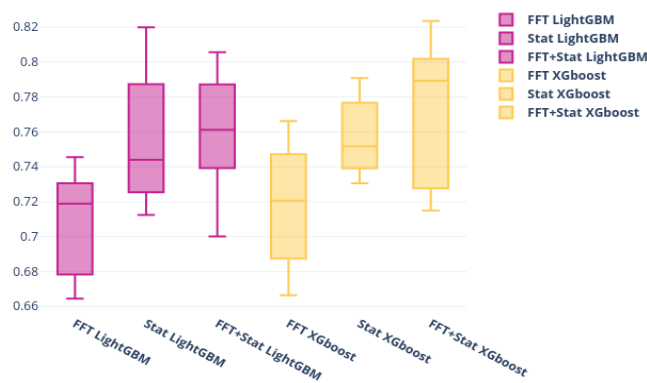
The main purpose of this experiment is to investigate traditional machine learning for partial discharge detection. To accomplish the ensuing task, we have invoked 4 classifiers namely: AdaBoost, Random Forest, LightGBM and XGBoost, trained on a combination of feature representations based on signal processing techniques and statistics. In Particular, we have examined 4 feature sets: **Stat**, **FFT**, **Stat + FFT** and **Preprocessing**. Note that this latter feature set has been inspired from the work of the challenge winners [87]. We obtain at the end 16 detection systems. Tables 4.10 gives the averaged MCC results over 10 runs of each system. We also report in Figure 4.7 and 4.8 the whisker plot of the compared models.

**Table 4.10:MCC scores (%) of Boosting machines.**

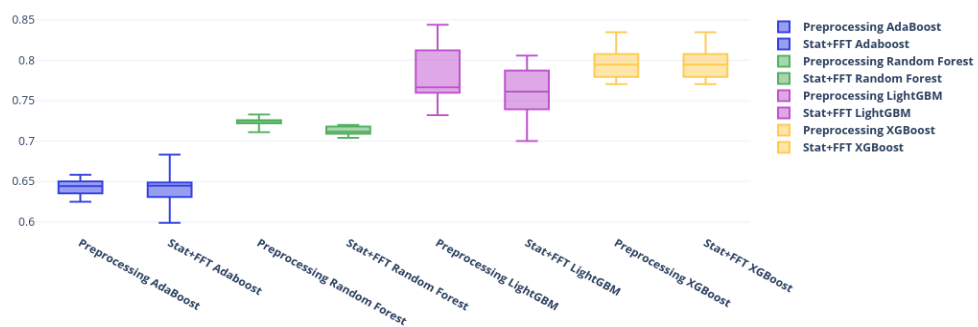
|                               | <b>AdaBoost</b> | <b>LightGBM</b> | <b>Random-Forest</b> | <b>XGBoost</b> |
|-------------------------------|-----------------|-----------------|----------------------|----------------|
| <b>FFT-features</b>           | 51.1%±2%        | 70.85%±3%       | 63.25%±1%            | 71.76%±3%      |
| <b>Stat-features</b>          | 62.27%±2%       | 75.68%±4%       | 67.88%±0.4%          | 75.81%±2%      |
| <b>FFT+Stat features</b>      | 64.22%±2%       | 76.15%±4%       | 71.20%±1%            | 77.24%±4%      |
| <b>Preprocessing features</b> | 64.22±1%        | 77.98%±4%       | 72.37%±1%            | 79.61%±2%      |



**Figure 4.7: MCC Score for AdaBoost and Random Forest.**



**Figure 4.8: MCC Score for LightGBM and XGBoost.**



**Figure 4.9: MCC Score for all classifiers trained on preprocessing and FFT+Stat.**

After examining the whisker plots given in Figure 4.7 and 4.8, we can derive the following main observations:

