HAL
open science

Arc fault detection with machine learning

Hien Duc Vu

To cite this version:

Hien Duc Vu. Arc fault detection with machine learning. Engineering Sciences [physics]. Université de Lorraine, 2019. English. NNT : 2019LORR0152 . tel-02516761

HAL Id: tel-02516761

https://hal.univ-lorraine.fr/tel-02516761

Submitted on 24 Mar 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE LORRAINE

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

École doctorale IAEM Lorraine

Adaptation des méthodes d'apprentissage automatique pour la détection de défauts d'arc électriques

Thèse présentée et soutenue le 28 Octobre 2019

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention systèmes électronique)

par

Vu Hien Duc

Composition du jury

Directeur de thèse:	**Serge WEBER,** Université de Lorraine
CoDirecteur de thèse:	**Patrick SCHWEITZER,** Université de Lorraine
Examinateur:	**Pascal MAUSSION,** INP de Toulouse
Rapporteurs:	**Dominique HOUZET,** Institut polytechnique de Grenoble
	Virginie FRESSE, Université Jean-Monnet-Saint-Étienne
Invité:	**Nicolas BRITSCH,** Hager

Institut Jean Lamour - Équipe MAE 406

The science protects...

Résumé

La détection des arcs électriques se produisant dans un réseau électrique par des approches d'apprentissage représente le cœur des travaux exposés dans cette thèse. Le problème a d'abord été vu comme la classification de séries temporelles à taille fixe en deux classes: normal et défaut. Cette première partie s'appuie sur les travaux de la littérature où les algorithmes de détection sont organisés principalement sur une étape de transformation des signaux acquis sur le réseau, suivie d'une étape d'extraction de caractéristiques descriptives et enfin d'une étape de décision. L'approche multicritères adoptée ici a pour objectif de répondre aux imprécisions systématiquement constatées. Une méthodologie de sélection des meilleures combinaisons et de transformation et de descripteurs a été proposée en exploitant des solutions d'apprentissage. La mise au point de descripteurs pertinents étant toujours difficile, l'évaluation des possibilités offertes par l'apprentissage profond a également été étudiée. Dans une seconde phase l'étude a porté sur les aspects variables dans le temps de la détection de défaut. Deux voies statistiques de décision ont été explorées l'une basée sur le test de probabilités séquentiel (SPRT) l'autre basée sur les réseaux de neurones artificiels LSTM (Long Short Time Memory Network) chacune de ces deux méthodes exploite à sa manière la durée d'une première étape de classification comprise entre 0 et 1 (normal, défaut). La décision par SPRT utilise une intégration de la classification initiale. LSTM apprend à classer des données à temps variable. Les résultats du réseau LSTM sont très prometteurs, il reste néanmoins quelques points à approfondir. L'ensemble de ces travaux s'appuie sur des expérimentations avec des données les plus complètes et les plus large possible sur le domaine des réseaux alternatifs 230V dans un contexte domestique et industriel. La précision obtenue approche les 100% dans la majorité des situations.

Abstract

The detection of electric arcs occurring in an electrical network by machine learning approaches represents the heart of the work presented in this thesis. The problem was first considered as a classification of fixed-size time series with two classes: normal and default. This first part is based on the work of the literature where the detection algorithms are organized mainly on a step of the transformation of the signals acquired on the network, followed by a step of extraction of descriptive characteristics and finally a step of decision. The multi-criteria approach adopted here aims to respond to systematic classification errors. A methodology for selecting the best combinations, transformation, and descriptors has been proposed by using learning solutions. As the development of relevant descriptors is always difficult, differents solutions offered by deep learning has also been studied. In a second phase, the study focused on the variable aspects in time of the fault detection. Two statistical decision paths have been explored, one based on the sequential probabilistic test (SPRT) and the other based on artificial neural networks LSTM (Long Short Time Memory Network). Each of these two methods exploits in its way the duration a first classification step between 0 and 1 (normal, default). The decision by SPRT uses an integration of the initial classification. LSTM learns to classify data with variable time. The results of the LSTM network are very promising, but there are a few things to explore. All of this work is based on experiments with the most complete and broadest possible data on the field of 230V alternative networks in a domestic and industrial context. The accuracy obtained is close to 100% in the majority of situations.

Declaration

The work in this thesis is based on research carried out at the Hagergroup and the laboratory Jean Lamour, France. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my work unless referenced to the contrary in the text.

Copyright © 2019 by VU HIEN DUC.

"The copyright of this thesis rests with the author. No quotations from it should be published without the author's prior written consent and information derived from it should be acknowledged".

Acknowledgements

At first, I would like to thank my thesis supervisors, Serge WEBER and Patrick SCHWEITZER. I have greatly benefited from their skill and intuition at solving problems, I have also learned a lot about writing and communication from them. I also would like to acknowledge Nicolas BRITSCH, who provided much-needed support and guidance along my Ph.D. journey. Without them, I would not have been able to complete this thesis.

Secondly, I would like to extend my warmest and most sincere thanks to Christopher BOURNARIE, Dennis DECKER and all members of the arc fault detection team for the great working condition. Many thanks to Edwin CALDERON for his collaboration and numerous interesting discussions he provided.

Last but not least, I would like to acknowledge all the members of the IJL MAE 406 team and my co-workers at Hager for their welcome and their friendliness. Particularly, Patrice ROTH, Susan TRULSON and Christine DAUDENS for their advice administrative. And of course, Slavisa JOVANOVIC, Patrice JOYEUX, Vincent KENNEL and Gauthier DEPLAUDE for their listening and their advice.

Contents

Résumé **iii**

Abstract **iv**

Declaration **v**

Acknowledgements **vi**

Introduction **1**

0.1 Context and motivation 1

0.2 Contributions 2

0.3 Outline of the thesis 2

1 Problematic and approaches **4**

1.1 Problematic 4

1.1.1 Arc fault 4

1.1.1.1 Parallel arc fault 4

1.1.1.2 Series arc fault 5

1.1.1.3 Causes the of arc fault 7

1.1.2 Standard requirement for arc fault detection product 7

1.1.2.1 Loads and configuration 8

1.1.2.2 Disturbances 9

1.1.2.3 Tripping time 10

1.1.3 Arc fault generation 10

1.1.3.1 Carbonized cable 10

1.1.3.2 Open contacts 12

1.1.3.3 Parallel arc generation 13
1.1.4 Literature detection method 14
1.1.4.1 Physic model based 14
1.1.4.2 Features based detection 15
1.2 Literature approaches 19
1.2.1 Fixed length input 20
1.2.1.1 Multi criteria arc fault detection 20
1.2.1.2 Deep-learning 21
1.2.2 Variable time arc fault detection 22
1.2.2.1 Predefined time windows statistic methods 23
1.2.2.2 Long short term memory arc fault detection 24
1.3 Conclusion 25

2 Multi Criteria Series Arc Fault Detection 26
2.1 General description 26
2.2 Descriptor selection 28
2.2.1 Feature selection overview 29
2.2.2 Arc fault descriptor selection 31
2.3 Multi criteria arc fault detection 36
2.3.1 Binary classification 36
2.3.2 Feature combination 37
2.4 Experimentation result 43
2.4.1 Arc fault detection database 43
2.4.2 AFFs pool construction 44
2.4.2.1 Transformations 45
2.4.2.2 Descriptor 46
2.4.2.3 Neural network wrapper descriptor selection 49
2.4.2.4 Features combination 56
2.5 Conclusion 59

3 Arc fault detection with deep neural-networks 60
3.1 Auto-encoders 60

3.1.1 Stacked auto-encoders for series arc fault detection 62
3.2 Convolutional neural-network 67
3.2.1 Arc fault detection with transfer learning 69
3.2.2 Arc fault detection from scratch 71
3.3 Conclusion 81

4 Time variable arc fault detection 82
4.1 Truncated Sequential probability ratio test decision 82
4.1.1 Fixed time detection method performance 82
4.1.2 Sequential probability ratio test 84
4.1.3 Decision with Truncated SPRT 86
4.1.4 Experimental results 90
4.2 Long short time memory arc fault detection 93
4.2.1 Recurrent neural network 94
4.2.2 Long short term memory neural network 97
4.2.3 Experimentation results 99
4.3 Conclusion 102

5 Conclusions 104
5.1 Prototype Implementation 105
5.2 Perspective 106

Bibliography 107

Résumé substantiel 119

List of Figures

1.1 Example of arc fault 4
1.2 Parallel arc fault 5
1.3 Series arc fault 5
1.4 AC arc fault evolution 6
1.5 Common causes of arc fault 7
1.6 Required configurations for AFDD. R: Resistive load; M: Masking load; Masking loads are mentioned in the previous paragraph. The AC source (230V-50 Hz) generated in accordance with the standard for distribution networks in Europe. 1: With arc fault; 2: Normal. . . 9
1.7 Example of cable specimen and its evolution. a: Specimen cable with slit; b: Start carbonized; c: Completed carbonized and ready for generate arc fault; d: Burned cable. 11
1.8 Example of carbonized cables 11
1.9 Carbonized workbench, a: Overview; b: Different applied voltages. . . 12
1.10 Example of arc fault signal generated with carbonized cable. 12
1.11 Open contact arc generator. a: Diagram; b: Electrodes; c: Used platform. 13
1.12 Test circuit for parallel arc. a: Diagram; b: Test configuration; c: Used platform. 14
1.13 Example of commercial AFDD. 15
1.14 Arc fault detection algorithm. 16
1.15 Arc fault feature extraction. 16
1.16 Example of current waveform and arc fault feature. 17

1.17 Example of current waveform with different loads. a: Vacuum cleaner; b: Arcing vacuum cleaner; c: Resistive load; d: Arcing resistive load; e: Heat gun; f: Arcing heat gun; g: Resistive load and air compressor; h: Resistive load and arcing air compressor (configuration C); i: Air compressor and arcing resistive load (configuration D). 18
1.18 Fixed and variable time arc fault detection. 19
1.19 Common counting techniques for arc fault detection. 23

2.1 Multi AFFs detection methodology. 27
2.2 Filter method. 30
2.3 Wrapped method. 31
2.4 Embedded method. 31
2.5 Wrapper method for descriptor selection. 33
2.6 Depth-first and Breadth-first search. 34
2.7 Example of Depth-first search 34
2.8 Confusion matrix 37
2.9 Illustrative of group distribution when CAFF provides a better discrimination than AFF. 40
2.10 Arc fault feature combination based on inter-group Euclidean distance. L_mse is the lowest mse up to the current stage; R_CAFF is the best CAFF that can be achieved up to the current stage; i: The number of complementary AFFs used for combination at the current stage; n: The number of complementary AFFs available. 42
2.11 Example of signal 44
2.12 Example of transform with arc fault and non-arc signal 46
2.13 Illustrative of customize descriptors 2 - 7 47
2.14 Example of ten descriptors, DFT frequency band 1-10 KHz with non-arc and arc fault signal. a-Current without arc fault at steady state; b-Current with arc fault at steady state; c-Current with arc fault and transient (change of functioning mode). 48
2.15 Label and arc fault feature construction. 49
2.16 Neural-network for arc fault detection. 50

2.17 Descriptor selection with low frequency FFT and CZT (1-20 KHz). . 51
2.18 Descriptor selection with middle frequency FFT and CZT (20-80 KHz). 51
2.19 Descriptor selection process. 52
2.20 Descriptor selection for high frequency FFT and CZT (80 - 150 KHz). 53
2.21 Descriptor selection for wavelet transforms-decomposition level 4 and 5. 53
2.22 Descriptor selection for wavelet transforms-decomposition level 2 and 3. 54
2.23 Descriptor selection for derivative. 55

3.1 Auto-encoder architecture. 61
3.2 Stacked auto-encoder for arc fault detection 63
3.3 Stacked auto-encoders training performance 64
3.4 3 layers stacked auto-encoder training performance 66
3.5 Example of a Convolutional Neural-Network 67
3.6 Example of convolution on raw pixels 68
3.7 Example of max pooling 68
3.8 Examples of real data, a: Current waveform in the case of normal functioning, b: Spectrogram of signal a in the frequency band 1-10 kHz, c: Scalogram of signal a, d: Current waveform in arcing situation, e: Spectrogram of signal d in the frequency band 1-10 kHz, f: Scalogram of signal d. 69
3.9 Adapted GoogLeNet for arc fault detection 70
3.10 Training accuracy 71
3.11 First CONV layer of CNN_arc_1 (20 filters of size 7x7) 72
3.12 Example of filters on first CONV layer weights GoogLeNet 73
3.13 Sample and activations; a-Sample, b-Strongest activation after two CONV layers, c-Activation of first CONV layer, d-Activation of second CONV layer. 73
3.14 First CONV layer weights CNN_arc_2 74
3.15 Strongest activation after two CONV layers CNN_arc_2 74
3.16 Training accuracy 76

3.17 Overview multiple classifier systems. 77
3.18 Example of boostrapping. 78
3.19 Example of boosting. 79

4.1 Sequential probability ratio test 85
4.2 Sequential probability ratio test for arc fault detection 88
4.3 Maximum break time with linear interpolation. 89
4.4 CRMS evaluation. 90
4.5 Example of sprt decision 92
4.6 Recurrent neural network, a- Example of network; b-Unfold architecture 95
4.7 Backpropagation 96
4.8 Backpropagation through time 96
4.9 Long short term memory cell architecture 98
4.10 Example of input sequence in normal functioning and with arc . . . 99
4.11 Neural network for arc fault detection with LSTM cells 100
4.12 LSTMs trainning accuracy 102

List of Tables

1.1 Detection time for series arc. 10

1.2 Detection time for parallel arc. 10

1.3 Representative methods and corresponding accuracies. 20

2.1 Arc faults features extraction. 55

2.2 Descriptor selection with redundant descriptor. 56

2.3 Arc faults features and ratio distance n^o1. 57

2.4 Arc faults features and ratio distance n^o2. 58

3.1 Size of code and accuracies 64

3.2 Number of layer and accuracies 65

3.3 Series arc fault with CNNs classification results 71

3.4 Series arc fault detection with CNNs 75

3.5 Adaptive boost CNN. 80

4.1 Calculated detection time for series arc. 89

4.2 Normalization coefficient 90

4.3 Series arc fault detection with LSTM 101

Introduction

The work presented in this thesis is conducted as part of a thesis in collaboration with Hager Group. The company aims to enhance the performance of arc fault detection products.

0.1 Context and motivation

An arc fault is an electric arc that occurs accidentally in series or parallel with appliances. Parallel arc is produced between two different voltages (for example phase to neutral) and series arc occurs in the same phase. Frequently, arc faults are caused by damaged cables and electrical devices or bad electrical contact. If an arc fault is maintained for a long duration, the energy produced by arc may lead to the ignition of a fire accident. To protect again this phenomenon, arc fault detection devices have been required in several electrical installations for about 20 years. The United States and Canada required them to protect most residential outlets. In Europe, the adoption is also starting, for example in Germany (2017) and United Kingdom (2018) for some type of installation.

Many different arc fault detection devices can be found on the market, but none of them can guarantee absolute protection. They are all susceptible to false positive or false negative (fail to detect an arc fault or recognize a normal function as an arcing condition). The parallel arc can be easy to be detected due to their effect on the level of the line current, the series arc fault is much more complicated to handle. Until now there is no solution which can guarantee to detect all the arc faults without ever producing undesirable tripping. That's the reason why series arc fault detection is an active research topic. An accuracy of about 98-99 % can be observed in the

state of art results, however, they are frequently only tested on several specific cases, furthermore the small percentage of error can be problematical. We can imagine the consequence of a protection device failed to trip or the trip every day for no reason. The chaotic nature of the arc fault and the number of possible scenarios make the conventional model become impractical and imprecise. With the rise of processing power for the embedded system, internet of things and the promising results of machine learning in many fields, several pieces of research for arc fault detection methods based on machine learning can be found in the literature. We believe that the machine learning-based algorithm can bring better performance for arc fault detection.

0.2 Contributions

Motivated by the fact that we can achieve state-of-the-art result with machine learning-based algorithms, in this thesis, we focus on solving the series arc fault detection problem with different techniques. Several results have been presented in our publications [1, 5]. The main contributions of the thesis are as follows:

- We proposed a method that allows choosing the optimal arc fault features for the detection task. Also, the method permits to create a detection algorithm based on multiples arc fault features.
- We showed that different deep learning-based techniques may provide decent results for the series arc fault detection problem.
- We presented two methods to optimize the detection performance by taking into account the time variable aspect of arc fault detection.

0.3 Outline of the thesis

The thesis consists of six chapters and organized as follows:

- Chapter 1 describes the arc fault detection problem and the different approaches to resolve it.

- Chapter 2 presents a method to achieve multi-criteria arc fault detection based on supervised feature selection. We show that by combining several arc fault feature we can achieve better performance that can't be reached by using only one feature.

- Chapter 3 is dedicated to series arc fault detection based on deep learning with the convolutional neural network and autoencoder. The results suggest that this kind of architecture can be used for arc fault detection task. However, the processing cost may discourage the implementation of these methods on an embedded platform.

- Chapter 4 proposes two different approaches to deal with the variable time aspect of the arc fault. Namely, sequential probability test and long short term memory network.

- Chapter 5 is the conclusion that presents an example of an algorithm's implementation on an embedded system and the perspective for future research.

Chapter 1

Problematic and approaches

1.1 Problematic

1.1.1 Arc fault

The standard IEC 62606 defines arc as a luminous discharge of electricity between an insulating medium, usually accompanied by the partial volatilization of the electrodes [6]. An arc fault is a dangerous unintentional parallel or series arc between conductors. Examples of arc discharge can be found in Figure 1.1.

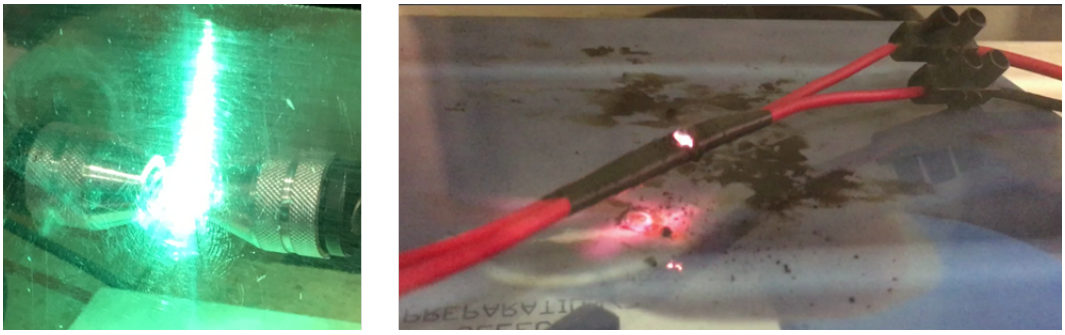

Figure 1.1: Example of arc fault

1.1.1.1 Parallel arc fault

There are two types of parallel (Figure 1.2) arc which may appear on the household electrical network: line to line (L-N) and line to ground (L-G, N-G).

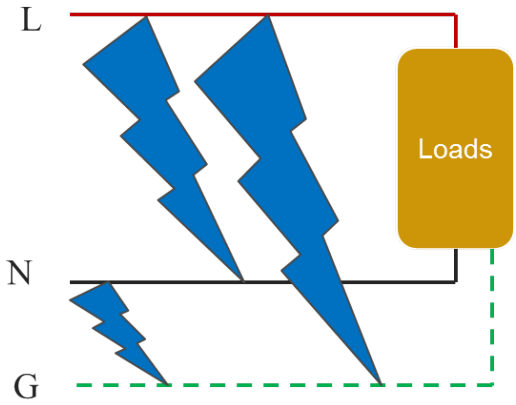

Figure 1.2: Parallel arc fault

In general, the current level of a parallel arc fault is very high and depends on the line characteristics. Thermal magnetic circuit breakers can protect the installation again the first situation (line to line) which is similar to a short circuit. In the case of the line to ground, a differential circuit breaker can also detect the default. However, intermittently arc may have a random current value which is not always be detected by two mentioned devices.

1.1.1.2 Series arc fault

A series arc occurs in one line (Figure 1.3), in the AC system the arc current and arc voltage are varying with time.

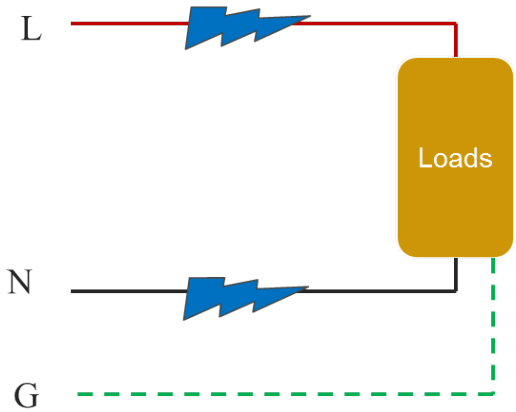

Figure 1.3: Series arc fault

In the case of the series arc, the line current is strongly depended on the appli-

ances which present on the network. This is the reason why series arc fault can only be detected by specific protection devices such as AFDD or AFCI.

For the positive half cycle, when the voltage applied between two electrodes reaches the breakdown threshold the discharge or electric arc starts. The arc voltage decreases and stabilizes to an almost constant value. Later, the supply voltage can go lower than a threshold in which arc can no longer be maintained. After the zero-crossing, the negative half cycle starts. When the voltage reaches the negative breakdown threshold arc start again, the arc voltage will stabilize around a negative constant value. Then arc will disappear when the voltage becomes too low. The phenomenon will reproduce in the same way for the next positive and negative half-cycles as described in Figure 1.4.

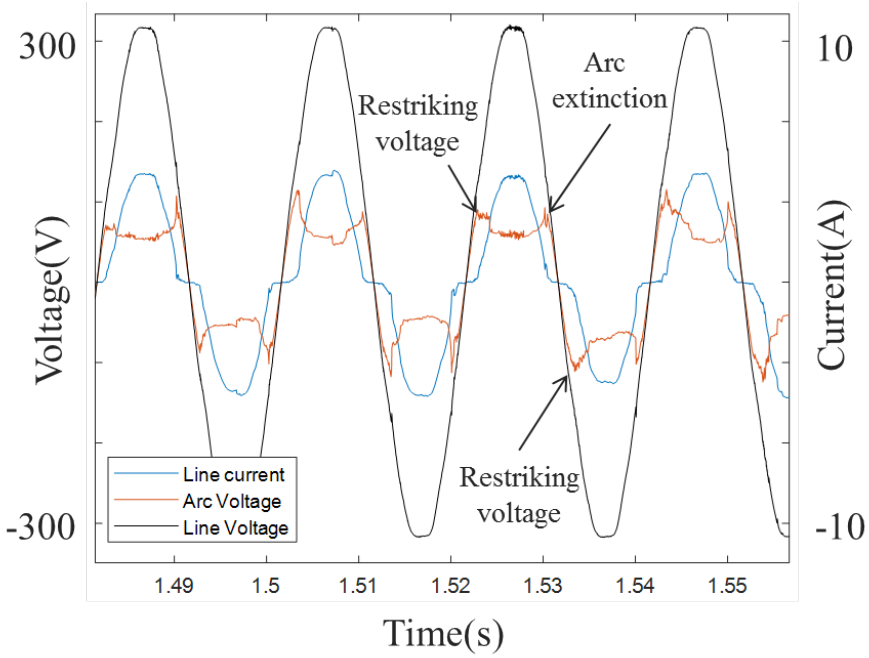

Figure 1.4: AC arc fault evolution

During the arcing, these following features are frequently observed [21]:

- Broadband noise in the current or voltage waveform.
- Flat zero-current around the zero-crossing (due to extinguish and reappear of arc).

- Variation in time of current waveform (arc may drastically change the condition of the electrical network).

1.1.1.3 Causes the of arc fault

There are many possible causes of an arc fault, they are directly related to the state of electrical installation. Among these, some common causes can be listed:

- Damaged cable (excessive use by bending or pulling, destroyed by animal, UV radiation, temperature,...).
- Bad quality connector.
- Lose connection at terminal.

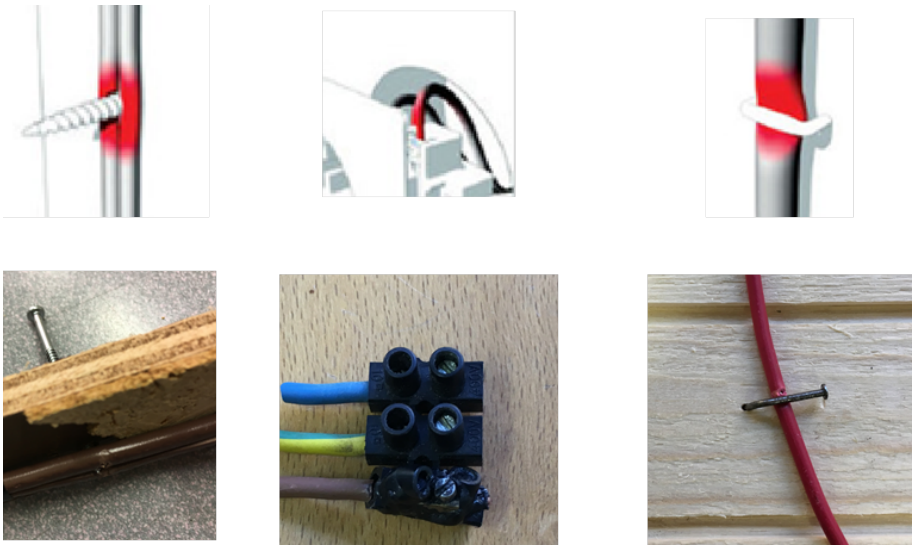

Figure 1.5: Common causes of arc fault

The Figure 1.5 presents some possible situations which lead to arc fault.

1.1.2 Standard requirement for arc fault detection product

AFCI (Arc Fault Circuit Interrupter) and AFDD (Arc Fault Detection Device) have been commercialized in the USA, Canada, and Europe. In the USA the standard UL1699 defines the requirement for an AFCI product and IEC62606 for AFDD in Europe [6, 7]. There are many points common between the two standards (required

appliances, test configurations,etc.). In this study, we will focus on the standard IEC 62606. The standard demands that the protection device should detect arc fault in several conditions and configurations within a fixed time. Besides, a protection device should also answer for a number of unwanted trip tests.

1.1.2.1 Loads and configuration

According to the Europe standard, an AFDD should be able to function with at least these following loads:

- Vacuum cleaner rated at 5A to 7A.
- Electronic switching mode power supply with at least 2.5A load current.
- Air compressor with peak current about 65A.
- Electronic lamp dimmer 600W.
- Two 40W fluorescent lamp and additional 5A resistive load.
- 12V halogen lamps powered with electronic transformer and 5A additional resistive load.
- Electric hand tool such as driller with minimum 600W.
- Resistive load with multiple load currents.

AFDD should detect arc fault and do not make any unwanted trip in normal situations with the mentioned loads in these following configuration (Figure 1.6):

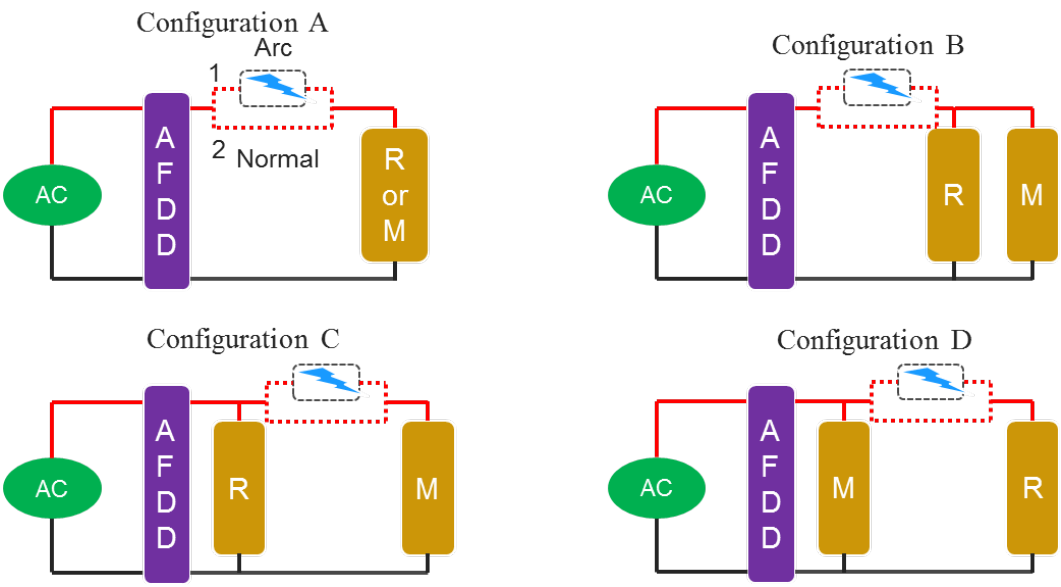

Figure 1.6: Required configurations for AFDD. R: Resistive load; M: Masking load; Masking loads are mentioned in the previous paragraph. The AC source (230V-50 Hz) generated in accordance with the standard for distribution networks in Europe. 1: With arc fault; 2: Normal.

1.1.2.2 Disturbances

AFFDs should function when these following disturbances appear:

- Voltage amplitude variations.
- Voltage unbalance.
- Power frequency variations.
- Magnetic fields.
- Current oscillatory transients.

Some detailed tests for Electromagnetic compatibility (EMC) such as: conducted sine-wave form voltages or currents, fast transients, conducted common mode disturbances in the frequency range lower than 150 kHz, etc. are also mentioned in the standard.

1.1.2.3 Tripping time

For AFDD in Europe, the detection time is related to the current RMS (root mean squared). AFDD should trip within limited time which are described in Table 1.1 and 1.2.

Current (RMS)	2.5A	5A	10A	16A	32A
Response time	1s	0.5s	0.25s	0.15s	0.12s

Table 1.1: Detection time for series arc.

Test arc current (RMS)	75A	100A	150A	200A	300A	500A
N	12	10	8	8	8	8
N is the number of half cycles at the rated frequency (50 Hz) within 0.5s						

Table 1.2: Detection time for parallel arc.

These indicated time values were estimated according to the necessary energy to start a fire incident.

1.1.3 Arc fault generation

There are two possibles ways to generate the series arcing fault which are mentioned in the standard.

1.1.3.1 Carbonized cable

The first way consists of generating an arc fault with a cable specimen. The cable specimen is a cable with two parallel conductors. It should be conditioned to create a carbonized path between the two cable conductors. To achieve that, we can connect the cable to a circuit that provides an open circuit voltage of at least 7 kV and 30mA short circuit current. The needed time to carbonize the cable is about 10 seconds. If the cable is properly prepared, it must be able to handle a 100W/230V load. The evolution of cable specimens can be found in Figure 1.7.

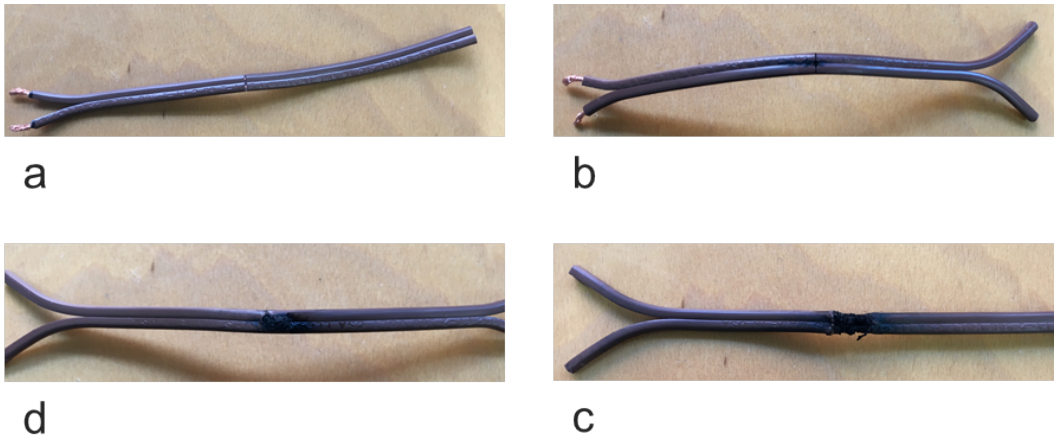

Figure 1.7: Example of cable specimen and its evolution. a: Specimen cable with slit; b: Start carbonized; c: Completed carbonized and ready for generate arc fault; d: Burned cable.

To increase the variety of data, two types of household cables (solid-core wire and stranded core) have been used to simulate the arcing condition. We have also mixed different cables (Figure 1.8) to simulate the fault.

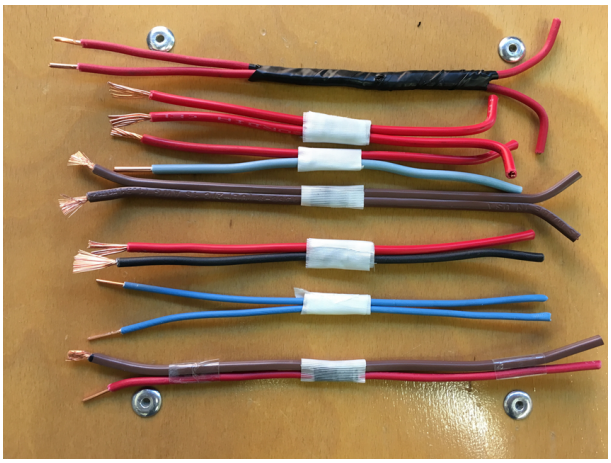

Figure 1.8: Example of carbonized cables

Figure 1.9 shows the workbench to create the carbonized cable.

Figure 1.9: Carbonized workbench, a: Overview; b: Different applied voltages.

Figure 1.10 presents the current, voltage and arc voltage of arc fault signal generated with a cable specimen.

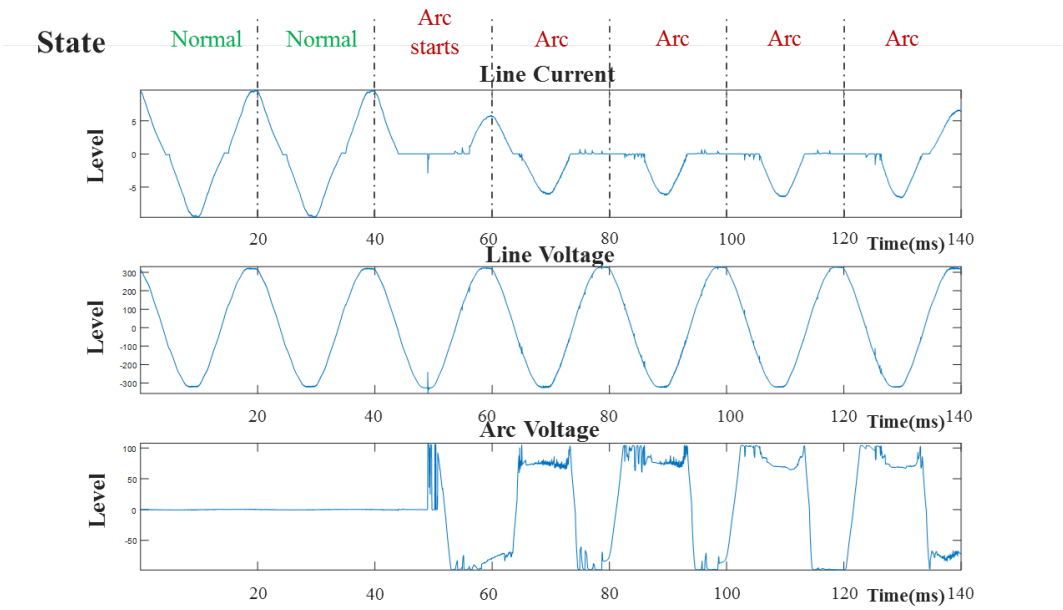

Figure 1.10: Example of arc fault signal generated with carbonized cable.

1.1.3.2 Open contacts

The second way to generate arc fault consists in using a stationary electrode and a moving electrode (Figure 1.11). One electrode should be a carbon-graphite rod and the other should be a copper rod. With an appropriate distance between two electrodes, we can generate a consistence arc fault.

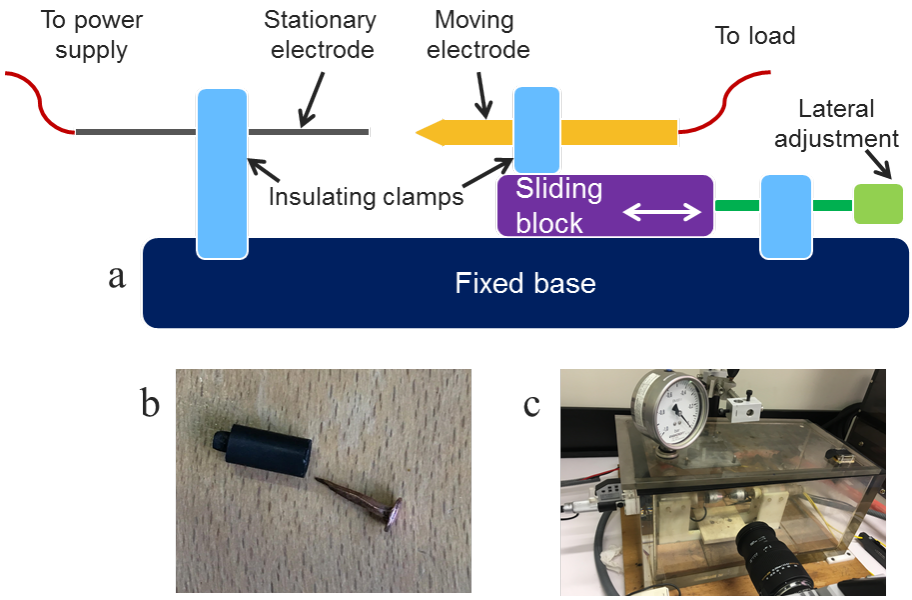

Figure 1.11: Open contact arc generator. a: Diagram; b: Electrodes; c: Used platform.

1.1.3.3 Parallel arc generation

To generate a parallel arc fault, a steel blade can be used. This steel blade can be attached to a lever arm to keep a cutting angle which permits to produce arcing conditions. The blade needs to be well-positioned that solid contact can be made with one conductor and arcing contact with the second conductor. The conductor cable should be used only one time. The arc current can be adjusted with the impedance Z (Figure 1.12).

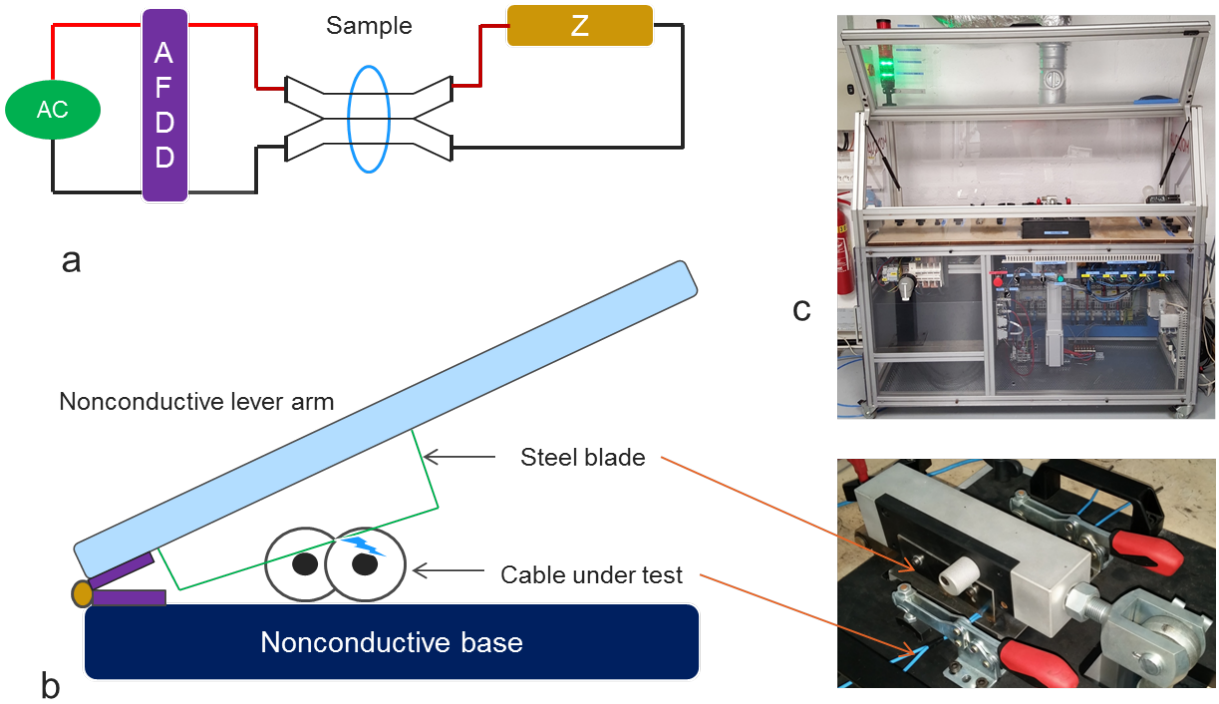

Figure 1.12: Test circuit for parallel arc. a: Diagram; b: Test configuration; c: Used platform.

1.1.4 Literature detection method

1.1.4.1 Physic model based

Because the arcing fault can produce an important amount of heat and ignite a fire, many research papers for arc fault detection have been published. The first possible approach consists of using the conventional arc model based on physics parameters such as current, voltage, distance, etc. We can find several arc model in the lecture: Mayr model [8],Cassie model [9],Habedank [10].

For example Cassie arc model represents the AC arc by the following equation:

$$\frac{1}{g}\frac{dg}{dt} = \frac{1}{\tau}(\frac{u^2}{U_0^2} - 1) \tag{1.1.1}$$

Where g is the conductance of the arc, τ is the arc time constant, u is arc voltage and U_0 is constant of arc voltage.

The other mentioned models are also similar to Cassie's model which require several specifics parameters to establish a detection method. In practice, it's very hard to

determinate all necessary parameters in every installation. Besides, this approach can not take into account the randomness variation of an arc fault condition in time. Due to these reasons, this approach is rarely used in recent arc fault detection research.

1.1.4.2 Features based detection

The second possible approach is the detection based on arc fault feature. This approach shows great success in the literature and implemented the commercial product (Figure 1.13). However, as we will see in the paragraph 1.2.1.1, none of these methods guarantees the perfect discrimination, they are all susceptible to false negative or false positive (indicate no presence of an arcing fault, when in reality it is present or recognize a normal functioning as an arcing condition). This phenomenon can be explained by the evidence that all methods are based on some features of an arc fault, these features can be shared with load and network conditions such as noisy loads, plug-in or unplug appliances, change of functioning mode an appliance on network, etc.

Figure 1.13: Example of commercial AFDD.

In general, an arc fault detection algorithm can be divided into three main parts:

measurement - feature extraction, the classification between normal and abnormal situation and decision (Figure 1.14).

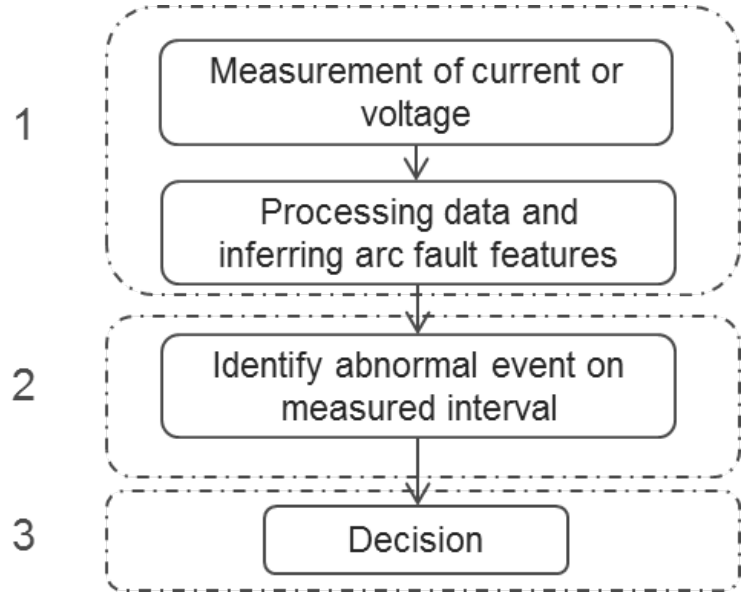

Figure 1.14: Arc fault detection algorithm.

Frequently, the current or voltage of an electrical network will be monitored and then the feature extraction step helps to find arc fault feature (AFF) from the acquired signal. To achieve a good performance at detection, appropriate AFFs are mandatory.