1. Gradient Boosting Machines (LightGBM and XGBoost) yield the best MCC scores compared to their counterparts AdaBoost and Random Forest. This is rather expected since Gradient Boosting Machines are simple yet powerful learning algorithms that provide state-of-the-art performance in many applications [88].
2. FFT feature set negatively impacts the performance of all detection systems, whereas systems trained on Stat achieve the highest performance results. Similar findings were also reported in our previous experiments on sequential modeling. A possible cause of this peculiar behavior might be related to the process of extracting the FFT features. In our experiments (3.4), in the case of Stat features, we have divided each signal into chunks of 1000 values, and then extracted various statistics from **each chunk** separately, whereas in case of FFT feature set, we have applied the fast fourier transform and Welch’s method on the **entire** signal to extract 19 signal features. Note that an anomaly, i.e. a partial discharge, might occur within a single chunk only; therefore, extracting FFT and Welch from the whole signal can lead to a feature representation that does not efficiently express the characteristics of an anomaly.
3. FFT+ Stat feature set-based detectors achieve comparable results to systems trained on Stat features; thus, the combined feature sets maintain the performance of Stat-based detectors. However, AdaBoost exhibits a different behavior. When trained on both Stat and FFT feature sets, its performance drops drastically.
4. Models trained on the preprocessed feature set exceed the other alternatives by a large margin.

Based on the above observations, we can conclude that Gradient Boosting Machines trained on Stat or pre-processed feature sets produce the best detection rates.

#### 4.3.2 Summary: Evaluation on the test set (private / public score)

In order to evaluate the efficiency of our traditional machine learning-based detectors, we have measured the MCC scores on the challenge test set. To achieve this, we have followed the same steps discussed in Section (4.2.5). The results of the submissions are given in Tables 4.11, 4.12, 4.13, 4.14). Table 4.11: Submission scores (%) of AdaBoost with four different feature sets.

| <b>Adaboost</b>      |                     |                      |                            |                               |
|----------------------|---------------------|----------------------|----------------------------|-------------------------------|
|                      | <b>FFT features</b> | <b>Stat features</b> | <b>FFT + Stat features</b> | <b>Preprocessing features</b> |
| <b>Private score</b> | 41.02%              | 34.66%               | 44.96%                     | 50.5%                         |
| <b>Public score</b>  | 17.62%              | 37.75%               | 30.77%                     | 46.64%                        |
| <b>Final score</b>   | 27.6%               | 36.42%               | 36.42%                     | 50.19%                        |

**Table 4.12: Submission scores (%) of Random Forest with four different feature sets.**

| <b>Random Forest</b> |                     |                      |                            |                               |
|----------------------|---------------------|----------------------|----------------------------|-------------------------------|
|                      | <b>FFT features</b> | <b>Stat features</b> | <b>FFT + Stat features</b> | <b>Preprocessing features</b> |
| <b>Private score</b> | 43.77%              | 54.39%               | 52.80%                     | 66.18%                        |
| <b>Public score</b>  | 21.44%              | 45.92%               | 37%                        | 54.84%                        |
| <b>Final score</b>   | 31.04%              | 49.56%               | 43.77%                     | 59.72%                        |

**Table 4.13 : Submission scores (%) of LightGBM with four different feature sets.**

| <b>LightGBM</b>      |                     |                      |                            |                               |
|----------------------|---------------------|----------------------|----------------------------|-------------------------------|
|                      | <b>FFT Features</b> | <b>Stat features</b> | <b>FFT + Stat features</b> | <b>Preprocessing features</b> |
| <b>Private score</b> | 58.74%              | 54.88%               | 55.29%                     | 68.44%                        |
| <b>Public score</b>  | 34.16%              | 51.75%               | 47.98%                     | 67.36%                        |
| <b>Final score</b>   | 44.72%              | 53.09%               | 51.12%                     | 67.82%                        |

**Table 4.14: Submission scores (%) of XGBoost with four different feature sets.**

| <b>XGBoost</b>       |                     |                      |                            |                               |
|----------------------|---------------------|----------------------|----------------------------|-------------------------------|
|                      | <b>FFT Features</b> | <b>Stat features</b> | <b>FFT + Stat features</b> | <b>Preprocessing Features</b> |
| <b>Private score</b> | 42.31%              | 35.93%               | 42.60%                     | 57.42%                        |
| <b>Public score</b>  | 14.03%              | 35.98%               | 29.16%                     | 55.61%                        |
| <b>Final score</b>   | 26.19%              | 35.95%               | 34.93%                     | 59.38%                        |

The submission scores given in the above table indicate that detectors that use the pre-processed feature set demonstrate superiority over the other alternatives, which coincides with our previous findings.

#### 4.4 Summary of experimental findings

Table 4.15 gives the submission scores of the top 3 detectors.