Few detection algorithms have a direct feature extraction part such as fractal, current integral, current variation [11, 13] thus AFF can be inferred without any additional step. In the other detection method, the feature extraction part can be divided into two sub-parts: transformation and descriptors (Figure 1.15).

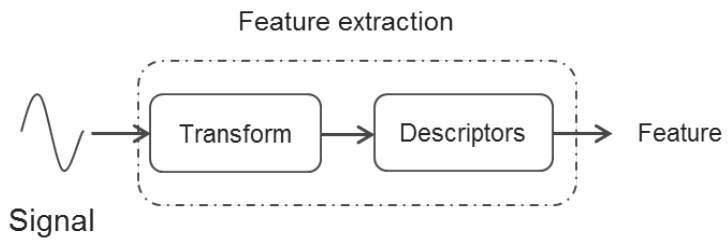

Figure 1.15: Arc fault feature extraction.

Several transformations have been used on arc fault detection field such as:

Fourier transform [14, 15] [47], Wavelet transform [14] [16, 17], filtering [15] [46] [49] [50], correlation [18]. Similarly, a number of descriptors can be listed: variation of sub-spectrum energy between two adjacent power cycles as a descriptor for discrete Fourier transform (DFT) [14] ,harmonic ratio - DFT [19], mean value of the differences [20]between on low-frequency spectra of the current, measured in two subsequent observations with chirp zeta transform [21], eigenvalue – Kalman filtering [22]. The classification part can be simple such as a fixed threshold for AFFs [23], or more complicated such as an artificial neural network (ANN) [14] [24,25], Support vector machine (SVM) [11] [26]. The counting technique or fuzzy logic are used as the decision strategy part [21] [27, 28] [44] [49]. The Figure 1.16 shows an example of the current waveform and arc fault feature.

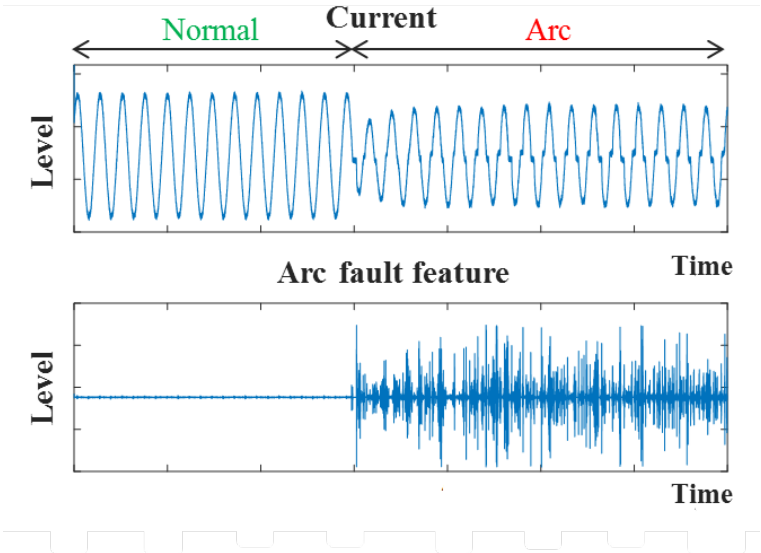

Figure 1.16: Example of current waveform and arc fault feature.

The main difficulty of series arc fault detection is discriminating between arcing and normal situations for all kinds of loads. The current waveform may vary in many different ways depending on the network's conditions and plugging appliances (Figure 1.17).

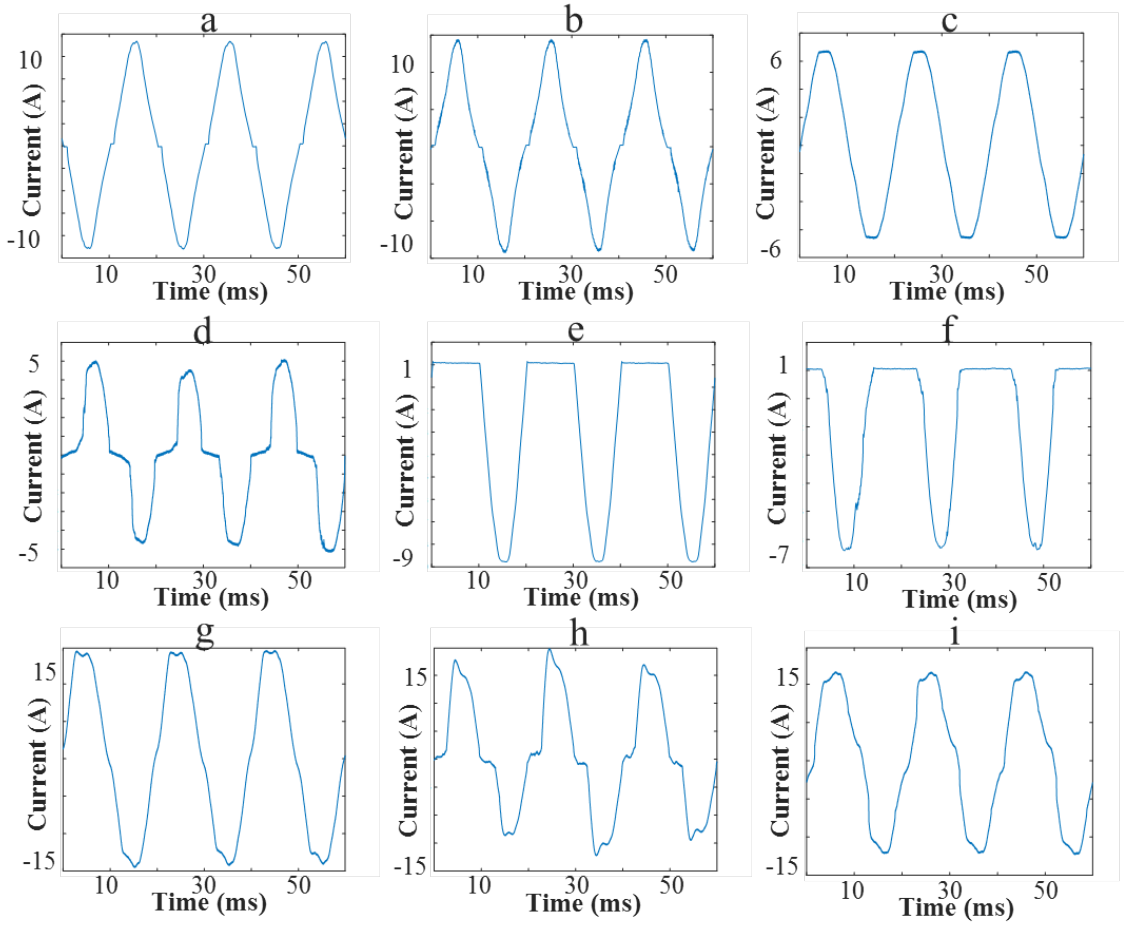

Figure 1.17: Example of current waveform with different loads. a: Vacuum cleaner; b: Arcing vacuum cleaner; c: Resistive load; d: Arcing resistive load; e: Heat gun; f: Arcing heat gun; g: Resistive load and air compressor; h: Resistive load and arcing air compressor (configuration C); i: Air compressor and arcing resistive load (configuration D).

In general, to detect a dangerous arcing condition one or more AFF can be used [13] [20]. The most used AFFs are current zero-crossing, broadband noise, randomness of current variation [18] [29, 30] [48]. However, these features can also be found in a normal functioning network. For example, a vacuum cleaner may produce a lot of arc features at zero-crossings. The randomness of current variation can also be mimicked by changing the functioning mode of an appliance, power on-off an appliance in a short time. Also, broadband noise may be generated by electromagnetic interference or noisy load. Due to this problem, every detection algorithm

proposes a threshold for discriminate between dangerous arcing and normal situation, in other words, a compromise between unwanted trip (false negative) and fail to detect arc fault (false positive). They have different consequences, an unwanted trip may drive to lost working time, disabled service. A fail to detect arc fault is more problematical, and it may lead to a loss in equipment, even life. Therefore these errors should be minimized as much as possible. It can be seen that an AFF may give a better detection performance than another AFF in some situations and vice versa for the rest [31].

1.2 Literature approaches

Arc fault detection problem can be considered as a time series classification with two classes: arcing and normal. The time series corresponds to the acquisition of either current or voltage over a defined duration. Besides, the time series is either fixed or variable. Indeed, at the low current level, the system can afford to take more time to confirm the fault than at a high current level (Figure1.18). At the low current level, up to 1 second of the signal can be used as input and lower to 0.12 second when the current level goes up.

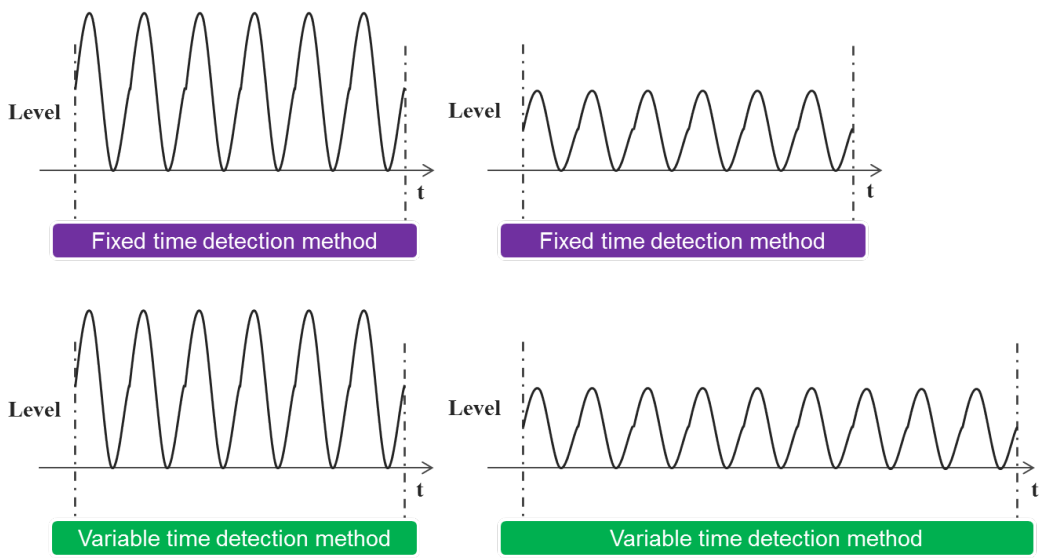

Figure 1.18: Fixed and variable time arc fault detection.

1.2.1 Fixed length input

The first and most popular approach consists of using fixed length series for the detection of an arc fault. The size of the windows shouldn't be higher than 6 periods in the case of European installation (120 ms is the shortest detection time). We can list several detection methods which use fixed-length signal as input: [11] [45] [51]. This approach has the advantage of being easy to craft input features and design classifiers. Taking into account these advantages, at first, we invested time in the development of a detection method that relies on elaborated input features and classifiers.

1.2.1.1 Multi criteria arc fault detection

Table 1.3 presents several detection methods that exploit different characteristics. The last 2 columns present for the most problematic case the accuracy obtained.

Method	Feature type		Lowest prediction	Problematical loads
	Direct feature	Transform		
Sparse representation and neural network [25]	250 sparse coefficients from every current half cycles	-	accuracy 88%	Computer
Chirp Zeta Transform and current difference [21]	The mean value of the difference between two subsequent observation windows	CZT transform	96.7%	Electrical drill
Fractal theory and SVM [11]	Box-Counting dimension and Information dimension	-	98%	Micro-oven and induction cooker
Autoregressive Bispectrum Analysis [26]	-	Two-dimensional Fourier transform of third-order cumulants	97%	Vacuum cleaner
High-frequency energy and current variation [12]	Current integral of one period	Short-time Fourier transform	96%	Electrical drill

Table 1.3: Representative methods and corresponding accuracies.

In the case of simple loads such as resistive or inductive, the prediction accuracy

of some methods can reach 100%. Even with nonlinear loads (computer, electrical drill...) where the detection task becomes more complicated, some algorithms still stay with accuracy very close to 100%. Each method struggles only with one or several types of load. Therefore, the multi-criteria approach may lead to state-of-the-art results if all available methods can be correctly combined. The idea about using more than one AFF to increase detection performance has been mentioned in some publications and patents: verifying the presence of different AFFs [20], using the time characteristics together with the analysis of frequency [13]. However, some very important elements for multi AFFs detection are still missing: the choice of AFFs which should be used together, the combination algorithm to make an efficient detection algorithm from chosen AFFs.

1.2.1.2 Deep-learning

With the rise of deep learning, more approaches to classify time series have been presented. Generally, these approaches tend to replace all or a part of the feature extraction step with the deep learning. They aim to combine the feature learning process while tuning the discriminate classifier; therefore, they reduce the need for feature engineering and domain-specific knowledge. Among these different techniques, auto-encoders, convolutional neural network (CNN), and echo state networks are most commonly used.

For example, Guifang Liu and al. used Fourier transform and stacked auto-encoders as feature extraction to analyze gearbox fault [52]. Or Mohendra Roy et al. proposed a method to detect anomaly in condition monitoring based on stacked auto-encoders as feature extraction and an on-line sequential extreme learning machine network as classifier [53].

Ma Q et al.; Bianchi et al.; Aswolinskiy et al. used echo state networks to classify time series [54, 56]. Z.Wang and W.Yan; C.Liu et al.; G.Devineau; Le Guennec A et al. proposed classification methods based on some variation of CNNs [57, 60]. The results in classification accuracy for time series with deep learning are sometimes better than classical methods [61]. In some related research fields such as computer vision, big data only deep learning-based method can achieve the state of the art

performance.

Besides, we start to find on the literature detection method which does not depend on classical engineered features (features are automatically learned from data) [25]. Motivated by the fact that deep learning may bring performance, we decided to experiment with these techniques for the arc fault detection task.

1.2.2 Variable time arc fault detection

In the household network, there is no need to detect immediately when an arcing situation is just starting. Recognize the arc fault before any severe damage on the system and limited false alarms are more important. Logically, the more time waiting, the more information can be acquired, therefore lead to a better decision. Several methods can be used for classifying variables length time series for arc fault detection. The two most used approaches are: lengthy independent features such as frequency or time-frequency domain and predefined time windows for analysis then combine with statistic methods. For example, Yu-Wei Liu et al. used one power cycle as the time window, the analysis of 30 power cycles and the counting method to detect series arc fault [14]. Or Artale et al. developed a detection algorithm based on frequency analysis that employed with different sizes of observation windows to adapt with the current RMS [21]. Edwin et al. presented a method that detects arc with various response time (20-200ms), the response time depends on the type of load [49].

Besides that, dynamic time warping has been widely used for the classification in other domains [62, 69]. However, in this application, the length of the input series can be greatly varied and the main pattern of input current waveform is repeated between period so it's not practiced to adapt input signals to achieve the same length. Another solution is proposed in the article [61]. It consists of using a recurrent neural network. For example, the RNN is used to detect the fault node in wireless sensor network [70]. Or Arnaz Malhi et al. used RNN to predict the machine health status [71]. In this thesis, we present two variable time arc fault detection methods. The first method is based on predefined time observation and statistic analysis of multiple observations. The second method used long short-term

memory neural network.

1.2.2.1 Predefined time windows statistic methods

For arc fault detection, one or multiple power cycles are widely used as analysis window because the current waveform is repeated after each cycle. It's easier to look into the variation of the current signal when choosing a short analysis window. On the other hand, longer analysis windows may give better frequency resolution. The statistic method may be applied to the results obtained by multiple windows analysis and it helps to make a better decision. For ac fault detection, we can find these following common counting techniques:

- Successive event based (arc fault confirmed if k successive abnormal events appeared) - Figure 1.19a.
- Simple counter (arc fault confirmed if k abnormal events occurred over n analysis windows) - Figure 1.19b.
- Counter with indicator point-based (arc fault confirmed if k abnormal events appeared, it takes into account the order of event- for example, decay faster than it accumulates or opposite) - Figure 1.19c.

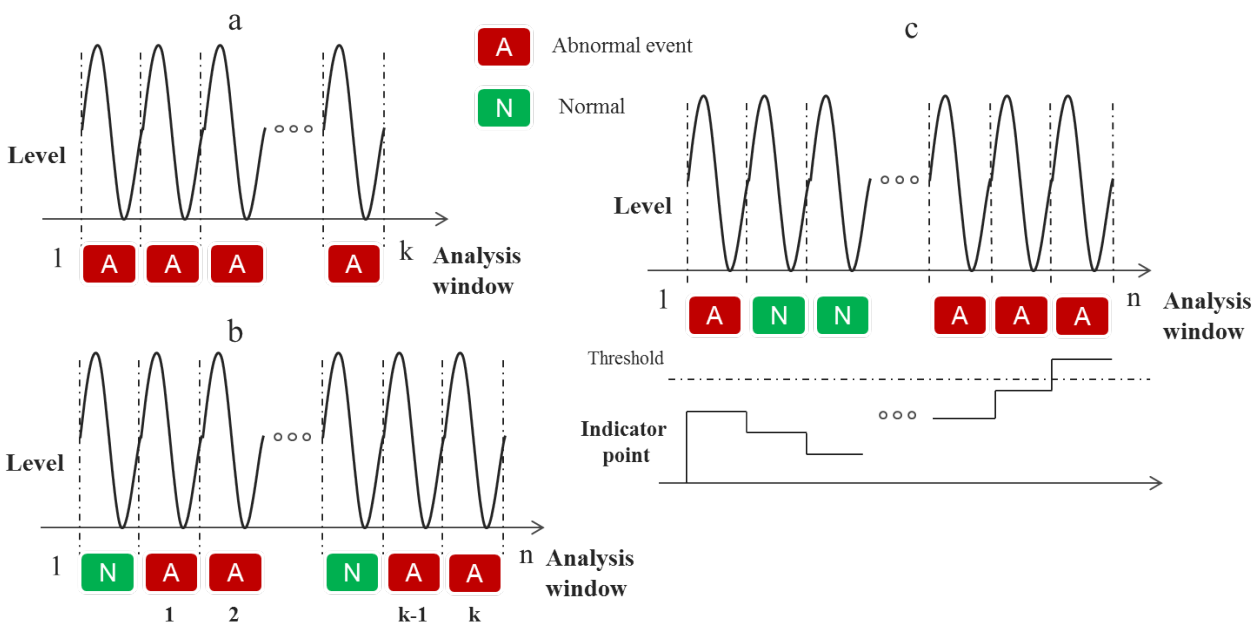

Figure 1.19: Common counting techniques for arc fault detection.

These techniques are widely used because frequently the "isolated" false output can be noticed when using only one window of analysis [21] [49]. However, the choice of counter parameters is always empirically based and aim to fix several considered samples. There is no evidence that the given counting technique has been optimized in terms of information wise nor detection performance. To address this problem we developed a decision method based on a sequential probability ratio test (SPRT) [72] and current RMS. SPRT enables to conclude with a minimum amount of data. The number of analysis windows required can be defined as a random variable- n. The boundary of the decision depends on the expected value of n called average sample (ASN) number. Compared to the corresponding best-fixed sample size test (e.g counting techniques 1& 2) under the same condition of precision the ASN of SPRT is lower. However, ASN of SPRT can become very large in the case of small probabilities error [73] and we have a time constraint to decide in the case of an arc fault. To avoid this problem we applied truncated SPRT with an adapted maximum sample size. This method also helps us to efficiently control the ratio of false alarm and non detected arcing events. More information about our SPRT based method can be found on Chapter 4.

1.2.2.2 Long short term memory arc fault detection

Recurrent neural network (RNN) is suitable for resolve the classification problem of variable length series with multiple segments. Because the temporal relation of the input signal can be well processed with the internal memory of the network. They are widely used in the field where variable input length is dominant such as speech recognition and handwriting recognition [74, 77]. Long short term memory (LSTM) is a type of RNN that has a better capacity of learning long term dependencies than classical RNN. For arc fault detection, LSTM can learn to exploit the variation of current in long duration and the dependencies between multiple windows analysis. The detail of the developed method and its performance is presented on Chapter 4.

1.3 Conclusion

In this chapter, we have presented the problematic of arc fault detection in domestic electrical networks. In addition, we saw the requirements of the standard for arc fault detection products and the way we conducted our experiments. We also introduced the basic elements for the detection task and the state of the art. In general, the most detection efficient methods in the literature are based on current waveform analysis. We investigated further in this direction with machine learning-based approaches.

Chapter 2

Multi Criteria Series Arc Fault Detection

2.1 General description

The orientation proposed below consists of carrying out a multi-criteria detection of arc faults and our research aims to take advantage of the many characteristics of arc faults (AFF) to create the most efficient detection algorithm. The proposed method consists of two main steps: building a pool of arc fault features, selecting the relevant features and then using them together in an efficient way (Figure 2.1).

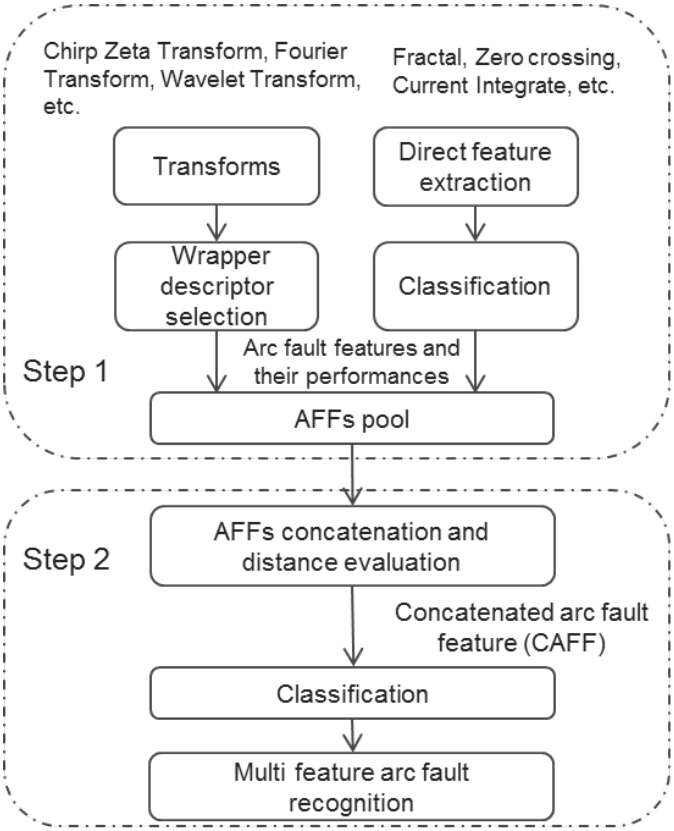

Figure 2.1: Multi AFFs detection methodology.

Step 1.Arc fault feature pool construction:

- Transform and descriptor selection:

  For AFFs which have separate transforms and descriptors, the wrapper selection method is applied in order to guarantee the best set of descriptors for a given transformation. The selection is based on a classification algorithm, which may be any machine learning technique such as an artificial neural network, a support vector machine, etc. Prediction mean squared error is used as the criterion for evaluating the chosen classification method. This is the most commonly used cost function for evaluating classification machine learning-based techniques [32].

- Direct feature extraction:

  Some AFFs can be obtained by the direct feature extraction method [11, 12]. The only information required before adding them into the AFF pool is their respective accuracy in the detection task. Each AFF should be evaluated

using the chosen classification algorithm at the descriptor selection stage to determine the information required.

Step 2.Arc fault feature combination:
The second step consists in ranking all AFFs, removing the irrelevant ones and keeping only those performing well. The combined arc fault feature (CAFF) is then made by combining the best AFF with the other AFFs which have passed the ranking stage. The euclidean distance on feature space has been used as the criterion for selecting which AFF should be combined with the best performing AFF. To be more precise, the ratios of the distance between the false positive (FP) and the false negative (FN) elements and the correct classified elements of the best performing AFF after concatenation were evaluated. The CAFF was considered the best if it had the highest ratio of distance. In the end, the CAFF was used as input for arc fault recognition.

2.2 Descriptor selection

In order to build an efficient detection algorithm from many arc fault features, we need to collect as much as AFF possible. There are many AFFs have been presented in different papers, some of them are a direct feature, therefore, they can be reused without any additional step. On the other hand, for AFFs which are composed of transform and descriptor need to be reconsidered. Because for a given transform, many possible descriptors can be used.
For example, Artale et al. used chirp zeta as transform (CZT), the mean value of the differences between the two low-frequency spectra of the current measured in two subsequent observation windows and differences between the maximum values of spectra in specified frequency intervals as descriptors [21]. Hadziefendic et al presented an algorithm that uses Fourier transform (FT) and the fifth current harmonic as descriptor [19]. These descriptors are interchangeable between different transforms. The fifth current harmonic may also worth to use with CZT and the other descriptors with FT.
Descriptors are often chosen based on the observation of experimented data. In

the literature, frequently the descriptor set is chosen with the help of a relatively small number of appliances, generally less than 10 appliances [11] [18] [21] [23] [31]. Consequently, the performance of the descriptor set can be only warranted for the correspond experimentation. To go further, the arc fault detection method should be able to work on many different installations. So to obtain the desired AFF, the step to determine the optimal set of the descriptor is very important. Note that the optimal descriptor set will be strictly related to the transform.

If only a few descriptors from all available descriptors are used, a lot of information provided by the transform will be wasted. On the other hand, using a large number of descriptors has its shortcomings. It requires sophisticated detection rules which are hard to achieve by the classical heuristic method (observation and multi-thresholds). Detection rule based on machine learning can simplify the task, however, too many descriptors may induce loss of generalization and overfitting problems due to noise and redundancy [33]. Therefore only an optimal number of descriptors should be used.

2.2.1 Feature selection overview

The problem of finding an optimal subset of feature, variable or descriptor is very common. The field of research to resolve this problem is called feature selection. Recently, the rise of computation power and the ease of access to data give the possibility to improve model accuracy. The analyses of high-dimension data become much more popular and the results with the model with high-dimension data are very promising in different fields: medical, speech recognition, object recognition,etc.

The descriptor selection problem can be formulated as:
Consider a subset of descriptor $(d_1, ...d_n)$ and a criterion function C, scores the quality of feature subsets. We are looking for the optimal descriptor subset which satisfies certain conditions for example: find the subset that maximizes the criterion value or finds the smallest subset of features satisfies a given performance.

The selection method can be divided into three categories: supervised, unsupervised and semi-supervised. In this thesis, we will focus on supervised feature selection because labels for data are always available (we can always know which

situation of network, with and without arc on the training phase). The supervised feature selection method can be categorized into the filter method, wrapper method, and embedded method. The filter method (Figure 2.2) separates descriptor selection from classifier. It relies on the measurement of the general characteristics from data such as distance, dependence, mutual information, Pearson correlation [79]. The filter method is very useful for high-dimensional problems due to its low computation cost. However, the selected subset does not guarantee the best performance possible for a classifier.

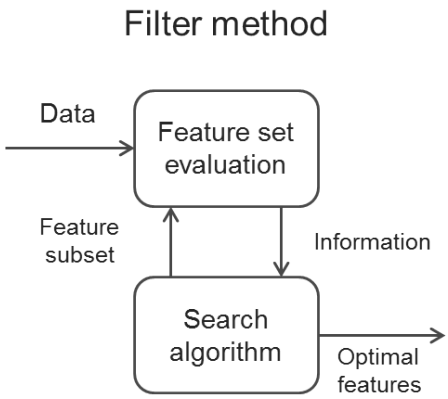

Figure 2.2: Filter method.

The wrapper selection works by generating candidate subsets from all available descriptors then evaluates each subset with a classification algorithm. It uses directly the accuracy of the classifier to determine the desired subset (Figure 2.3). The optimal subset is only guaranteed for the selected classification method. The wrapper method requires a lot more computation compares to the filter method. The wrapper method may be prone to the over-fitting problem but classifiers designed with wrapper selection can achieve better performance than filter methods. Several popular strategies to generate subset can be listed such as: exhaustive search (brute force [35], branch and bound), heuristic (hill-climbing, best-first search) [36] meta-heuristic (genetic algorithm, particle swarm) [37, 38]. Each method has its own inconvenient and advantage. In comparison to the other methods, the exhaustive search method guarantees the best subset of descriptors will be found. The inconvenient of exhaustive search isthe computational cost which may increase ex-

ponentially with the number of descriptors. For example, if n descriptors need to be considered, the traditional exhaustive search method has to examine $\sum_{k=1}^{n} \frac{n!}{k!(n-k)!}$ possible combinations before obtaining the optimal subset.

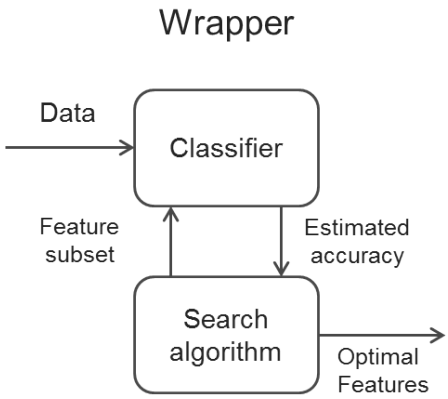

Figure 2.3: Wrapped method.

The embedded method is a hybrid of wrapper and filter method, it uses both statistical analysis and model fitting with classification algorithm [32, 33]. Similar to the wrapper method the selected subset is only optimized for a classification algorithm (Figure 2.4). The embedded method introduces additional constraints to optimize the classifier. Some algorithms can be listed such as LASSO, Concave minimization, l1 -support vector machine [43].

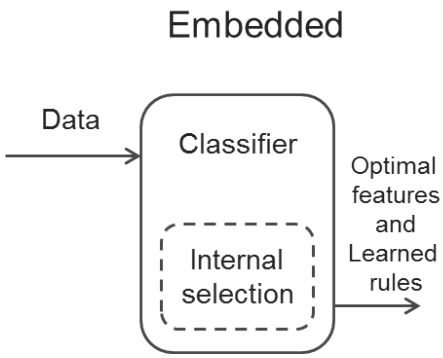

Figure 2.4: Embedded method.

2.2.2 Arc fault descriptor selection

For the case of descriptors selection for a given transform, the wrapper method is the most appropriate because it measures the usefulness of the descriptor. This

property of the wrapper method is very important because a descriptor that isn't efficient by itself can provide a significant performance improvement when taken with other descriptor [33, 34]. The number of descriptors considered for each transform is always less than twenty for arc fault detection field, therefore, the computation cost and the risk of over-fitting can be ignored.

Let S denote a set of n ($n \in \mathbb{N}$) measurements which can be divided into two subset: training data S_{n_1} with n_1 samples and testing data S_{n_2} with n_2 samples. Each sample contains a discrete-time sequence $\{x\}$ of current measured and the associated label L:

$$\begin{cases} L = 1 & \text{in case of arc fault} \\ L = 0 & \text{otherwise} \end{cases}$$

We are looking for an optimized set of descriptors for a given transform TF. The result obtained after a transformation can be noted as a discrete series:

$$\{y\} \text{ with } \{x\} \overset{TF}{\rightarrow} \{y\}$$

For the transform under evaluation k descriptor d($d_i : \{y\} \rightarrow \mathbb{R}$) can be used:

$$d_1, d_2 ... d_k : \{y\} \rightarrow \{z_k\}; (z_k \in \mathbb{R}^k, \{z_k\} := z_1, z_2 ... z_k)$$

For a given subset $\{z_i\} \subseteq \{z_k\}$ a supervised classification algorithm can be deployed with the training data set to discriminate between an arc and no arc samples. Many classification algorithms can be used to do this task such as Logistic regression, support vector machine, artificial neural network, decision tree, naive Bayes classifier. Let C denoted the trained classifier with the subset, we can defined the prediction label of any sample $\{x\}$ on test set as LP ($LP \in \mathbb{R}$), $C : \{z_i\} \rightarrow LP$. We can use the mean square error as the objective function to evaluate the performance of the subset z_i:

$$MSE(\{z_i\}) = \frac{1}{n_2} \sum_{m=1}^{n_2} (L_m - LP_m)^2 \tag{2.2.1a}$$

The descriptor selection problem now can be described as finding the subset $\{z_{opt}\}$ minimize the mean square error :

$$\{z_{opt}\} \subseteq \{z_k\}, MSE(\{z_{opt}\}) \leq MSE(\{z_i\}) \forall \{z_i\} \subseteq z_k$$

The Figure 2.5 describes the descriptor selection step for n given transforms:

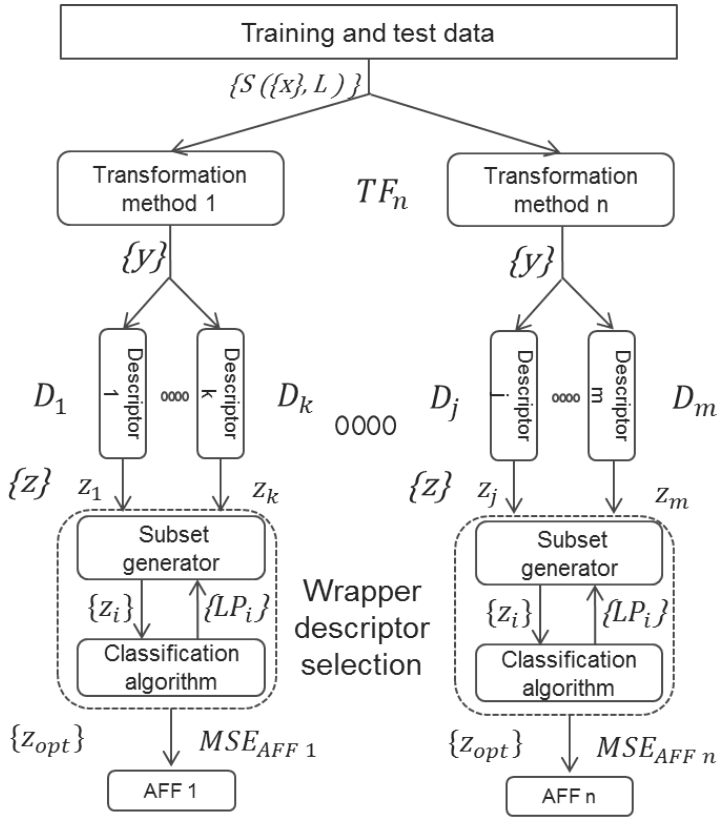

Figure 2.5: Wrapper method for descriptor selection.

One simple solution for obtaining the desired result consists of evaluating every possible subset of $\{z_k\}$ (brute force search). However, this solution has the highest computational cost and many unnecessary subsets would be evaluated. For example, if one of the most relevant descriptors is removed, the classification performance is affected. It is no use evaluating all the subsets without this relevant descriptor. In order to avoid this problem, the search solution, the branch and bound elimination algorithm, can be deployed. First the $MSE(\{z_k\})$ will be calculated, the bound value $\theta = MSE(\{z_k\})$ is set to the child nodes (subset derived from $\{z_k\}$) on the first level of depth. On this level, all descriptors will be removed one by one from z_1 to z_k and the corresponding subset: $\{z_{k1}\}, ... \{z_{kk}\}\, with\, \{z_{ki}\} \setminus \{z_k\} = z_i$ is evaluated. z_i can be defined as significant descriptor if $MSE(\{z_i\}) \geq \theta$ or less-significant descriptor if $MSE(\{z_i\}) < \theta$. After this first level, either depth-first search or breadth-first search algorithms [39] can be used. Figure 2.6 shows the principle of two search algorithms.

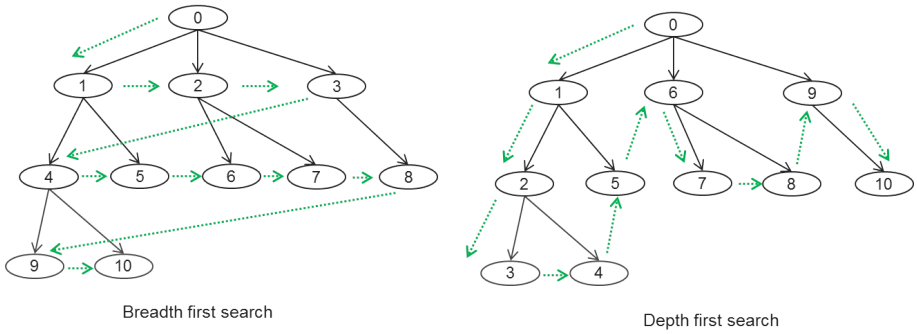

Figure 2.6: Depth-first and Breadth-first search.

The depth-first search algorithm gives priority to investigating on a branch before backtracking, and the breadth-first search to exploring neighbor nodes first before going deeper. If the number of descriptors is too high the breadth-first algorithm may help to limit the calculation time by fixing a depth level. Conversely, if a performance has been defined, the depth-first algorithm may converge faster to the optimal solution with the lowest number of descriptors. In this thesis, the depth-first has been chosen. The child nodes are created by removing one less-significant descriptor from the parent nodes. This newly created child node must be different from any node on the left to avoid any unnecessary computation. The upper bound for the any child node can be defined as follows: $\theta_{child} = MSE(\{z_{parent}\})$ where $\{z_{parent}\}$ is the subset which generates the child node. For a given transform TF the selection algorithm stops when every node has been evaluated. As a result both optimal subsets of descriptor $\{z_{opt}\}$ and $MSE(\{z_{opt}\})$are determined.

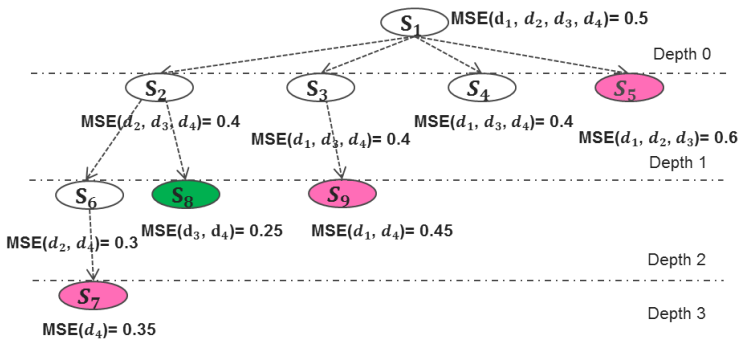

Figure 2.7: Example of Depth-first search

Figure 2.7 shows an example using a depth-first search with branch and bound

elimination for descriptor selection. The MSE should be minimized. The descriptor subset was evaluated from S1 to S9. After the first depth level evaluation, in step 5 (S5), descriptor 4 (d4) is considered as a significant descriptor and therefore all subsets without d4 are not evaluated. Red nodes violate the upper bound (MSE of parent node), therefore no more child nodes are generated from these nodes. Therefore subset (d3, d4) is the optimal subset in this example.

Pseudo code for descriptor subset selection based on backward elimination is as following:

```
Descriptor set {z_k} := z_1 , z_2 ,... z_k
MSE({z_k}) = Compute (MSE of classifier trained from {z_k})
Global variable min_MSE = MSE({z_k}) , best_set = {z_k}
List of evaluated descriptor set : EVA_list
List of MSE value corresponds for each removed descriptor : MSE_list
List of less significant descriptor : LSD_list
for each z_i of {z_k}
   {z_ki} = remove z_i from {z_k}
   MSE({z_ki}) = Compute (MSE of classifier trained from {z_ki})
   MSE_list[i] = MSE({z_ki})
   if MSE({z_ki}) < min_MSE
      LSD_list = LSD_list append z_i
   end if
end for
for each z_i of LSD_list
   {z_n} = remove z_i from {z_k}
   B_ELI ( {z_n} , MSE_list[i ] )
end for
end of program

Function B_ELI is
Input: Descriptor set {z_n} and MSE
for each z_i of LSD_list
```

```
    {z_ni} = remove z_i from {z_n}
    if {z_ni} ∈ EVA_list
        continue
    end if
    EVA_list = EVA_list append {z_ni}
    MSE({z_ni}) = Compute (MSE of classifier trained from {z_ni})
    if MSE({z_ni}) < MSE
        if MSE({z_ni}) < min_MSE
            min_MSE = , MSE({z_ni})
            best_set = {z_ni}
        end if
    B_ELI ( {z_ni} , MSE({z_ni}) )
    end if
end for
return
```

2.3 Multi criteria arc fault detection

The idea behind multi-criteria arc fault detection is: if one AFF is sensitive with certain types of load and other is not, we can combine several AFFs to make a better detection system. The more AFF used, the more chance to find the separation between arcing and normal situation but it's also harder to find this separation.

2.3.1 Binary classification

The problem of arc fault detection is a binary classification problem. As mentioned before we are looking for a classifier $C : \{z_i\} \rightarrow LP$. If we add a threshold $ThS \in \mathbb{R}$ which helps to discriminate between an arc and normal situation.

$$\begin{cases} \text{if } LP \geqslant ThS \ : \ \text{Arc fault} \\ \text{if } LP < ThS \ : \ \text{Normal} \end{cases}$$

For a sample with associated label L there will be four possibilities which can be represented in a confusion matrix (Figure 2.8):

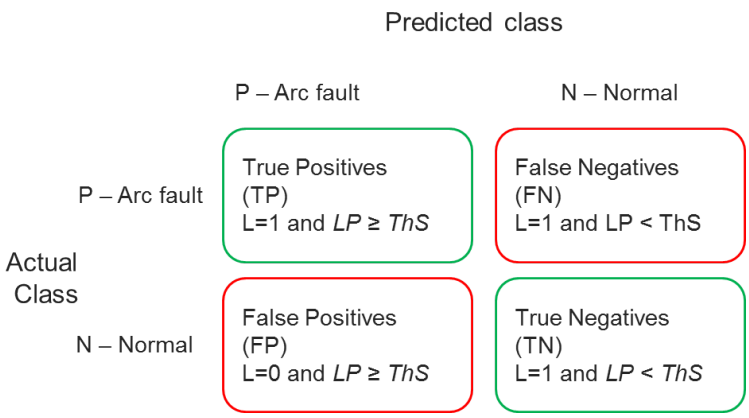

Figure 2.8: Confusion matrix

In the first case, the given sample has been correctly classified. The given sample is conventionally called a true positive element (the term true stays for the element has been correctly classified and positive term shows that the event we want to detect occurs). In the second case, C misclassifies the considered sample as normal, so a false negative. Similarly for the false positives and true negatives cases.

2.3.2 Feature combination

The problem of feature combination that we address in this thesis can be defined as:

Given the training set S_{n_1}: $(\{x_i\}, L_i)_{i=1,....,n_1}$ of n_1 samples and the associated label $L \in \{0,1\}$. With many arc fault features $AFF_1, ..., AFF_m$,$AFF_i : \{x\} \rightarrow \mathbb{R}^k$ where k denotes the dimension of the given arc fault feature . The feature combination problem consists of finding an appropriate classification $C : \{S_{n_1}\} \rightarrow \{0,1\}$ from the training set and arc fault features set.

There are many methods to combine features, the most classical method consists of concatenating different feature vectors into a single vector and then train the classifier with the concatenated vector [78]. Another way to combine many features consists of adding one more dimensional reduction method such as principal com-

ponent analysis or linear discriminant analysis on the concatenated vector before start the training phase. It is also worth to mention the combination method which creates many separated classifiers. Each classifier is trained with only one feature and then combining many classifiers through integration technics such as majority voting, decision tree, etc.

In the arc fault detection field, many AFFs can be used. The first and second combination methods may create a very long vector thus lead to complicated training and high risk of low performance and over-fitting. In addition, the provided features may be redundant or irrelevant which decreases furthermore the efficiency of methods. The last method uses each feature independently at training phase consequently the relation between different features is not taken into account.

A solution to avoid using too long input vector and take advantage of relationships between multiple features in the training phase consist of concatenating a limited number of features and then training the classier with the concatenated vector. This solution comes with the cost of computation because we need to train a number of possible combinations of features before achieving the optimal feature set. In practice, along with classification performance, the available data, the complexity of classification algorithm and training time are the main parameters should be taken into account.

Taking these parameters into account, a suitable combination method has been developed. The combination method consists of 3 steps: the ranking step on every available AFFs, the distance evaluation of different generated concatenated arc fault features (CAFFs) and the classification step which uses the CAFFs with the highest ratio of distance. The ranking step helps to find the most relevant arc fault feature AFF_1 and to remove the irrelevant (poor detection performance)AFFs. Consider AFF_1 with the lowest MSE($MSE(\{AFF_1\}) \leq MSE(\{AFF_i\})\ \forall i \neq 1, AFF_i \subseteq P$).

If $MSE(\{AFF_1\})$ is higher than the desired error (noted d_mse), a solution for obtaining the performance we want might be to combine AFF_1 with the other AFFs. The concatenation method for combining features can be used. For example in the case of two AFFs, from $AFF_1 =: z_1, z_2, ...z_i$ and $AFF_2 =: z_j, z_{j+2}, ...z_{j+k}$ concatenated feature $\{CAFF\} =:: z_1, z_2, ...z_{i,j}\ , z_{j+2}, z_{j+k}$ can be formed.