**Table 4.15: Top 3 submitted scores.**

|                      | <b>GRU2+Attention &amp; Bi-LSTM2</b> | <b>GRU2 &amp; Bi-LSTM2 + Attention</b> | <b>LightGBM</b> |
|----------------------|--------------------------------------|--|-----------------|
| <b>Private score</b> | 63.07%                               | 61.94%                                 | 68.44%          |
| <b>Public score</b>  | 65.50%                               | 65.29%                                 | 67.36%          |
| <b>Final score</b>   | 64.45%                               | 63.85%                                 | <b>67.82%</b>   |

The results indicate that LightGBM trained on preprocessed features achieves superiority over sequence models Bi-LSTM. Note that gradient boosting machine is a powerful yet simple learning algorithm that does not require dedicated hardware like GPUs, NPUs or TPUs. This finding coincides with the **Data-centric AI principle** [89]. Data-centric AI is the discipline of systematically engineering the data needed to successfully build an AI system. Consequently, the focus has to shift from big data to good data, i.e. **extracting meaningful features can considerably boost the overall performance**.

#### 4.5 Chapter summary

In this chapter, we have presented the results of our experimental enquiries. Several lessons can be derived from our analysis:

- Gate mechanism models (GRU and Bi-LSTM) demonstrated superiority over simple recurrent networks (RNN) on the detection performance.
- Attention-based systems need to be trained for longer runs in order to provide state-of-the-art performance.
- Classifiers trained on preprocessed features achieve a significant improvement in the detection performance compared to the other features.
- FFT features negatively impact all detection systems (sequence and traditional models), in contrast to statistical features.
- LightGBM achieves the best MCC score when compared to the other alternatives. Most importantly, when trained on preprocessed features, **LightGBM even surpasses the ensemble of sequence models Bi-LSTM (any feature set)**.

**In order to build a successful fault detector, the focus has to shift from model-centric to data-centric, i.e. understand the input training data and preprocess it.**

# Conclusion

## 1. Contributions and summary of experimental findings

In this thesis, the primal objective was to conduct an analysis and comparison among classification systems able to recognize partial discharge anomalies in power line signals. To this end, we conducted multiple experiments to analyze the behavior of anomaly detection systems. In particular, we have carried out 2 sets of experiments. First, we examined four sequential models: Recurrent neural network, Gated recurrent unit, Long short-time memory and Bidirectional long short-time memory. We trained these models on 3 different features, namely: statistical, signal processing-based features and a combination of these two sets of features. We also examined varying some of these learning models parameters. Second, we investigated four ensemble learning classifiers for building our detectors, namely: AdaBoost, Random Forest, LightGBM and XGBoost trained on the aforementioned feature sets. Most importantly, we analyzed the effect of some preprocessing strategies on the overall detection performance. The experimental investigation indicates the following:

- The training of sequential and ensemble learning models on FFT features deteriorates the detection scores, while statistical features demonstrate a positive impact.
- The partial discharge detectors trained using LSTM and its variants give very good results compared to RNN. Moreover, increasing the number of layers of these architectures does not significantly improve the overall detection rates.
- The attention-based architectures usually are complex and encompass many parameters that need to be trained. Therefore, we can conclude that attention-based detectors need to be trained for longer runs in order to improve the MCC scores.
- The anomaly detection system based on gradient boosting machine classifiers (LightGBM and XGBoost) trained on preprocessed signals yields the best performance in terms of both detection scores and training time. Consequently, we can conclude that a higher number of features is not always beneficial and can negatively affect the performance of the classification system. Most importantly, LightGBM even surpasses the ensemble of sequence models Bi-LSTM (any feature set). We can conclude that in order to build a **successful fault detector, the focus has to shift from model-centric to data-centric**, i.e. understand the input training data and preprocess it.

## 2. Limits and future work

One extension of this work would be testing other advanced neural network architectures such as Convolutional Recurrent Neural Network (CRNN), which involves a CNN (Convolutional Neural



Network) followed by a RNN. CRNNs have demonstrated state-of-the-art performance in many time-series applications like Sound event detection [90] and Video monitoring [91], since convolution layers learn complex feature representations that well characterize partial discharge faults [92]. Another future work direction involves testing dimensionality reduction techniques like Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) [93]. Dimensionality reduction is one of the techniques widely used by data scientists while performing feature engineering. The goal is to provide a better representation of the feature space, hence, a proper characterization of faults. Another appealing work direction would be to study other feature selection techniques like Wavelets [94].