By using the result from AFF_1 and its trained classifier, the data set S_{n_1}can be divided into four groups: $TP_{AFF_1}, FP_{AFF_1}, TN_{AFF_1}$and FN_{AFF_1}. TP, FP, FN, and TN stand for true positive, false positive, false negative and true negative groups. The calculated label LP is a real number between 0 and 1, thus a threshold *ThS* is needed to define these groups. Since the sensitivity and specificity (proportion of false positives and false negatives) of the model are not of concern for the moment, the threshold *Ths* can be fixed at 0.5 (which is analog to the balanced value).

Since AFF1 is insufficient to completely resolve the detection problem, the four groups $TP_{AFF_1}, FP_{AFF_1}, TN_{AFF_1}$ and FN_{AFF_1} cannot be easily distinguished. The groups $FP_{AFF_1}$and FN_{AFF_1} always overlap with groups $TN_{AFF_1}$and TP_{AFF_1} . If any other AFF is more efficient than AFF_1 for sensitive elements (FP_{AFF_1}or FN_{AFF_1}), this AFF may help to better separate the groups FP_{AFF_1} , FN_{AFF_1} from TP_{AFF_1} and TN_{AFF_1} . As the distance between these groups and their distribution on feature space are related to prediction performance [40], higher detection performance can be expected after this combination. An illustration of group distribution on CAFF feature space is shown in Figure 2.9.

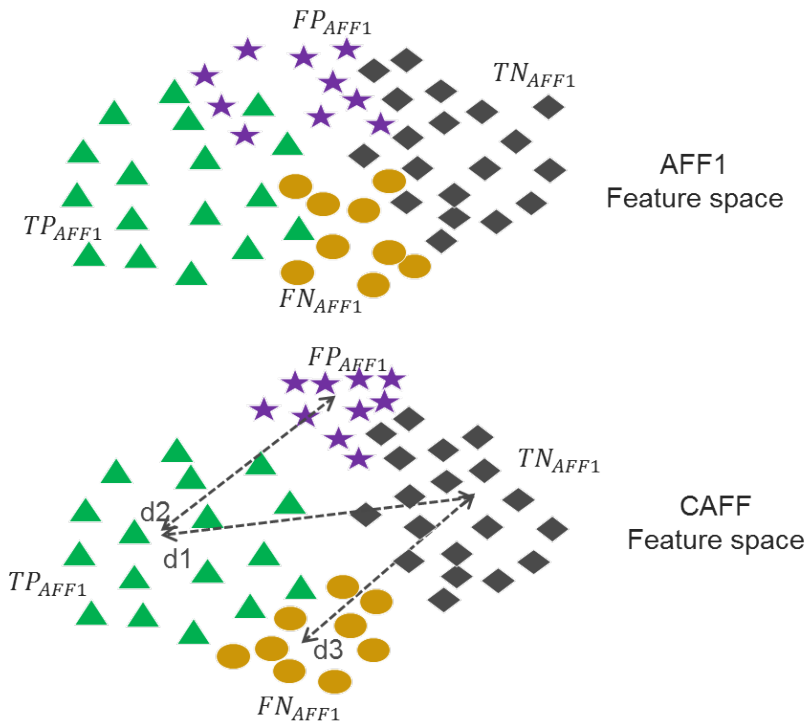

Figure 2.9: Illustrative of group distribution when CAFF provides a better discrimination than AFF.

If it is assumed that the group TP_{AFF_1} has k elements, the centroid of this group after concatenation (CAFF feature space) can be determined with the following relation (Equation 2.3.2):

$$Centroid(TP_{AFF_1}) = \frac{TP^1_{AFF_1} + TP^2_{AFF_1} + ... + TP^k_{AFF_1}}{k} \tag{2.3.2}$$

The centroids of the other groups can be determined with the same formula. Let $\vec{u}, \vec{v} \in \mathbb{R}^n$ be the centroids of group TP_{AFF_1} and TN_{AFF_1}, the inter-group distance d_1 between TP_{AFF_1} and TN_{AFF_1} can be calculated (Equation 2.3.3):

$$d_1 = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + ... (u_n - v_n)^2} \tag{2.3.3}$$

Similarly for the other inter-group distances:

d_2 : distance (TP_{AFF_1}, FP_{AFF_1}); d_3 : distance (TN_{AFF_1}, FN_{AFF_1})

The most efficient CAFF is that which has the best separation between groups (TP_{AFF_1}, FP_{AFF_1}) and (TN_{AFF_1}, FN_{AFF_1}) . In other words, the one with highest

sum of ratio distances:

$$\frac{d_2 + d_3}{d_1}$$

The higher the inter-group distances d_2, d_3 is, the better separation we can expect. However, it is also important to take into consideration the dimensions of different CAFFs feature space and it is for this reason that distance d_1 has been used as a normalized coefficient. The next step consists in discriminating between arc fault and non-arc situations with the help of a classification method and the most efficient CAFF (created from AFF_1 and another AFF).

If the mse of CAFF is still higher than the desired mse, the number of AFFs used for combination should be increased. More specifically, after each stage of evaluation, if the desired performance is still out of reach, the number of complementary AFFs is increased by one. The algorithm stops when the desired error has been achieved or all possible CAFFs have been examined. In the second case, the pool of arc fault features should be revised (add new transforms and use more suitable descriptors). Figure 2.10 describes the combination method in detail.

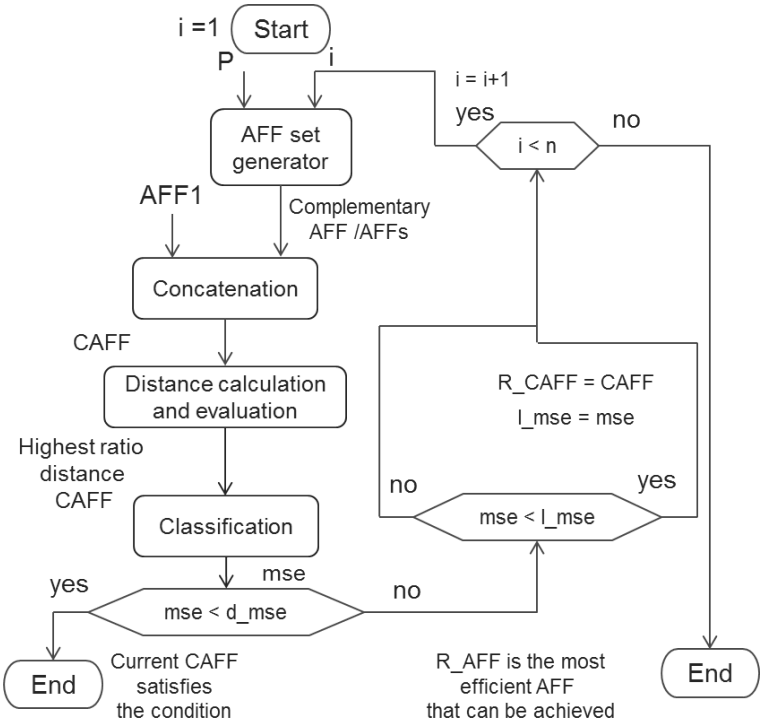

Figure 2.10: Arc fault feature combination based on inter-group Euclidean distance. L_mse is the lowest mse up to the current stage; R_CAFF is the best CAFF that can be achieved up to the current stage; i: The number of complementary AFFs used for combination at the current stage; n: The number of complementary AFFs available.

In the beginning, i is equal to 1 (stage 1). Every possible CAFF(AFF_1 concatenate with another AFF) is evaluated. The highest distance CAFF is used to train a classifier and then the classifier's performance is compared to the desired mse. If the current CAFF satisfies the condition, the combination algorithm reaches its end. Otherwise, CAFF will be noted as R_CAFF and its mse as l_mse. The variable i will be incremented by 1 (stage 2) and every CAFF (concatenated from AFF_1 and two other AFFs) is evaluated. The process is repeated until a satisfied CAFF has been found or i equals n. In the second scenario, R_CAFF is the most efficient CAFF and l_mse is the lowest error that can be achieved. In comparison to the wrapper selection method used in the descriptor selection, the distance evaluation method greatly reduces the computation cost because there is only one classifier which requires training at each stage. The computation of ratio distance is negligi-

ble compared to the cost of the training classifier. If the AFF pool has n elements, in the worst-case scenario the wrapper method needs to train $\sum_{k=1}^{n} \frac{n!}{k!(n-k)!}$ classifiers while the distance evaluation method requires to train only n classifiers.

2.4 Experimentation result

This section describes in detail how the method presented in this work has been applied. The results obtained can be found at the end of the section.

2.4.1 Arc fault detection database

A database is essential for the proposed method and the construction of the database is very important as it directly affects how efficient the obtained detection method can be. The number of appliances, combinations, and disturbances presented in an electrical network are very large and it is impractical to build a universal database that covers all possible situations. There is no need to make a universal detection method for all installations and it is always possible to construct a useful database for any installation when the number of situations is narrow.

In this series of experiments, the European household network was studied. For this installation, the standard IEC62606 –"General requirements for arc fault detection devices" was used as the main reference for database construction. The database contained the sample with arc fault and non-arc fault situations. The arc fault is created by a carbonized cable specimen according to standard IEC 62606. The non-arc samples are also generated with the same configuration of appliances or network. In order to guarantee good performance against false-positive errors, many samples contain the transient state of appliances. Besides, the standard cross-talk test has been also taken into account in the database. The following types of appliances were used for constructing the database: masking loads (air conditioner, air compressor, computer, dimming lamp, electric drill, vacuum cleaner, halogen lamp, fluorescent lamp, and hairdryer) and resistance. At least two or three different brands were used for each type of appliance.

Each sample contains a measurement of network voltage, current and arc voltage

for a fixed duration. It is essential to measure the current for the method presented. Measuring arc voltage helps to accurately label the samples and every period of each sample is labeled. If the arc voltage stays at the noise level for one period, the period is labeled as normal. If the arc voltage waveform corresponds to an arcing situation, the period is labeled as arc influenced. The period which is not completely affected by the arc fault is not labeled (an example is shown in Figure 2.11). There are a total of 16,231 samples - 3,405 samples with arc fault, and 12,826 without arc fault. The signals were acquired with the sampling frequency of 1 MHz because all feature extraction methods in this study operate at a frequency lower than 1 MHz.

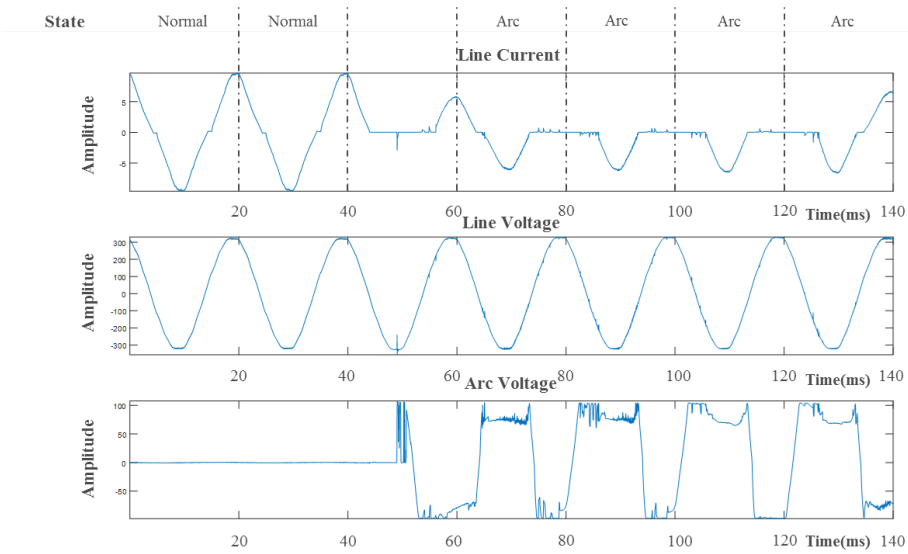

Figure 2.11: Example of signal

2.4.2 AFFs pool construction

After the database was established, the next step consisted in creating an arc fault features pool. This step was accomplished by applying different feature extraction methods to the current signature of the database. In this series of experiments, only those feature extraction methods which are composed of transforms and descriptors were used.

2.4.2.1 Transformations

Five different transforms were chosen in this study because they are used in arc fault detection literature [21] [27] [31] [41]. All transforms are defined below; $\{x_n\}$ represents the current time series input and $\{y_k\}$ is the result obtained with the respected transform, N is the number of elements in a current series:

- Current finite difference:
  $y_k = |x_{k+1} - x_k|\,; 1 \leq k \leq N-1$

- Discrete Fourier transform (FFT):
  $y_k = \left|\sum_{m=1}^{N} x_m.e^{-i\frac{2\pi}{N}km}\right|; 1 \leq k \leq N-1$

- Chirp Zeta transform (CZT):
  $y_k = \left|\sum_{m=1}^{N} x_m.e^{-i\frac{2f\pi}{fsN}km}\right|; 1 \leq k \leq N-1$

  Four different frequency bands were considered for FFT and CZT:

  1 kHz - 10 kHz

  10 kHz - 20 kHz

  50 kHz - 80 kHz

  80 kHz - 100 kHz

  100 kHz - 150 kHz

- Wavelet:
  $y(m,k) = \frac{1}{a^m}\sum_{m=0}^{N-1} x_n.g(\frac{k-b}{a^m})$ g(.) is the mother wavelet ,m is the decomposition level, in this paper a=2 .

  For mother wavelet: Daubechies 4 (DB4) and Meyer (Dmey) were selected with decomposition level 2, 3, 4 and 5 (LVL2,3, 4, 5) for each wavelet respectively.

Figure 2.12 shows an example of transform with arc fault and non-arc signal.

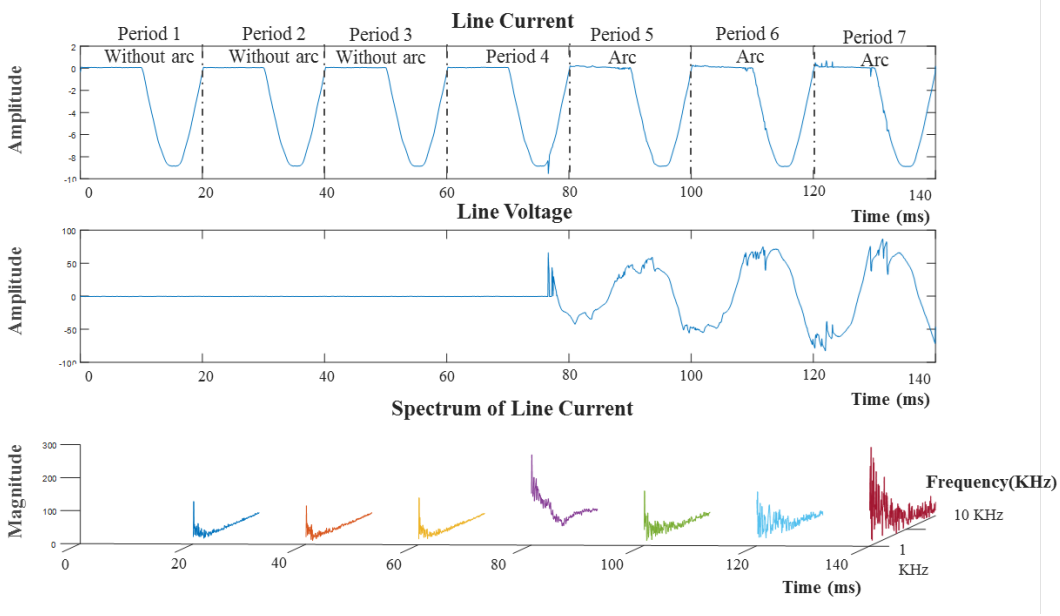

Figure 2.12: Example of transform with arc fault and non-arc signal

2.4.2.2 Descriptor

The results after each transformation are discrete series and they can be noted as $\{y_n\} := y_1, y_2, ...y_n$ In order to finish the feature extraction step, every transform needs to be associated with several descriptors. Based on the literature of arcing detection [32], the groups of descriptors were chosen as follows. The first group of descriptors is based on statistical analysis. In this study, the first, second, third and fourth-order moment was chosen to measure the shape of $\{y_n\}$.

- Mean value: $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_n$
- Variance : $s^2 = \frac{1}{n}\sum_{i=1}^{n}(y_n - \bar{y})^2$
- Skewness: $\frac{1}{n}\sum_{i=1}^{n}(\frac{y_n-\bar{y}}{s})^3$
- Kurtosis:$\frac{1}{n}\sum_{i=1}^{n}(\frac{y_n-\bar{y}}{s})^4$

The second group of descriptors relates to the analysis of the first peak of $\{y_n\}$ (the peak's value, location, and duration). Finding all the points of $\{y_n\}$ around the first peak, which are higher than the average value, allows the duration of the signal to be found. An illustration of these descriptors can be found in Figure 2.13.

- The max value: $y_m \in \{y_n\}, y_m \geq y_i \forall y_i \in \{y_m\}$
- Normalized index of the max: $\frac{m}{n}$
- Normalized highest pulse duration: $\frac{k-j}{n}$, $\begin{cases} j \leq m, y_i \geq \bar{y}\ \forall j \leq i \leq m \\ y_{j-1} < \bar{y} \text{ or } j = 1 \\ k \geq m, y_i \geq \bar{y}\ \forall m \leq i \leq k \\ y_{k+1} < \bar{y} \text{ or } k = n \end{cases}$

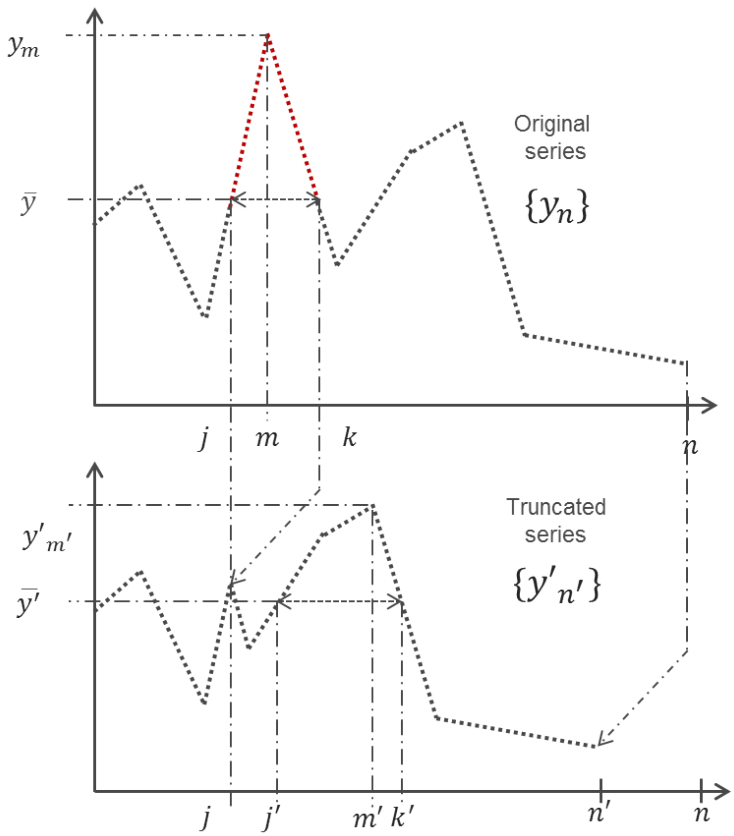

Figure 2.13: Illustrative of customize descriptors 2 - 7

The last group of descriptors gives information about the second peak of data series $\{y_n\}$. The highest pulse is removed (all points between j and k of $\{y_n\}$) in order to find the second peak. The truncated series $\{y'_{n'}\}$ can be defined as: $\{y'_{n'}\}$ $:= y'_{1'}, y'_{2'}, \ldots, y'_{n'} = y_1, y_2, \ldots y_j, y_k, \ldots, y_n$ The first peak of $\{y'_{n'}\}$ is now equivalent to the second peak of the data series.

- The second peak value is : $y'_{m'} \in \{y'_{n'}\}, y'_{m'} \geq y'_{i'} \forall y'_{i'} \in \{y'_{n'}\}$
- Normalized index of the second peak: $\frac{m'}{n'}$
- Normalized second pulse duration: $\frac{k'-j'}{n'}$

Note that: $\bar{y'} = \frac{1}{n'} \sum_{i=1}^{n'} y'_{n'}$; $\begin{cases} j' \leq m', y'_i \geq \bar{y'} \ \forall j' \leq i \leq m' \\ y'_{j'-1} < \bar{y'} \text{ or } j' = 1 \\ k' \geq m', y'_i \geq \bar{y'} \ \forall m' \leq i \leq k' \\ y'_{k'+1} < \bar{y'} \text{ or } k' = n' \end{cases}$

The Figure 2.14 shows an example of 10 descriptors for a given transform and their numbering.

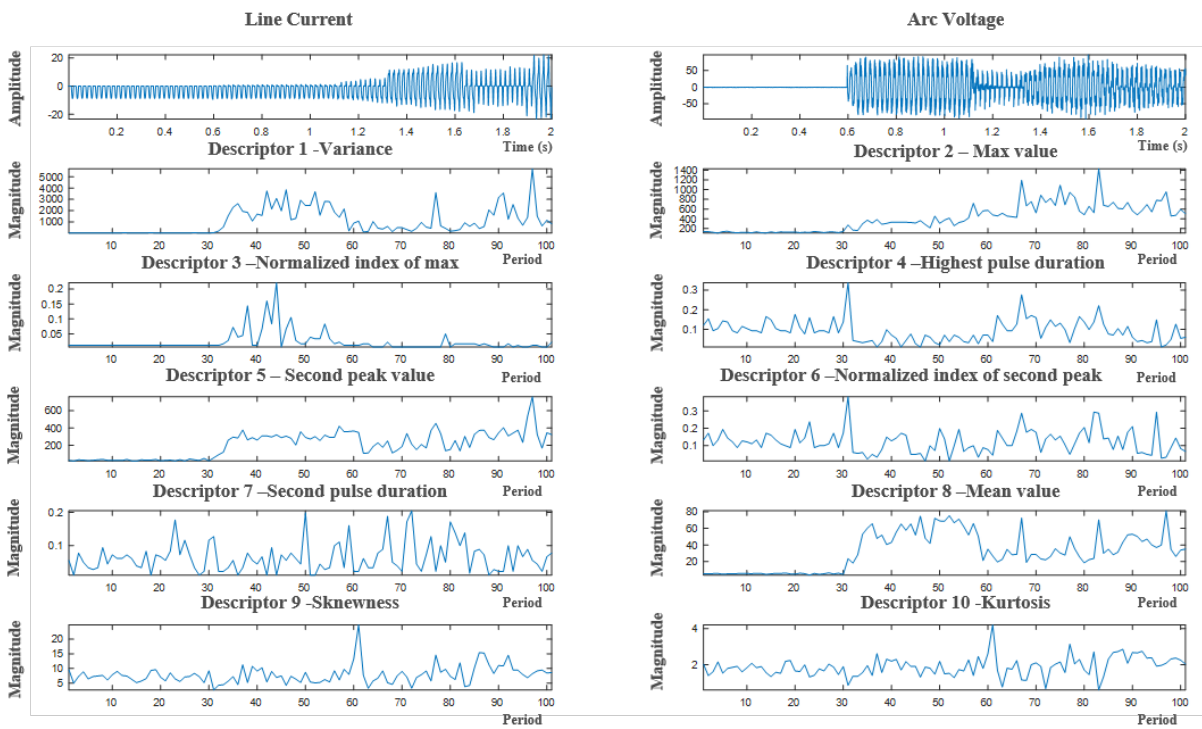

Figure 2.14: Example of ten descriptors, DFT frequency band 1-10 KHz with non-arc and arc fault signal. a-Current without arc fault at steady state; b-Current with arc fault at steady state; c-Current with arc fault and transient (change of functioning mode).

2.4.2.3 Neural network wrapper descriptor selection

It was our objective to select the set of descriptors that provides the best performance for each transform selected in this thesis; irrelevant descriptors for each transform were eliminated. In order to accomplish this, a fully connected feed-forward artificial neural network (ANN comprising four layers) was used as the classifier.

The first hidden layer is composed of 20 neurons and the second of 8 neurons (Figure 2.16). The inputs of the neural network are composed of the arc features obtained from the different descriptors whose calculation procedure is explained in Figure 2.15. The analysis is based on the line current without arcing (labeled 0) or with arc (labeled 1). The calculation is performed in a window of 20 ms for each descriptor. Three successive periods (60 ms) with the same label provide three different sets of values ($\{z_i\}, \{z_{i+1}\}, \{z_{i+2}\}$). Based on the ten descriptors selected, the neural network, therefore, has 30 entries. One of the 10 descriptors was subsequently excluded and this was done for each of the ten descriptors. In each case, the newly generated subset is composed of 27 values for the ANN input. Two or more descriptors can then be removed from the list for analysis.

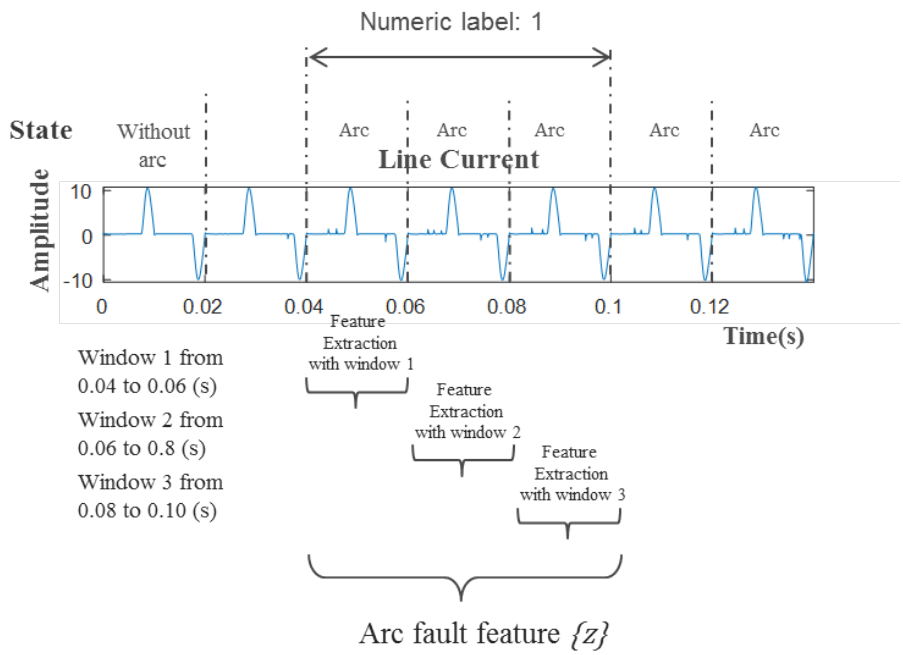

Figure 2.15: Label and arc fault feature construction.

For each generated subset, the mean squared error (when the value of the ANN

output equals LP) is estimated according to the equation (2.2.1a). In this experiment, $n_2 = 16231$ and for any given sample m of the database, the squared error can be calculated as follows:

$$SquaredError(\{z_i\}) = \begin{cases} (1 - LP)^2 \text{ when an arc occurs} \\ (0 - LP)^2 \text{ when no arc occurs} \end{cases}$$

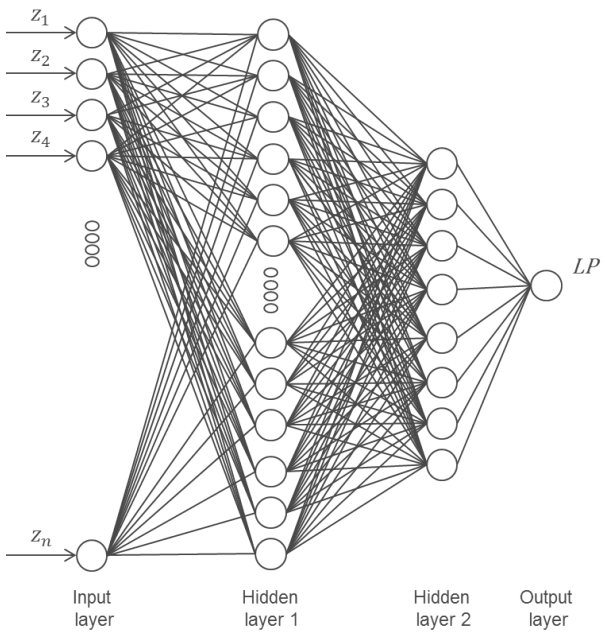

Figure 2.16: Neural-network for arc fault detection.

Figure 2.17 shows the results obtained from the error depending on the descriptors considered. A "removed descriptor" of 0 indicates that all the 10 descriptors have been considered, 1 refers to the fact that the first descriptor has been removed and so on. In some cases, two or more descriptors can be removed with the backward elimination algorithm mentioned above. The results are presented with respect to the frequency band rather than by the type of transform.

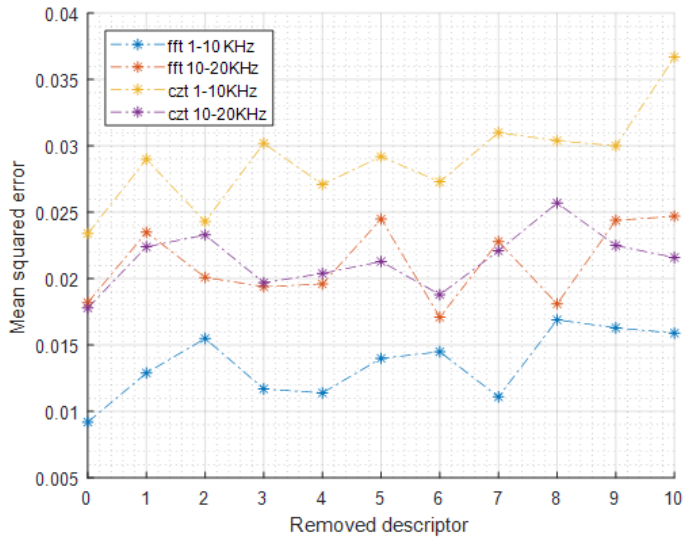

Figure 2.17: Descriptor selection with low frequency FFT and CZT (1-20 KHz).

At the frequency band 1-10 kHz the FFT has lower MSE than CZT. For the frequency band 10-20 kHz, FFT, and CZT give similar results (the red and magenta lines respectively), their respected minimal MSE are 0.0171 and 0.0178. The optimized descriptor sets are almost the same (all descriptors are important) except for FFT at 10-20 kHz. Descriptor 6 should be removed in order to achieve the lowest MSE (Figure 2.17).

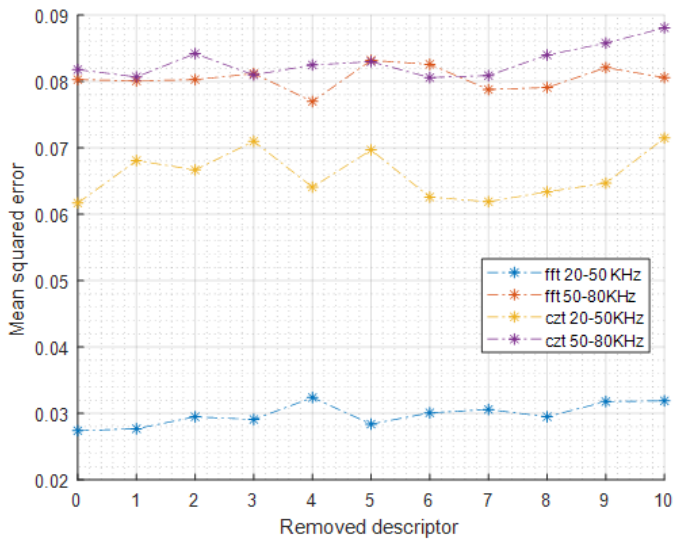

Figure 2.18: Descriptor selection with middle frequency FFT and CZT (20-80 KHz).

The frequency bands 20-50, 50-80 kHz show poorer performance compared to the 1-10 kHz and 10-20 kHz band. All descriptors are necessary for FFT and CZT at the frequency band 20-50 kHz. The descriptor 6 needs to be removed for CZT 50- 80 kHz. The analysis of the results obtained from FFT in the band 50-80 kHz shows that descriptors 2,3,5,6,9 and 10 are relevant for the detection (Figure 2.18). It remains to be decided whether the other descriptors should be retained or not. The process is illustrated in (Figure 2.19). By considering the set that contains all descriptors (z), the mean square error is equal to 0.0803. When one of the descriptors 1, 4, 7 or 8 is removed, the error decreases (z_1, z_4, z_7, z_8). Three subsets, derived from subset z_1 , can be generated, namely $z_{1,4}, z_{1,7}, z_{1,8}$ (descriptor {1,4}; {1,7} or{1,8} removed). Only the subset $z_{1,7}$, reduces the error (MSE = 0.077 lower than 0.081) and subsets $z_{1,4}$ and $z_{1,8}$, show higher errors in comparison to the parent subset; therefore descriptor 4 or 8 should not be removed with descriptor 1. As a result, subsets $z_{1,4,7}$ and $z_{1,7,8}$ were not evaluated. The subsets derived from subset z_4, that is subsets $z_{4,1}, z_{4,7}, z_{4,8}$ can be created. The subset $z_{4,1}$ was evaluated before and the error of subset $z_{4,7}$, and $z_{4,8}$ is yet to be found. Since their errors are higher than the error obtained with z_4, there is no need to evaluate further. Similarly, the last subset $z_{7,8}$ has a higher error than the error obtained with z_8. In conclusion, descriptors 1 and 7 should be removed to achieve the best performance.

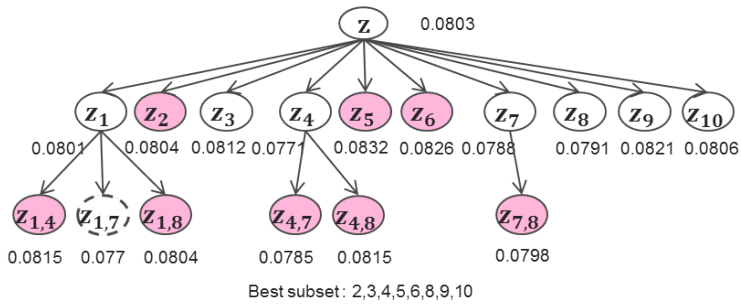

Figure 2.19: Descriptor selection process.

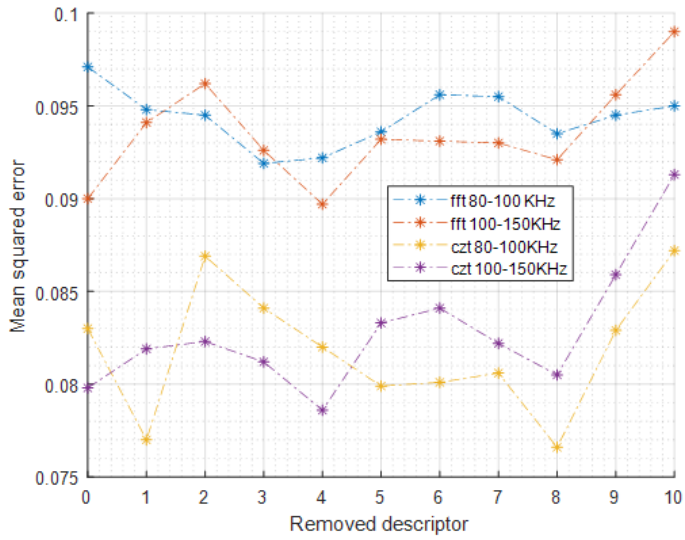

Figure 2.20: Descriptor selection for high frequency FFT and CZT (80 - 150 KHz).

Figure 2.20 shows that the detection performances of frequency bands 80-100 and 100-150 kHz are mostly identical to the results obtained with frequency bands 20 -50 and 50-80 kHz. Some descriptors, such as 8, 6, and 4, are less efficient for this band.

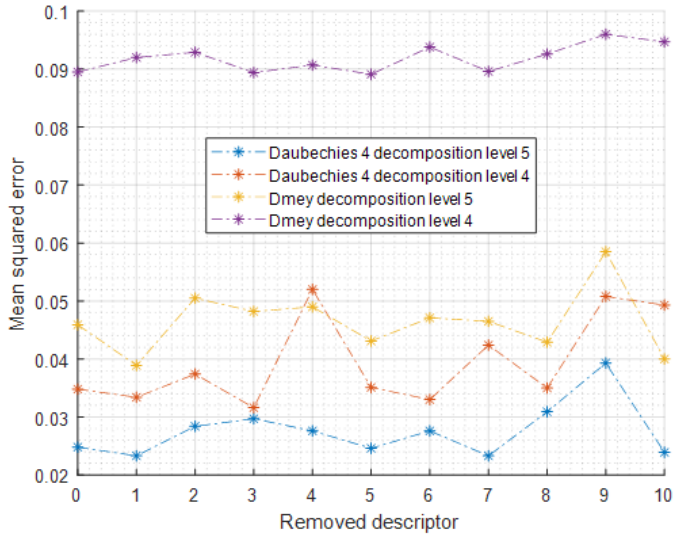

Figure 2.21: Descriptor selection for wavelet transforms-decomposition level 4 and 5.

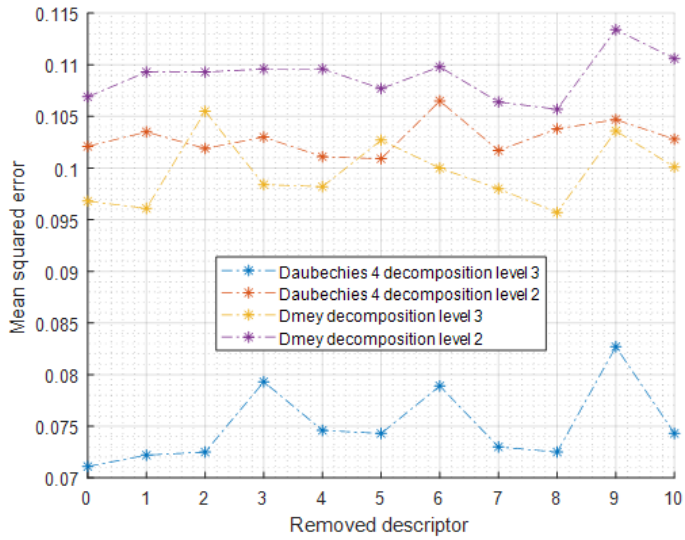

Figure 2.22: Descriptor selection for wavelet transforms-decomposition level 2 and 3.

Same as FFT and CZT, wavelet transform shows better performance at the lower frequency band (high level of decomposition). According to the results, the Daubechies 4 wavelet gives better results than the Dmey wavelet but overall performance lower than FFT and CZT. For the decomposition level 5 of Daubechies 4 and Dmey wavelets the respected descriptor 7 and 1 should be removed. For the decomposition level 2, 3 and 4 the descriptor 5, 8 are most the irrelevant (Figure 2.21 and 2.22).

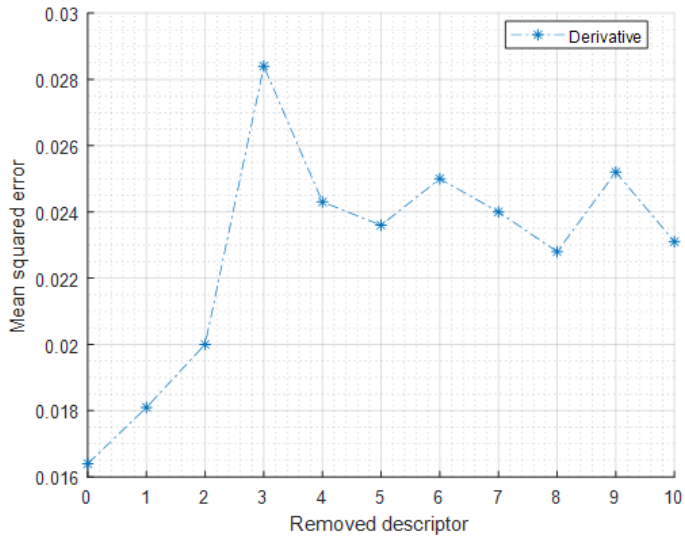

Figure 2.23: Descriptor selection for derivative.

Figure 2.23 shows the results from the transform based on the derivative analysis. The minimum value of mse (0.016) is obtained when all descriptors are used together. The performance of derivative transform is good and only slightly lower than that obtained with FFT at frequency band 1-10 kHz. Table 2.1 summarizes the results obtained for all the analyzed transforms.

Transform	Optimized descriptor list	MSE	Accuracy	Transform	Optimized descriptor list	MSE	Accuracy
FFT 1-10 kHz	1,2,3,4,5,6,7,8,9,10	0.0092	99.11%	CZT 100-150 kHz	1,2,3,5,6,7,8,9,10	0.0786	89.56%
CZT 1-10 kHz	1,2,3,4,5,6,7,8,9,10	0.0234	97.60%	DB4 LVL5	1,2,3,4,5,6,8,9,10	0.0233	97.58%
FFT 10-20 kHz	1,2,3,4,5,7,8,9,10	0.0171	98.13%	Dmey LVL5	2,3,4,5,6,7,8,9,10	0.0389	95.57%
CZT 10-20 kHz	1,2,3,4,5,6,7,8,9,10	0.0178	98.12%	DB4 LVL4	1,2,4,5,6,7,8,9,10	0.0316	96.34%
FFT 20-50 kHz	1,2,3,4,5,6,7,8,9,10	0.0274	96.87%	Dmey LVL4	1,2,3,4,6,7,8,9,10	0.0891	88.59%
CZT 20-50 kHz	1,2,3,4,5,6,7,8,9,10	0.0617	92.07%	DB4 LVL3	1,2,3,4,5,6,7,8,9,10	0.0711	91.59%
FFT 50-80 kHz	2,3,4,5,6,8,9,10	0.077	91.54%	Dmey LVL3	1,2,3,4,5,6,7,9,10	0.0957	87.90%
CZT 50-80 kHz	1,2,3,4,5,7,8,9,10	0.0806	89.68%	DB4 LVL2	1,2,3,4,6,7,8,9,10	0.1009	87.75%
FFT 80-100 kHz	1,2,3,4,5,6,7,8,9,10	0.0912	89.68%	Dmey LVL2	1,2,3,4,5,6,7,9,10	0.1057	87.20%
CZT 80-100 kHz	1,2,3,4,5,6,7,9,10	0.0766	90.04%	Derivative	1,2,3,4,5,6,7,8,9,10	0.0164	98.26%
FFT 100-150 kHz	1,2,3,4,5,7,8,9,10	0.0897	89.80%				

Table 2.1: Arc faults features extraction.

Many transforms keep all descriptors in order to achieve superior performance. A redundant descriptor was added to demonstrate the efficiency of the selection process. Descriptor 11 is equal to (1 - descriptor 2); the value 1 was chosen because

the value of descriptor 2 is normalized between [0; 1] and the result is shown in Table 2.2. The MSE of each transform is increased when the redundant descriptor is added. Removing descriptor 2 may decrease the error in some cases. This may be explained by the fact that descriptor 11 contains information on descriptor 2. Removing any other descriptor other than descriptor 11(for example descriptor 9) increases the error.

Using redundant descriptors may affect detection performance. This phenomenon has been studied with theoretical analysis and empirical evidence on several research papers [42]. The redundant descriptor does not provide additional information and can perturb the learning algorithm. During the learning process, the interesting relationships between the relevant descriptors may be partially ignored by the presence of a redundant descriptor which, due to redundancy, will increase its weight in network learning. This will, therefore, reduce the performance.

Transform	MSE		
	All descriptor	Descriptor 2 removed	Descriptor 9 removed
FFT 1-10 kHz	0.0118	0.0117	0.0145
CZT 1-10 kHz	0.0256	0.0262	0.0347
CZT 10-20 kHz	0.0212	0.0239	0.028
FFT 20-50 kHz	0.0262	0.0269	0.032
CZT 20-50 kHz	0.0662	0.0636	0.0689
FFT 80-100 kHz	0.0954	0.0928	0.0955
DB4 LVL3	0.0725	0.0718	0.0827
Derivative	0.0176	0.0175	0.0232

Table 2.2: Descriptor selection with redundant descriptor.

2.4.2.4 Features combination

Descriptor selection shows that the most efficient AFF is discrete Fourier transform with all descriptors at frequency band 1-10 kHz (noted as AFF1). One way to achieve better performance on the detection task consists in combining arc fault

features, including AFF1. As mentioned above, the method to find complementary AFF is based on the sum of ratio distances between true and false positives, false negatives and true negative groups of AFF1 after concatenation. There are 42 FP and 101 FN samples when only AFF1 is used for the classification.