During this work, we have faced several difficulties. The training of the learning models took a very long time since we trained them using Kaggle GPU. Unfortunately, the platform provides us 30 hours of GPU per week, which prevents us from conducting further experimental investigations. In addition, we have had difficulties in all the electronic field (signal processing, fast fourier transform, etc.) as it was the first time we worked on it.

We have gained extensive knowledge and skills working on this project. We have discovered the Kaggle platform and its interesting competitions. In addition, we have improved our programming skills in Python. Moreover, we have learned the most important steps to conduct machine learning and deep learning projects, while performing tests using many classifiers.

## References

- [1] Xu, Xun, Yuqian Lu, Birgit Vogel-Heuser, et Lihui Wang. « Industry 4.0 and Industry 5.0—Inception, conception and perception ». *Journal of Manufacturing Systems* 61 (2021): 530-35. <https://doi.org/10.1016/j.jmsy.2021.10.006>.
- [2] V. V Martynov, D. N. Shavaleeva, and A. A. Zaytseva, “Information Technology as the Basis for Transformation into a Digital Society and Industry 5.0,” in 2019 International Conference “Quality Management, Transport and Information Security, Information Technologies” (IT QM 3 IS), 2019, pp. 539–543, doi: 10.1109/ITQMIS.2019.8928305.
- [3] « Introduction to Machine Learning, fourth edition - Ethem Alpaydin - Google Livres ». Consulté le 17 juin 2022.
- [4] Bhattacharyya, Pushpak, Hanumat G. Sastry, Venkatadri Marriboyina, et Rashmi Sharma. Smart and Innovative Trends in Next Generation Computing Technologies: Third International Conference, NGCT 2017, Dehradun, India, October 30-31, 2017, Revised Selected Papers, Part I. Springer, 2018.
- [5] Alsina-Pagès, Rosa Ma, Patrizia Bellucci, et Giovanni Zambon. Smart Wireless Acoustic Sensor Network Design for Noise Monitoring in Smart Cities. MDPI, 2021.
- [6] « Smart Grid Sensors: Principles and Applications - Hamed Mohsenian-Rad - Google Livres ». Consulté le 17 juin 2022.
- [7] Damir Timotijevic, “Anomaly Detection inTime-Series”, LU-CS-EX 2020-57 DEPARTMENT OF COMPUTER SCIENCE LTH j LUND UNIVERSITY
- [8] Agrawal, Shikha, et Jitendra Agrawal. « Survey on Anomaly Detection Using Data Mining Techniques ». *Procedia Computer Science, Knowledge-Based and Intelligent Information & Engineering Systems 19th Annual Conference, KES-2015, Singapore, September 2015 Proceedings*, 60 (1 janvier 2015): 708-13. <https://doi.org/10.1016/j.procs.2015.08.220>.
- [9] S. A. Boggs, "Partial discharge: overview and signal generation," in *IEEE Electrical Insulation Magazine*, vol. 6, no. 4, pp. 33-39, July-Aug. 1990, doi: 10.1109/57.63057 consulté 18 juin 2022.
- [10] Kawaguchi, Y.; Yanabu, S. Partial-discharge measurement on high-voltage power transformers. *IEEE Trans. Power Appar. Syst.* 1969, 88, 1187–1194.
- [11] Ibrahim, K.; Sharkawy, R.; Salama, M.; Bartnikas, R. Realization of partial discharge signals in transformer oils utilizing advanced computational techniques. *IEEE Trans. Dielectr. Electr. Insul.* 2012, 19, 1971–1981.
- [12] Khan, Q.; Refaat, S.S.; Abu-Rub, H.; Toliyat, H.A. Partial discharge detection and diagnosis in gas insulated switchgear: State of the art. *IEEE Electr. Insul. Mag.* 2019, 35, 16–33.
- [13] McGreevy, D.; Giussani, R.; Seltzer-Grant, M.; Singh, A.; Patel, A.; Calladine, S.; Gibb, G. Deployment of an online partial discharge monitoring system for power station with focus on gas turbine generators. In *Proceedings of the 2017 INSUCON-13th International Electrical Insulation Conference, Birmingham, UK, 16–18 May 2017*; pp. 1–5.