The results with one complementary AFF are shown in Table 2.3. If we examine precisely the time series that led to an FN or FP for AFF1 and the other AFFs, we see that the time series leading to a decision error are partialy different from one AFF to another. The combination of several AFFs should, therefore, improve the quality of the decision. The sums of ratio distance are listed in the fourth column; in this case, AFF1 with the CZT transform 10-20 kHz has the highest sum of ratio distance. As expected, this combination has the lowest mean squared error and the highest accuracy. The combination with the derivate method also gives an exceptional result. Overall, CAFFs always give better results than single AFFs. The best accuracy can be achieved with a single AFF of 99.11% and an MSE equal to 0.0092. With CAFF the accuracy can go higher, up to 99.81%, and lower the MSE to 0.00173. The combination of AFF1 and derivative leads to a higher MSE but is more accurate compared to Debauchie's wavelet. This can be explained by the fact that all classifiers have been trained with MSE as objective functions and therefore accuracy is not optimized.

Complement AFF	Shared FP with AFF1	Shared FN with AFF1	Sum of ratio distance	MSE	Accuracy
CZT 10-20 kHz	21	11	1.51	0.00173	99.81%
CZT 1-10 kHz	16	13	1.44	0.00293	99.70%
DB4 LVL5	35	13	1.43	0.00386	99.61%
Derivative	13	6	1.36	0.00396	99.70%
FFT 10-20 kHz	30	10	1.34	0.0041	99.58%
FFT 20-50 kHz	21	27	1.34	0.00452	99.56%

Table 2.3: Arc faults features and ratio distance n^o1.

The results with two complementary AFFs are shown in Table 2.4. The best performance combination is composed of CZT 10 – 20 kHz, DB4 LVL5 and FFT

1-10 kHz. As expected, this combination has the highest sum of ratio distance. The overall results are slightly better compared to the combination of two AFFs. Variation in performance gain across different combinations of AFFs can be observed. In several cases, using two complementary AFFs perform worse than the best performance combination at the previous stage.

Complement AFFs	Sum of ratio distance	MSE	Accuracy
CZT 10-20 kHz & DB4 LVL5	1.52	0.00049	99.85%
CZT 10-20 kHz & CZT 1-10 kHz	1.49	0.00066	99.84%
CZT 1-10 kHz & DB4 LVL5	1.43	0.001	99.83%
CZT 10-20 kHz & Derivative	1.41	0.00112	99.84%
DB4 LVL5 & FFT 10 - 20 kHz	1.39	0.00145	99.82%
CZT 10-20 kHz & FFT10-20 kHz	1.38	0.00151	99.81%
CZT 1-10 kHz & Derivative	1.37	0.00158	99.80%
DB4 LVL5 & Derivative	1.37	0.0016	99.80%
Derivative & FFT 10 - 20 kHz	1.37	0.0016	99.80%
CZT 10-20 kHz & FFT20-50 kHz	1.36	0.00163	99.81%
CZT 1-10 kHz & FFT 20-50 kHz	1.36	0.00185	99.73%
CZT 1-10 kHz & FFT 10-20 kHz	1.34	0.002	99.75%
DB4 LVL5 & FFT 20 - 50 kHz	1.34	0.0023	99.71%
Derivative & FFT 20 - 50 kHz	1.33	0.003	99.70%
FFT 10-20 kHz & FFT 20-50 kHz	1.24	0.00402	99.60%

Table 2.4: Arc faults features and ratio distance $n^o 2$.

In this experiment, the number of AFFs used for combination steps is limited to 3. Using more than three AFFs for combination can give better results in the training process but the ANN tends to lead to over-fitting (accuracy on the training set is higher than accuracy on the testing set). This phenomenon can be explained by the fact that adding more AFFs consequently increases the size of the input vector and the ANN becomes more complex. More data is therefore needed to

correctly train the ANN. Approximately 10 minutes of training time is required for each CAFF made from 2 AFFs and 30 minutes for CAFF when composed of 3 AFFs. Eight and a half hours are required to find the optimal CAFF with the exhaustive selection method whilst the distance evaluation method requires only 42 minutes. This demonstrates the usefulness of the proposed AFF combination method.

2.5 Conclusion

In this chapter, a methodology for optimizing arc fault detection performance with multiple arc fault features has been presented. The main originality of the proposed method is the use of supervised feature selection. The method consists of creating an arc fault feature pool and finding a combination of those features which satisfy the desired performance.

The wrapper selection method was first used on every transform to find the most efficient descriptor set. This selection step examines all possible descriptors and removes the irrelevant or redundant descriptors. This step was necessary to build a reliable set of arc fault features. Secondly, a supervised selection method based on Euclidean distance was used to find an appropriate combination in the pool of arc fault features. The combination of several arc fault features helps to reach a detection performance that cannot be achieved by using only one arc fault feature. Experimental results with basic (standard IEC 62606) and complicated situations (transient, multiple masking loads or disturbance on power network, etc.) have demonstrated the efficiency of the proposed methods. Twenty-one specific transforms associated with 10 different descriptors were evaluated and in terms of accuracy, the combination of FFT, CZT, and DB4 can reach 99.85%.

Chapter 3

Arc fault detection with deep neural-networks

In the previous chapter, we have seen that optimized input features are important for arc fault detection. Also the artificial neural network is an efficient method for classifying arc fault and normal situation with well-engineered features (features developed with domain knowledge). In this chapter, we investigate the possibility of obtaining the input features automatically from the data, therefore, reduce the need for feature engineering and domain-specific knowledge. With the rise of deep learning, several approaches to resolve our classification problem have been presented. In general, these approaches replace entirely or a part of the feature extraction step. They aim to perform the feature learning process while tuning the discriminative classifier. Among different techniques, auto-encoders, convolutional neural network (CNN), and echo state networks are most commonly used.

3.1 Auto-encoders

One of the most common approaches to using deep learning as feature extraction is stacked auto-encoder [64] [61]. An auto-encoder is a neural network that learns useful representation in an unsupervised manner. It always consists of two parts: encoder and decoder (Figure3.1).

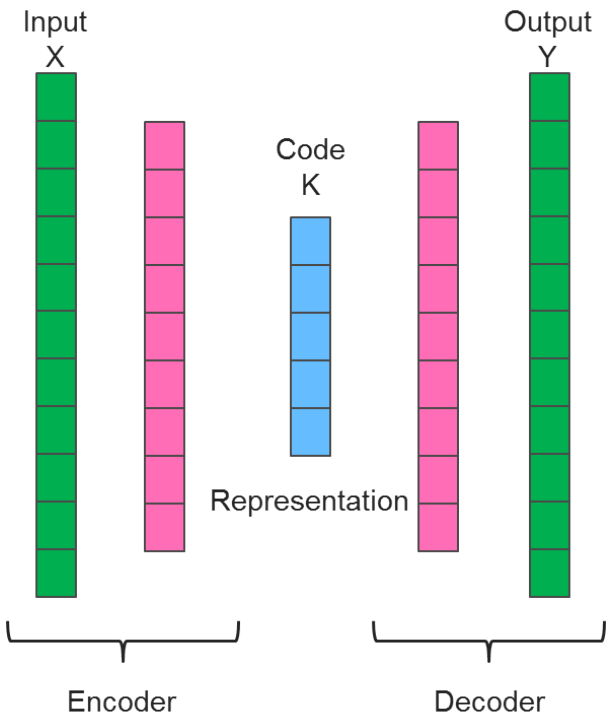

Figure 3.1: Auto-encoder architecture.

The input vector X is mapped to a representation K; $e : X \to K$. The function e is the encoder function. The presentation K is then mapped to the output Y; $d : K \to Y$. d is called the decoder function. In general, auto-encoders are trained to minimize the cost function:

$$C\left(X, d\left(e\left(X\right)\right)\right) = \left|Y - X\right|^2 \tag{3.1.1}$$

If the feature space K has lower dimension than X then $e(X)$ can be considered as a compressed representation of X. In this case, if the autoencoder uses only linear activation it may give a similar result to the principal component analysis algorithm. There are several different techniques to train auto encoder such as denoising auto-encoder, sparse auto-encoder, contractive autoencoder. The denoising auto-encoder is trained to reconstruct the input from its corrupted version $\tilde{X}$ rather than the original input X. The cost function of the denoising auto-encoder is:

$$C\left(X, d\left(e\left(\tilde{X}\right)\right)\right)$$

Where $\tilde{X}$ is a version of X with added noise. The sparse autoencoder is trained with sparsity regularization. Sparse autoencoders normally have more hidden units

than inputs, but the average activation rate of the hidden unit is small. To train the sparse auto-encoders, the Kullback–Leibler divergence between the desired sparsity rate and the average activation of the hidden unit can be used. Let the average activation of hidden unit j for the input x_i:

$$\hat{\rho}_j = \frac{1}{k}\sum_{i=1}^{k} a_j(x_i) \tag{3.1.2}$$

With a_j is the activation of hidden unit j, k is the number of data samples in the training set. The cost function to train sparse auto-encoder is now:

$$C\left(X, d\left(e\left(X\right)\right)\right) + \beta \sum_{j=1}^{n} KL(\hat{\rho}_j || \rho)$$

With $C\left(X, d\left(e\left(X\right)\right)\right)$ is defined in equation 3.1.1, ρ is the desired sparsity rate and n is the number of hidden unit. β controls the influence of penalty term on the learning process. The contractive encoder is trained to be less sensitive to slight variations of input samples. To achieve this, a penalty term is added to the cost function. The contractive auto-encoder uses the squared Frobenius norm of the Jacobian matrix of the encoder activation with respect to the input [65]. The cost function of contractive auto-encoders is defined as following:

$$C\left(X, d\left(e\left(X\right)\right)\right) + \beta \sum_{ij} \left\| \frac{\delta h_j(X)}{\delta X_i} \right\|$$

Where h is the hidden presentation of input X after the encoding stage.

3.1.1 Stacked auto-encoders for series arc fault detection

For the arc fault detection application we use auto-encoders as feature extraction. The classical auto-encoders have been chosen because we want the raw input signal to be mapped to lower dimensions and small variations of the input signal may be very important for the detection. For this experiment, we use the database which has been mentioned in paragraph 2.4.1. With the results from previous work on feature optimization, we decide to focus on low-frequency information (1-20 kHz) because they may achieve better detection performance (see Table 2.1). To do that we down-sampled all the signal on the database, the sampling frequency is now

50 kHz. We extended the frequency band to 25 kHz in order to simplify some calculation processes. The following architecture of auto-encoders has been studied for the arc fault detection task (Figure 3.2).

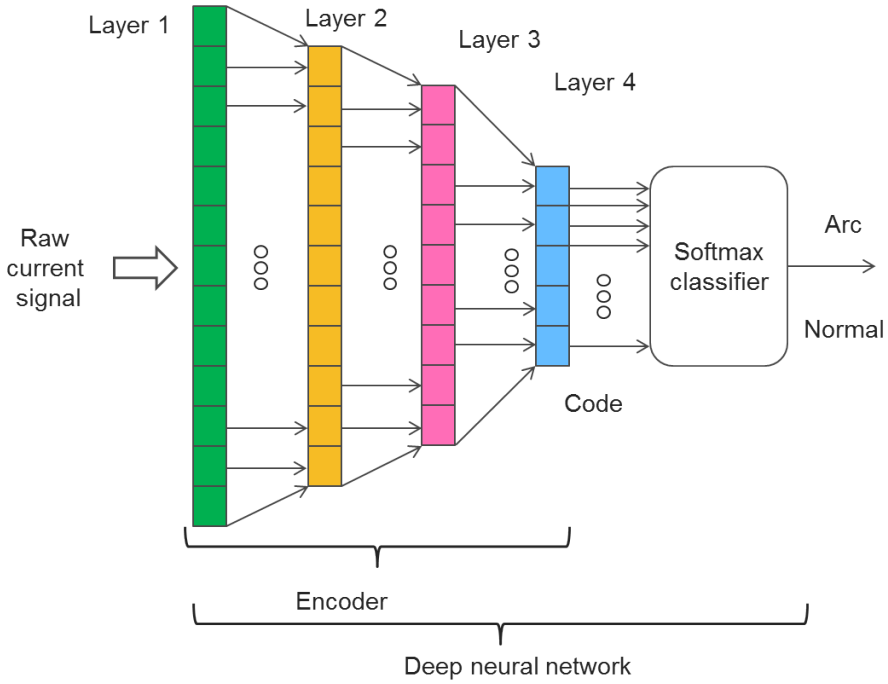

Figure 3.2: Stacked auto-encoder for arc fault detection

The raw current signal is compressed to the final code with one to four-layer of the auto-encoder. A softmax classifier has been used to discriminate between arc fault and normal situations. Each layer of auto-encoder is trained independently, the same for the softmax layer (the layer that uses the normalized exponential function for classification) [66]. The softmax function f: $\mathbb{R}^K \mapsto \mathbb{R}^K$ is defined as following:

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}, i = 1, ..., K \tag{3.1.3}$$

Where x is the input vector of K elements. The deep neural network is created by wiring the outputs of each layer to the inputs of the successive layer. And then the deep network is fine-tuned with the back-propagation algorithm. The database is randomly divided into two parts: training and testing with the ratio of 0.9/0.10. At the beginning, we studied the detection performance with only one layer of auto-encoders. The following code sizes (final representation or input vector size

for classification task) have been tested: 30, 40, 50, 60 and 70. These code sizes are similar to the input vector size used on our previous work (which helped us to achieve very good performance - see Table 2.1 and 2.3).

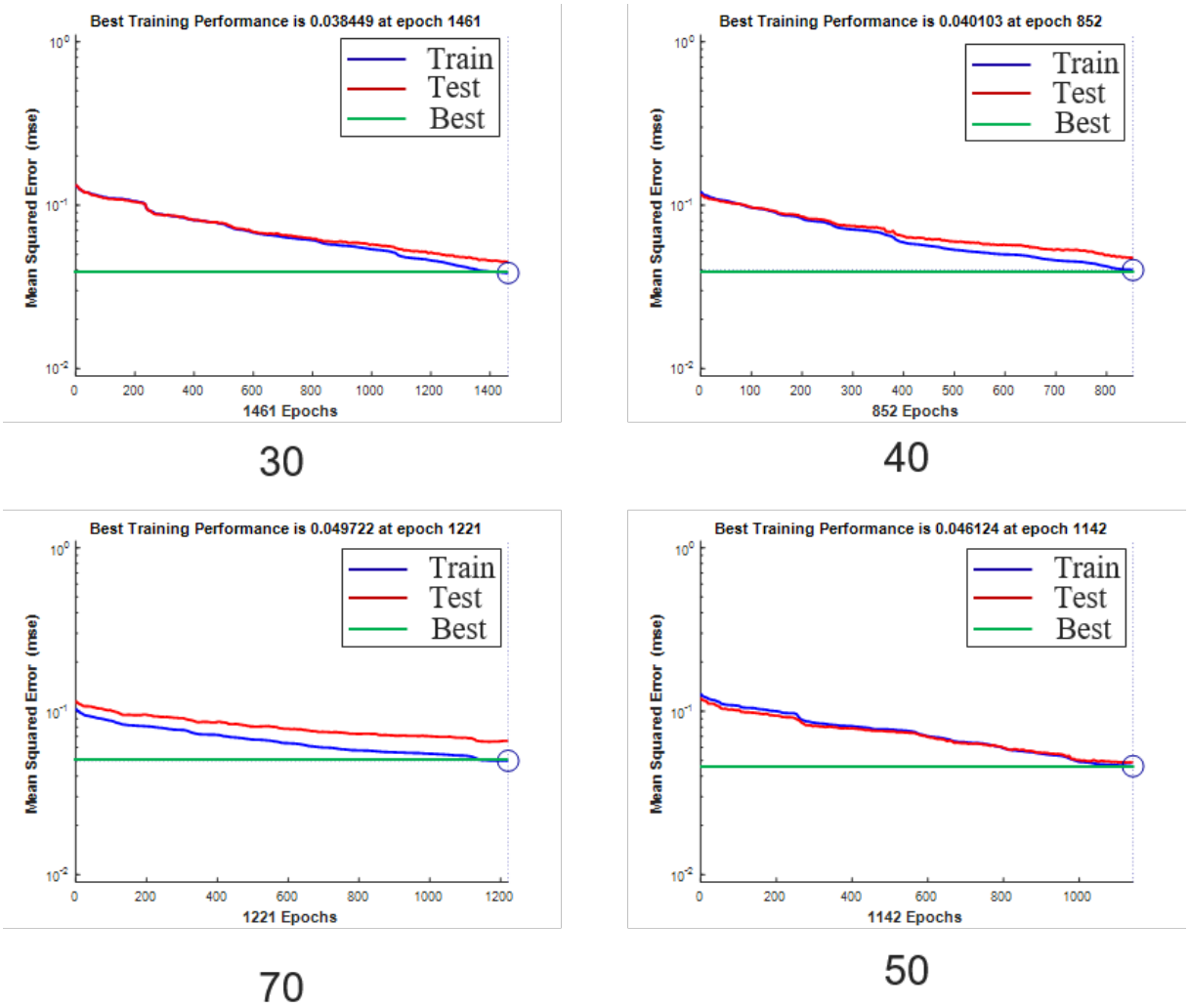

Figure 3.3: Stacked auto-encoders training performance

Table VI Size of code and accuracies					
Code size	30	40	50	60	70
Testing accuracy	94.1%	93.8%	94.8%	93.1%	92.3%
Training accuracy	95.3%	95.1%	94.9%	93.6%	94.3%

Table 3.1: Size of code and accuracies

The results (Table 3.1) show that stacked auto-encoder (SAC) method can

achieve acceptable performance on the detection task. The difference in performance accross different code sizes is noticeable. The more we increase the code size, the network is more difficult to train, this is the reason we can see a drop in the classification performance. Figure 3.3 shows that: at the training process all network tend to over-fitting with the number of iteration increase. The blue line is the mean squared error of classifier on the training set and red for the test set. The bigger the distance between these two lines the more over-fitting network is (memory from training data rather than create a generalized model). Only the network with the size of code equal to 50 seems to have a good generalization. It also has the best testing performance. The best size of code for stacked auto-encoders method in this experiment is 50.

Number of layer	1	2	3	4
Size of layer	50	300-50	300-150-50	600-300-150-50
Testing accuaracy	94.8%	94.3%	96.2%	92.7%
Training accuaracy	94.9%	95.6%	96.6%	93.3%

Table 3.2: Number of layer and accuracies

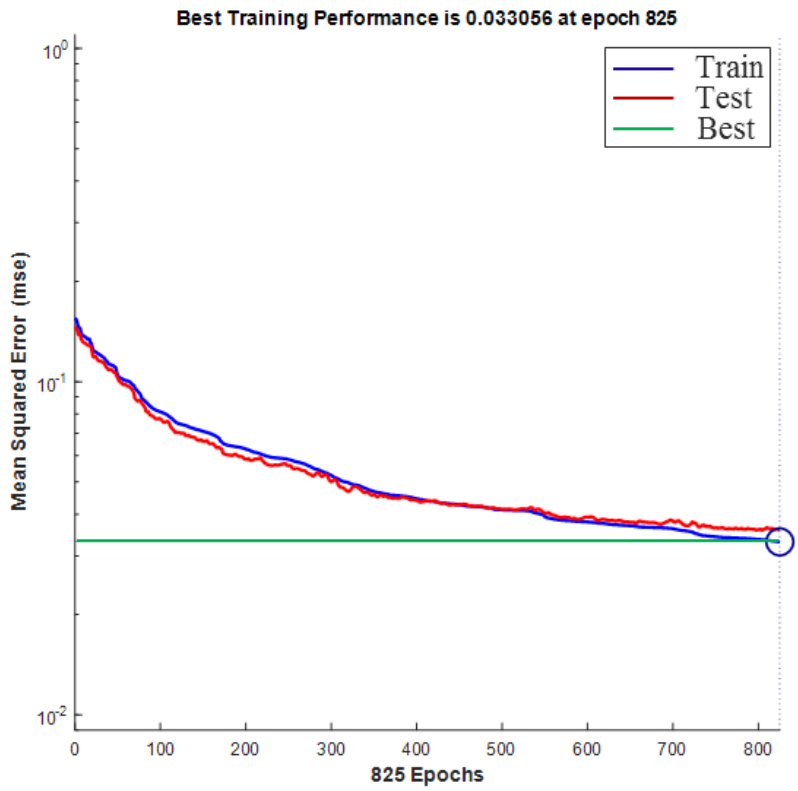

Figure 3.4: 3 layers stacked auto-encoder training performance

We also studied the detection performance with a different number of encoding layers. The size code is fixed to 50 which is the most relevant code size that we found. Compare to shallow networks multiple layer networks can be more efficient [80]. They may learn features with different levels of abstraction and create interesting specific features, therefore, achieve better generalization. On the other hand, they also come with inconveniences like over-fitting and more difficult to train. In this application, we can see with two-layer the average performance is almost similar to one layer. But the network tends to be more fitted on training data. This can be explained by the fact that the second kind of network has more parameters to learn than the one layer. The three-layer SAC has the best performance either on the training and testing process (Figure 3.4 and Table 3.2). Up from three layers, the networks are much harder to train and can't learn interesting features (lowest accuracy on both training and test set).

3.2 Convolutional neural-network

Convolutional neural-network was first designed for image recognition and achieved great success in this field. The time series classification (TSC) problem can also be treated as image recognition if we transform the 1D signal to the 2D texture image. This approach has been used in several research papers and showed state-of-the art accuracy [81]. Images representation of 1D signals can be useful in the case of arc fault detection because we can benefit from some widely used time-series transform such as wavelet transform and short-time Fourier transform. CNNs are similar to classical neural networks, they are made from multiples neurons. CNNs use the raw image pixels as inputs, compare to the multiples perceptron networks CNNs have much fewer parameters. Figure 3.5 describes the architecture of a convolutional neural network.