- [14] Misák, S.; Fulnecek, J.; Vantuch, T.; Buriánek, T.; Jezowicz, T. A complex classification approach of partial discharges from covered conductors in real environment. *IEEE Trans. Dielectr. Electr. Insul.* 2017, 24, 1097–1104.
- [15] <https://arxiv.org/ftp/arxiv/papers/1905/1905.01588.pdf> consulté le 20 juin 2022
- [16] ichau, G.; Hsu, C.-C.; Fink, O. Interpretable Detection of Partial Discharge in Power Lines with Deep Learning. *Sensors* 2021, 21, 2154. <https://doi.org/10.3390/s21062154>
- [17] <https://www.kaggle.com/ratthachat/workshop-lstm> consulté 25 juin 2022
- [18] Goldberg, David, et Yinan Shan. « The Importance of Features for Statistical Anomaly Detection ». In 7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15). Santa Clara, CA: USENIX Association, 2015.
- [19] R. Zhao, B. Du, L. Zhang and L. Zhang, "Beyond Background Feature Extraction: An Anomaly Detection Algorithm Inspired by Slowly Varying Signal Analysis," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 3, pp. 1757-1774, March 2016, doi: 10.1109/TGRS.2015.2488285.
- [20] IEC60270 “Partial Discharge Measurements” consulté le 11 mai 2022
- [21] Gabriel Michau, Chi-Ching Hsu, and Olga Fink , 2021 Mar 19, « Interpretable Detection of Partial Discharge in Power Lines with Deep Learning” , *Sensors* (Basel). , <https://www.mdpi.com/journal/sensors>
- [22] Greg Stone , “Partial Discharges in Electrical Insulation” , 2015
- [23] Karimi-Ghartemani, M., et M.R. Iravani. « A nonlinear adaptive filter for online signal analysis in power systems: applications ». *IEEE Transactions on Power Delivery* 17, n° 2 (avril 2002): 617-22. <https://doi.org/10.1109/61.997949>.
- [24] ATASPINAR. « Machine Learning with Signal Processing Techniques ». 4 avril 2018. <https://ataspinar.com/2018/04/04/machine-learning-with-signal-processing-techniques/>.
- [25] Karimi-Ghartemani, M., et M.R. Iravani. « A nonlinear adaptive filter for online signal analysis in power systems: applications ». *IEEE Transactions on Power Delivery* 17, n° 2 (avril 2002): 617-22. <https://doi.org/10.1109/61.997949>
- [26] Deng, Shi-Wen, et Ji-Qing Han. « Ramanujan Subspace Pursuit for Signal Periodic Decomposition ». *Mechanical Systems and Signal Processing* 90 (juin 2017): 79-96. <https://doi.org/10.1016/j.ymssp.2016.12.020>.
- [27] W. S. Gan, *Signal Processing and Image Processing for Acoustical Imaging*, [https://doi.org/10.1007/978-981-10-5550-8\\_5//13//](https://doi.org/10.1007/978-981-10-5550-8_5//13//)
- [28] Kim, Jiil, et Cheong Hee Park. « Partial Discharge Detection Based on Anomaly Pattern Detection ». *Energies* 13, n° 20 (janvier 2020): 5444. <https://doi.org/10.3390/en13205444>.
- [29] Serizel, R., Bisot, V., Essid, S., & Richard, G. (2017). Acoustic Features for Environmental Sound Analysis

- [30] Shang, Haikun, Kwok Lun Lo, et Feng Li. « Partial Discharge Feature Extraction Based on Ensemble Empirical Mode Decomposition and Sample Entropy ». *Entropy* 19, n° 9 (2017). //9//
- [31] Delimayanti, Mera Kartika, Bedy Purnama, Ngoc Giang Nguyen, Mohammad Reza Faisal, Kunti Robiatul Mahmudah, Fatma Indriani, Mamoru Kubo, et Kenji Satou. « Classification of Brainwaves for Sleep Stages by High-Dimensional FFT Features from EEG Signals ». *Applied Sciences* 10, n° 5 (janvier 2020): 1797. <https://doi.org/10.3390/app10051797>. //12//
- [32] Ambaye, Getachew Admassie. « Time and Frequency Domain Analysis of Signals: A Review ». *International Journal of Engineering Research* 9, n° 12 (s. d.): 6 //10//
- [33] Youngworth, Richard N., Benjamin B. Gallagher, et Brian L. Stamper. « An overview of power spectral density (PSD) calculations ». édité par H. Philip Stahl, 58690U. San Diego, California, USA,
- [34] Youngworth, Richard N., Benjamin B. Gallagher, et Brian L. Stamper. « An overview of power spectral density (PSD) calculations ». édité par H. Philip Stahl, 58690U. San Diego, California, USA, 2005. <https://doi.org/10.1117/12.618478>.
- [35] Parhi, K. K., & Ayinala, M. (2014). *Low-Complexity Welch Power Spectral Density Computation. IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(1), 172–182. doi:10.1109/tcsi.2013.2264711
- [36] « VSB - Technical University of Ostrava - VSB-TUO ». Consulté le 30 mai 2022. <https://www.vsb.cz/en>. //8 //
- [37] Meste, O., H. Rix, P. Caminal, et N.V. Thakor. « Ventricular late potentials characterization in time-frequency domain by means of a wavelet transform ». *IEEE Transactions on Biomedical Engineering* 41, n° 7 (juillet 1994): 625-34. <https://doi.org/10.1109/10.301729>
- [38] Allan, D.W., M.A. Weiss, et J.L. Jespersen. « A frequency-domain view of time-domain characterization of clocks and time and frequency distribution systems ». In *Proceedings of the 45th Annual Symposium on Frequency Control 1991*, 667-78. Los Angeles, CA, USA: IEEE, 1991. <https://doi.org/10.1109/FREQ.1991.145966>
- [39] Too, J., A. R. Abdullah, T. N. S. Tengku Zawawi, N. Mohd Saad, et H. Musa. « Classification of EMG Signal Based on Time Domain and Frequency Domain Features ». *International Journal of Human and Technology Interaction (IJHaTI)* 1, n° 1 (31 octobre 2017): 25-30.
- [40] D. Zekrif, Intelligent Systems: Current Progress. 2017
- [41] «Apprentissage automatique», 20 décembre 2021. <https://www.ibm.com/fr-fr/analytics/machine-learning>.
- [42] P. Dönmez, “Introduction to Machine Learning, 2nd ed., by Ethem Alpaydm. Cambridge, MA: The MIT Press2010. pages.” *Nat. Lang. Eng.*, vol. 19, no. 2, pp. 285–288, 2013, doi: 10.1017/s1351324912000290.

- [43] Cunningham, P., Cord, M., Delany, S.J. (2008). Supervised Learning. In: Cord, M., Cunningham, P. (eds) Machine Learning Techniques for Multimedia. Cognitive Technologies. Springer, Berlin, Heidelberg. [https://doi.org/10.1007/978-3-540-75171-7\\_2](https://doi.org/10.1007/978-3-540-75171-7_2)
- [44] S. Das, A. dey, A. Pal, and N. Roy, “Applications of Artificial Intelligence in Machine Learning: Review and Prospect,” *Int. J. Comput. Appl.*, vol. 115, pp. 31–41, Apr. 2015, doi: 10.5120/20182-2402
- [45] D. Sharma and N. Kumar, “A Review on Machine Learning Algorithms, Tasks and Applications,” *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 6, no. 10, pp. 1323–2278, Oct. 2017.(43)
- [46] Y. C A Padmanabha Reddy, P. Viswanath, and B. Eswara Reddy, “Semi-supervised learning: a brief review,” *Int. J. Eng. Technol.*, vol. 7, no. 1.8, p. 81, 2018, doi: 10.14419/ijet.v7i1.8.9977.
- [47] X. Goldberg, Introduction to semi-supervised learning, vol. 6. 2009.
- [48] Bechhofer, S., M. Hauswirth, J. Hoffmann, et M. Koubarakis. The Semantic Web: Research and Applications: 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008. <https://books.google.dz/books?id=-MpsCQAAQBAJ>.
- [49] Soetomo, M.A.A. Performance Analysis in Binary Classification Using Interval Scale Data as Samples, and Discriminant Analysis, Logistic Regression and Artificial Neural Network as Decision Models. George Washington University, 2002.
- [50] Chicco, D., Jurman, G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics* **21**, 6 (2020). <https://doi.org/10.1186/s12864-019-6413-7>
- [51] Moudiki, T. (2020). Linear model, xgboost and randomForest cross-validation using crossval:crossval\_ml. In *R-bloggers: R news and tutorials contributed by hundreds of bloggers*.
- [52] Goldstein, Markus, et Seiichi Uchida. « A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data ». *PLOS ONE* 11, n° 4 (19 avril 2016): e0152173.
- [53] Kumari, Roshan & Srivastava, Saurabh. (2017). Machine Learning: A Review on Binary Classification. *International Journal of Computer Applications*. 160. 11-15. 10.5120/ijca2017913083.
- [54] Alpaydin, E. (2009). Introduction to machine learning. MIT press.
- [55] Lior Rokach. Ensemble learning ,Israel,second edition,2019,vol 85
- [56] Duda, R. O., Hart, P. E., & Stork, D. G. (2000). Pattern classification. Second edition. N'ju-Jork.

- [57] Lu, Liang, Yaokai Feng, et Kouichi Sakurai. « *C&C Session Detection Using Random Forest* ». In *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, 1-6. Beppu Japan: ACM, 2017. <https://doi.org/10.1145/3022227.3022260>.
- [58] Vo, D. M., Nguyen, N.-Q., & Lee, S.-W. (2019). Classification of Breast Cancer Histology Images using Incremental Boosting Convolution Networks. *Information Sciences*. doi:10.1016/j.ins.2018.12.089
- [59] Touzani, S., Granderson, J., & Fernandes, S. (2018). Gradient boosting machine for modeling the energy consumption of commercial buildings. *Energy and Buildings*, 158, 1533–1543. doi:10.1016/j.enbuild.2017.11.039
- [60] Candido, C., A.C. Blanco, J. Medina, E. Gubatanga, A. Santos, R. Sta Ana, et R.B. Reyes. « Improving the Consistency of Multi-Temporal Land Cover Mapping of Laguna Lake Watershed Using Light Gradient Boosting Machine (LightGBM) Approach, Change Detection Analysis, and Markov Chain ». *Remote Sensing Applications: Society and Environment* 23 (août 2021): 100565. <https://doi.org/10.1016/j.rsase.2021.100565>.
- [61] Shehadeh, A., Alshboul, O., Al Mamlook, R. E., & Hamedat, O. (2021). Machine learning models for predicting the residual value of heavy construction equipment: An evaluation of modified decision tree, LightGBM, and XGBoost regression. *Automation in Construction*, 129, 103827. doi:10.1016/j.autcon.2021.103827
- [62] .[ Irawati, Mesayu Elida, et Hasballah Zakaria. « Classification Model for Covid-19 Detection Through Recording of Cough Using XGboost Classifier Algorithm ». In *2021 International Symposium on Electronics and Smart Devices (ISESD)*, 1-5, 2021. <https://doi.org/10.1109/ISESD53023.2021.9501695>
- [63] Formation Data Science | DataScientest.com. « Algorithmes de Boosting – AdaBoost, Gradient Boosting, XGBoost », 19 octobre 2020.
- [64] Z. Alom *et al.*, “A State-of-the-Art Survey on Deep Learning Theory and Architectures,”*Electronics*, vol. 8, no. 292, pp. 1–67, 2019, doi: 10.3390/electronics8030292.
- [65] Wang, SC. (2003). Artificial Neural Network. In: *Interdisciplinary Computing in Java Programming*. The Springer International Series in Engineering and Computer Science, vol 743. Springer, Boston, MA. [https://doi.org/10.1007/978-1-4615-0377-4\\_5](https://doi.org/10.1007/978-1-4615-0377-4_5)
- [66] S. The, S. Ai, I. Dalle, and S. Galleria, “Deep Learning in Neural Networks : An Overview,” Lugano,Switzerland, 2014
- [67] François Chollet, *Deep learning with Python* (2017), Manning, chapter 1 p.6
- [68] Danilo P. Mandic , A Generalized Normalized Gradient Descent Algorithm , *IEEE SIGNAL PROCESSING LETTERS*, VOL. 11, NO. 2, FEBRUARY 2004
- [69] Diederik Kingma , Jimmy Lei Ba entitled "Adam: a method for stochastic optimisation".2014
- [70] Bock, S., Goppold, J., & Weiß, M. (2018). An improvement of the convergence proof of the ADAM-Optimizer. arXiv preprint
- [71] Kirori, Z. (2019). Performance Analysis of Stochastic Gradient Descent-Based Algorithms for Time Series Sequence Modeling.

- [72] Yi, D., Ahn, J., & Ji, S. (2020). An Effective Optimization Method for Machine Learning Based on ADAM. *Applied Sciences*, 10(3), 1073.
- [73] Sherstinsky, Alex. « Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network ». *Physica D: Nonlinear Phenomena* 404 (2020): 132306.
- [74] Peng Lu, “Empirical Study and Multi-task Learning Exploration for Neural Sequence Labeling Models”, April, 2019
- [75] Junyoung Chung , “On Deep Multiscale Recurrent Neural Networks”, April, 2018
- [76] David Krueger, “Designing Regularizers and Architectures for Recurrent Neural Networks”, Janvier, 2016
- [77] Ran, Xiangdong, Zhiguang Shan, Yufei Fang, et Chuang Lin. « An LSTM-Based Method with Attention Mechanism for Travel Time Prediction ». *Sensors* 19, n° 4 (2019).
- [78] Brownetal, Recurrent Neural Network Attention Mechanisms for Interpretable System Log Anomaly Detection, MLCS'18, June 12, 2018, Tempe, AZ, USA
- [79] Raffel, Colin, et Daniel P. W. Ellis. « Feed-Forward Networks with Attention Can Solve Some Long-Term Memory Problems », 29 décembre 2015.
- [80] Y.-L. He, L. Chen, Y. Gao et al., Novel double-layer bidirectional LSTM network with improved attention mechanism for predicting energy consumption. *ISA Transactions* (2021), <https://doi.org/10.1016/j.isatra.2021.08.030>.
- [81] <https://www.kaggle.com>
- [82] J. Moolayil, *Learn Keras for Deep Neural Networks*. 2019
- [83] [« TensorFlow »]. <https://www.tensorflow.org/?hl=fr>.
- [84] scikit-learn. « Getting Started ». [https://scikit-learn/stable/getting\\_started.html](https://scikit-learn/stable/getting_started.html).
- [85] C. -W. Huang, C. -T. Chiang and Q. Li, "A study of deep learning networks on mobile traffic forecasting," *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2017, pp. 1-6, doi: 10.1109/PIMRC.2017.8292737.
- [86] Sherstinsky, Alex. « Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network ». *Physica D: Nonlinear Phenomena* 404 (2020): 132306. <https://doi.org/10.1016/j.physd.2019.132306>.
- [87] <https://www.kaggle.com/code/mark4h/vsb-1st-place-solution>
- [88] Y. Hua, "An Efficient Traffic Classification Scheme Using Embedded Feature Selection and LightGBM," *2020 Information Communication Technologies Conference (ICTC), 2020*, pp. 125-130, doi: 10.1109/ICTC49638.2020.9123302
- [89] <https://datacentricai.org/>
- [90] Parascandolo, Giambattista, Heikki Huttunen, et Tuomas Virtanen. « Recurrent neural networks for polyphonic sound event detection in real life recordings ». In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6440-44. Shanghai: IEEE, 2016. <https://doi.org/10.1109/ICASSP.2016.7472917>.

[91] Hu, Zhongxu, Youmin Hu, Jie Liu, Bo Wu, Dongmin Han, et Thomas Kurfess. « A CRNN Module for Hand Pose Estimation ». *Neurocomputing* 333 (mars 2019): 157-68. <https://doi.org/10.1016/j.neucom.2018.12.065>.

[92] Li, Gaoyang, Xiaohua Wang, Xi Li, Aijun Yang, et Mingzhe Rong. « Partial Discharge Recognition with a Multi-Resolution Convolutional Neural Network ». *Sensors* 18, n° 10 (18 octobre 2018): 3512. <https://doi.org/10.3390/s18103512>.

[93] Anowar, Farzana, Samira Sadaoui, et Bassant Selim. « Conceptual and Empirical Comparison of Dimensionality Reduction Algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE) ». *Computer Science Review* 40 (mai 2021): 100378. <https://doi.org/10.1016/j.cosrev.2021.100378>.

[94] Khokhar, Suhail, Abdullah Asuhaimi Mohd Zin, Aslam Pervez Memon, et Ahmad Safawi Mokhtar. « A New Optimal Feature Selection Algorithm for Classification of Power Quality Disturbances Using Discrete Wavelet Transform and Probabilistic Neural Network ». *Measurement* 95 (janvier 2017): 246-59. <https://doi.org/10.1016/j.measurement.2016.10.013>.