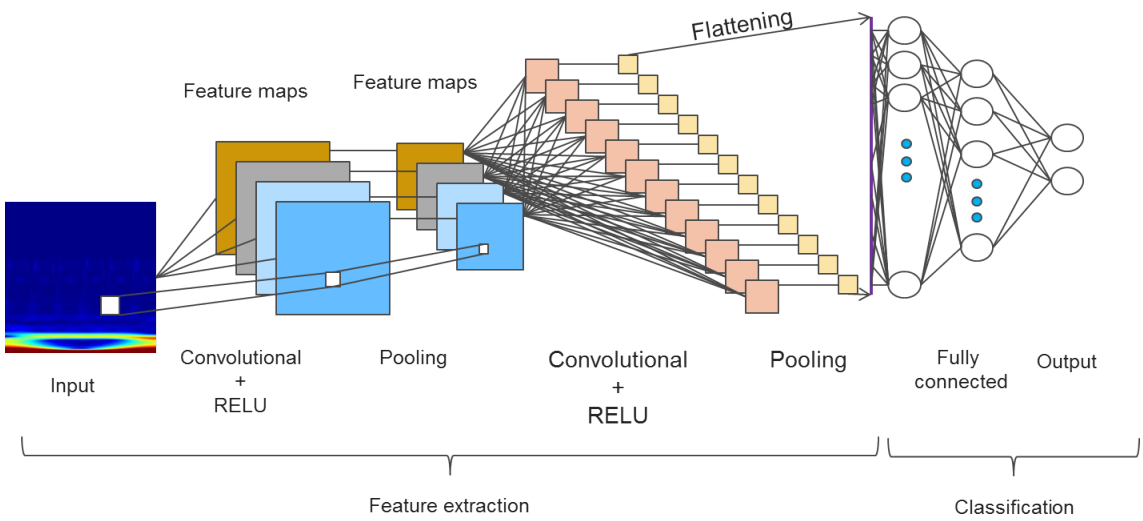

Figure 3.5: Example of a Convolutional Neural-Network

A CNN is composed of a sequence of layers: an input layer, multiple hidden layers, and the output layer. The common layers of CNN are: Convolution Layer (CONV), Pooling Layer, Rectified Linear Units layer (RELU) and Fully-Connected layer (FC). The input layer holds the pixel value of images, it has a variable size that depends on the input images (width, height, color channels). To compute the output for a given neuron, the CONV layer uses convolution operation (Figure 3.6) and the input on neuron's receptive field [67]. RELU layer applies the function

$f(x) = max(0, x)$ in all input elements. The pooling layer reduces the dimension of the input by combining them or sub-sampling (see Figure 3.7). The FC layer can be used to calculate the class score.

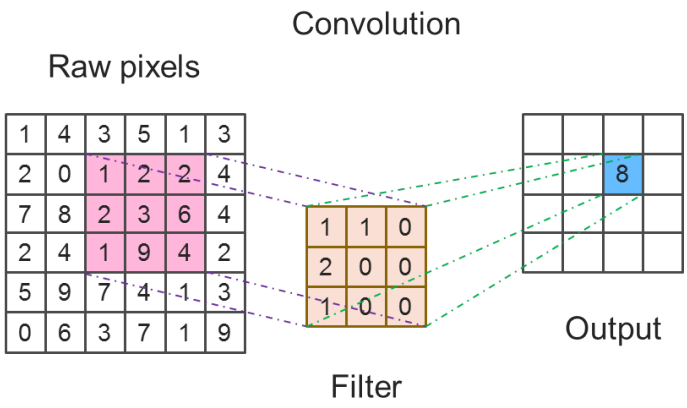

Figure 3.6: Example of convolution on raw pixels

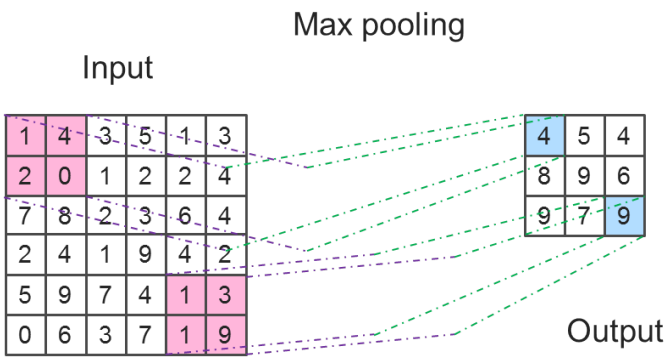

Figure 3.7: Example of max pooling

The samples on the mentioned database (paragraph 2.4.1) have been used in order to create input features (60 ms of current signal). The continuous wavelet transform (CWT) has been chosen to create input images. The scalogram representation (absolute value of CWT) of the signal is treated as an RBG image and used to train a convolutional neural network. The size of an image is 227x227 pixels which have been used in popular CNNs such as GoogLenet or Alexnet. We used also the short-time Fourier transform to create the low-frequency spectrogram of signal and raw current waveform as input. Figure 3.8 shows an example of input images.

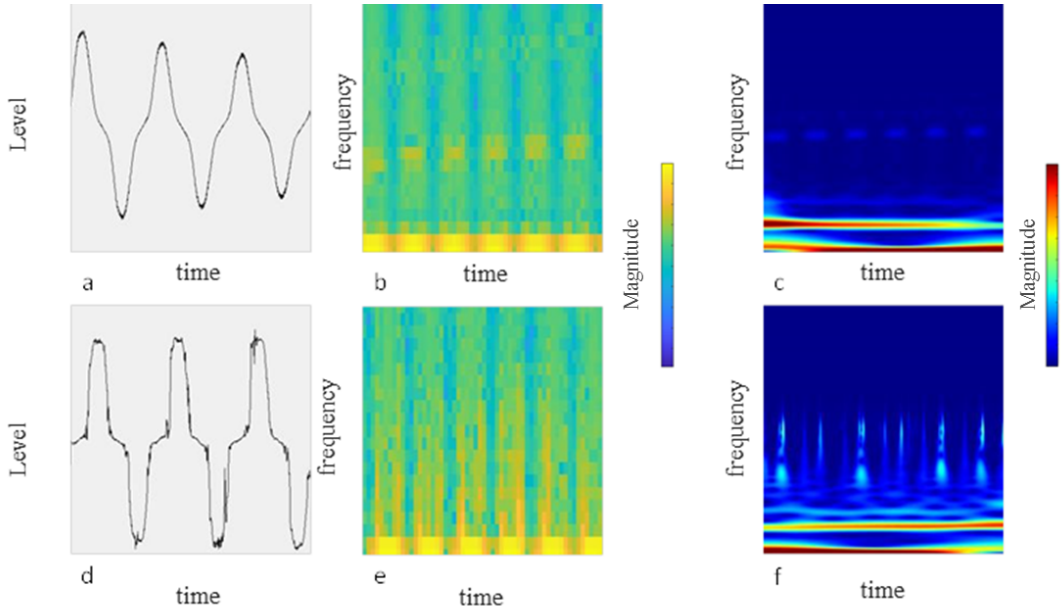

Figure 3.8: Examples of real data, a: Current waveform in the case of normal functioning, b: Spectrogram of signal a in the frequency band 1-10 kHz, c: Scalogram of signal a, d: Current waveform in arcing situation, e: Spectrogram of signal d in the frequency band 1-10 kHz, f: Scalogram of signal d.

Several common features of arc fault can be easily noticed on these images. For example broadband noise in Figure 3.8 - d and f. Zero-crossing in d, low-frequency variation in e. Signal without arc on the transient state has smooth variation (a) and much less frequency component compare to signal with arc (b-e and c-f).

3.2.1 Arc fault detection with transfer learning

In this section, we applied the transfer learning technique for testing the efficient convolutional neural network for the detection task. The transfer learning technique consists of using the knowledge of a trained model. They work surprisingly well in the computer vision field and can produce a boost to the model's generalization [82]. The model could be trained for a different problem. We know that in image recognition, the network frequently learns to detect edges, shapes, and some specific features. For arc fault detection we try to do the same, just with a different set of specific features that could be learned by fine-tuning the network. We adapted GoogLeNet to our arc fault detection problem. GoogLeNet is an efficient CNN which

won the large-scale visual recognition challenge [83]. The network has 22 levels of depth and used about 1.2 million images for training.

Because the specific features are frequently learned in the last layers of network [82] (FC and softmax in this case), we have replaced three last layers of the network by a new fully connected layer, soft-max and output layers (calculates the cross-entropy loss for classification problems with mutually exclusive classes). By default GoogLeNet is configured for 1000 categories, we adapted the output layer to 2 categories (arc fault and normal functioning). We also increased the learning rate of the fully connected layer to learn faster in this layer.

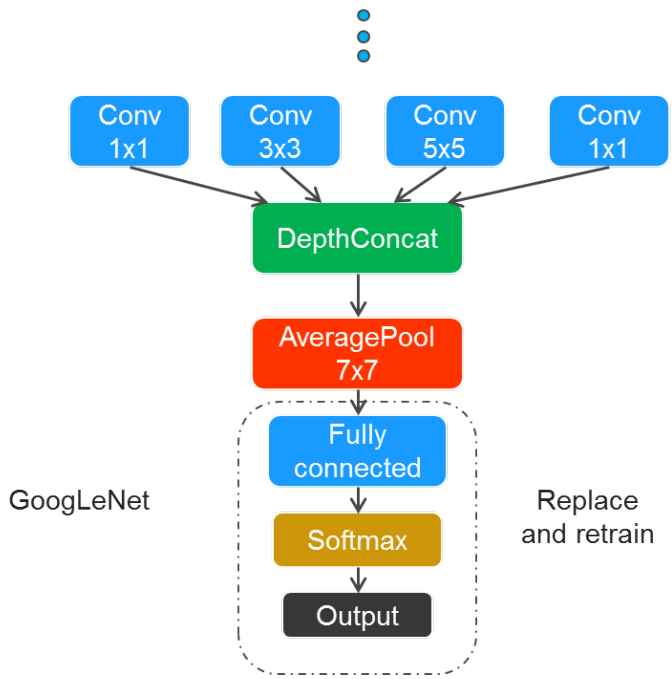

Figure 3.9: Adapted GoogLeNet for arc fault detection

To start the experimentation, we used the scalogram images as input (CNN1). Then the low-frequency spectrogram (CNN2), and finally the raw current waveform is used as input (CNN3).

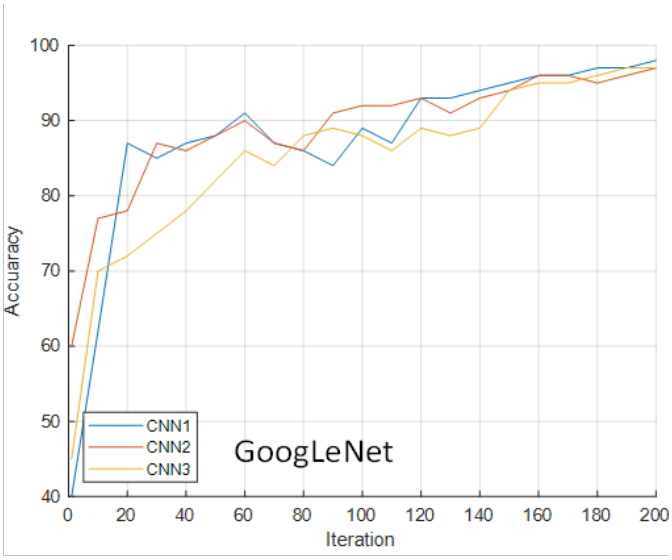

Figure 3.10: Training accuracy

Network	Input feature image	Accuracy
CNN1	Wavelet transform	98.3%
CNN2	Low frequency STFT	98%
CNN3	Raw current	97.1%

Table 3.3: Series arc fault with CNNs classification results

The results (Table 3.3 and Figure 3.10) show that we can achieve a better detection performance compare to SAC (96.6%- Table 3.2) but still lower than ANN with hand crafted features (99.85%- Table 2.4). The training time is about 3 hours for the adapted GoogLeNet.

3.2.2 Arc fault detection from scratch

The promising results from the transfer learning technique confirm the efficiency of using CNN for arc fault detection. We decided to go further with this detection methodology. A smaller CNN has been designed to resolve only the arc fault detection task. We trained the network with the images of the scalograms from our database because they give the best classification performance (98.3%- Table 3.3). We started with a simple CNN consists of one convolution layer, RELU, Max-

pooling, FC and softmax. This CNN is not able to learn specific feature and the performance is very poor (lower than 80% accuracy- Table 3.4).

We designed another CNN which has two convolution layer ($CNNarc_1$ with configuration CONV(7x7,20) (2-D convolutional layer with 20 filters of size 7x7)-RELU-MP(3x3)(Max pooling with filter of size 3x3)-CONV(2x2,20)((2-D convolutional layer with 20 filters of size 7x7)-FC-SOFTMAX(Classification with 2 classes)). The result seems promising with two layers of CONV (Table 3.4 - line 2). The first layer of CONV shows that the network tends to learn feature based on the level of the signal at different time-frequency zone (Figure 3.11). As we can see the different colors which signify the level of the signal in the small zone (the time-frequency zone). We can notice that all squared image-filter are unique which means time-frequency zone has been leaned for each filter.

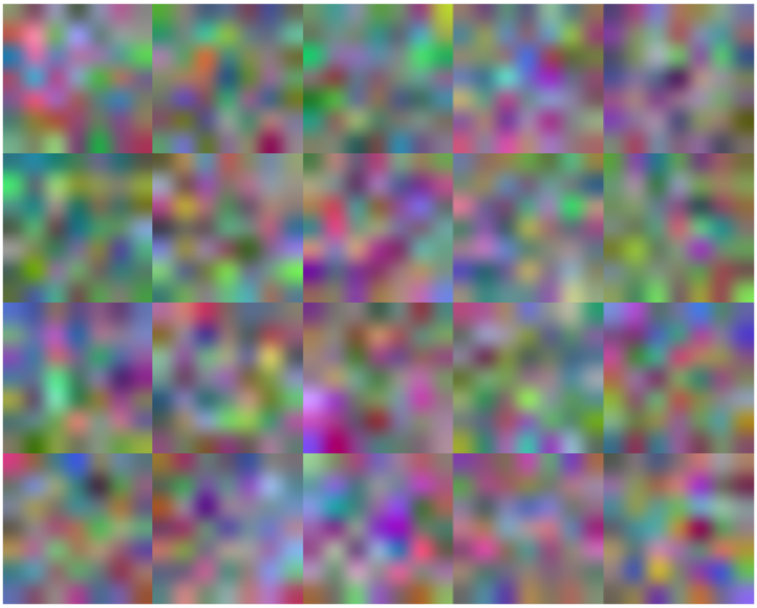

Figure 3.11: First CONV layer of CNN_arc_1 (20 filters of size 7x7)

It's different compared to the filters on GoogLeNet which tend to learn more generic features such as edge (Figure 3.12).

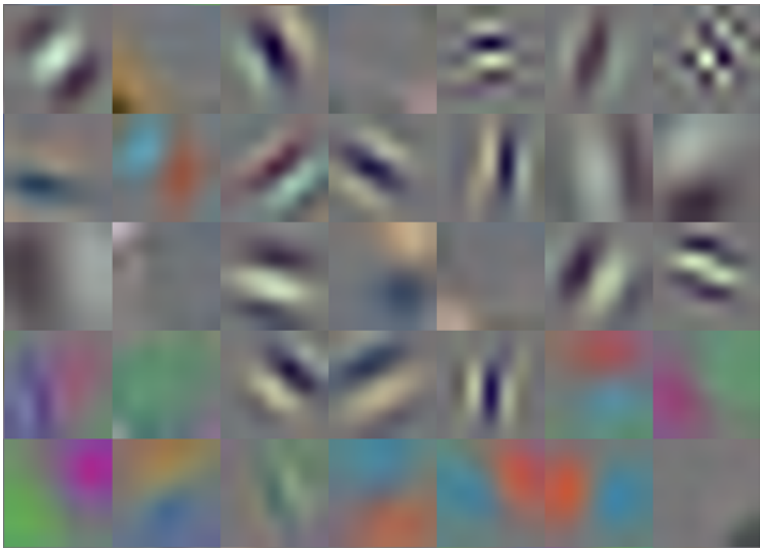

Figure 3.12: Example of filters on first CONV layer weights GoogLeNet

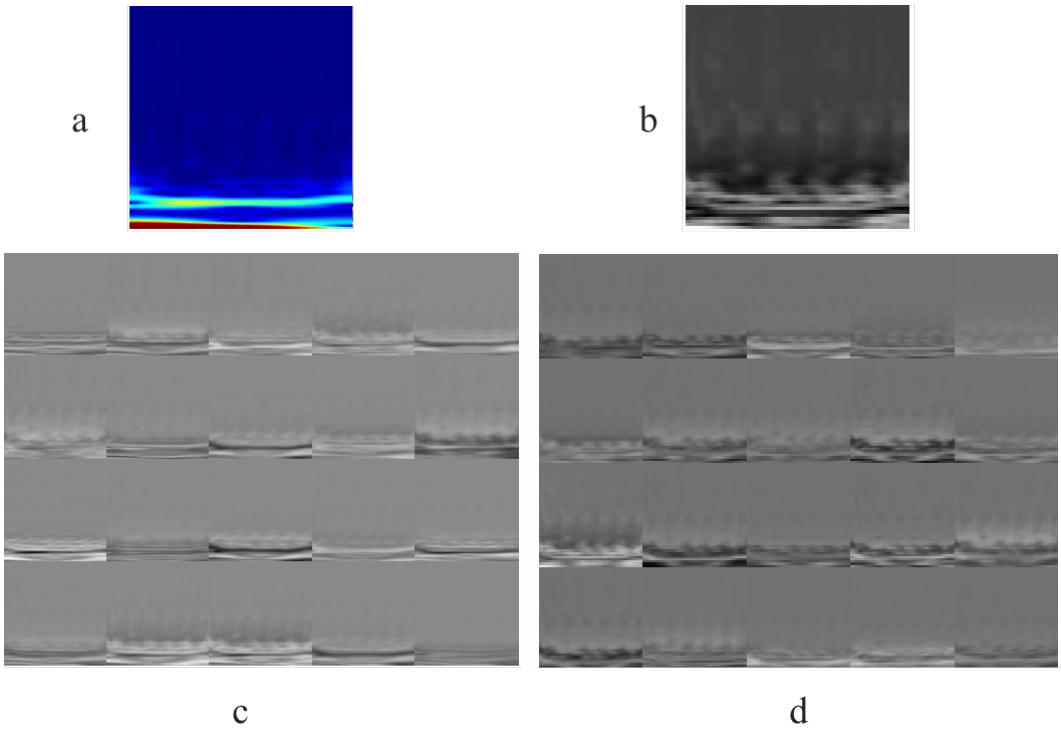

Figure 3.13: Sample and activations; a-Sample, b-Strongest activation after two CONV layers, c-Activation of first CONV layer, d-Activation of second CONV layer.

We can see in Figure 3.13 (c,d) the activations of a sample with arc fault, the network gives attention to the variation of signal in time with not only the low-frequency band but also middle band (activation with black zone near the bottom). With the

network $CNNarc_2$(CONV(10x10,20)-RELU-MP(3x3)-CONV(2x2,20)-FC-SOFTMAX) we can see at first layer the learned feature is similar to the $CNNarc_1$(Figure 3.14 and 3.11). Since the size of filter is higher (10x10 compare to 7x7) $CNNarc_2$ seems has better resolution (more different time-frequency zone - different color zone). The strongest activation after two CONV layer of the $CNNarc_1$ is also slightly different compare to the $CNNarc_2$ (Figure 3.15 and 3.13 -b). The middle frequency band gets more attention from the second network for the same input sample (darker zone near the bottom of the image). Overall, they are very close to each other the network in terms of validation accuracy and training process.

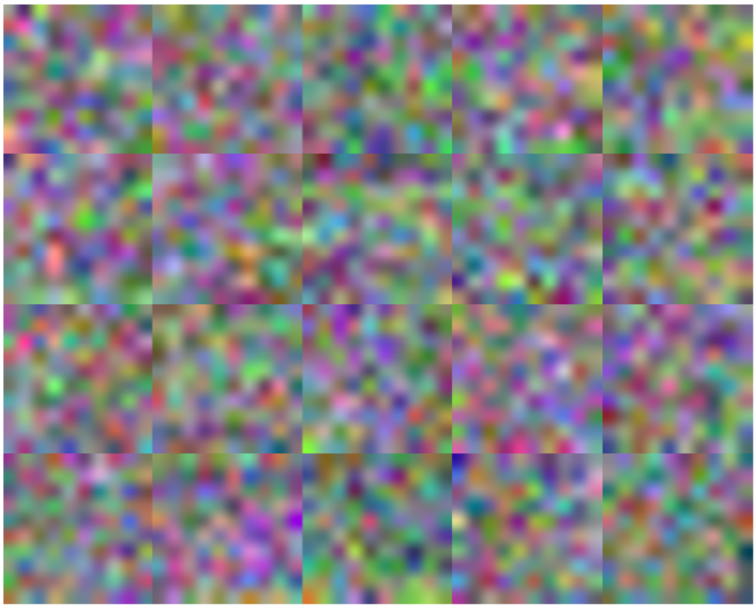

Figure 3.14: First CONV layer weights CNN_arc_2

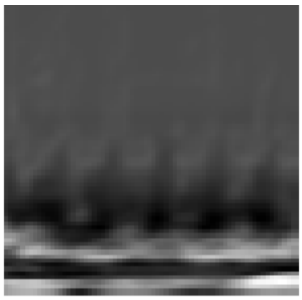

Figure 3.15: Strongest activation after two CONV layers CNN_arc_2

Various of configuration of CNN has been tested, the results are shown on Table 3.4.

Configuration	Validation accuracy
CONV(7x7,20)-RELU-MP(3x3)-FC-SOFTMAX	79.5%
CONV(7x7,20)-RELU-MP(3x3)-CONV(2x2,20)-FC-SOFTMAX	97.24%
CONV(10x10,20)-RELU-MP(3x3)-CONV(2x2,20)-FC-SOFTMAX	97.31%
CONV(10x10,20)-RELU-MP(3x3)-CONV(2x2,20)-RELU-FC-SOFTMAX	96.99%
CONV(7x7,20)-RELU-MP(3x3)-CONV(4x4,20)-RELU-CONV(2x2,20)-RELU-FC-SOFTMAX	96.30%
CONV(8x8,20)-RELU-MP(4x4)-CONV(4x4,20)-RELU-CONV(2x2,20)-RELU-FC-SOFTMAX	97.51%
CONV(6x6,20)-RELU-MP(4x4)-CONV(4x4,20)-RELU-CONV(2x2,20)-RELU-FC-SOFTMAX	96.68%
CONV(8x8,20)-RELU-MP(4x4)-CONV(4x4,20)-RELU -CONV(3x3,20)-RELU-MP(2x2)-FC-SOFTMAX	96.95%
CONV(8x8,20)-RELU-MP(4x4)-CONV(4x4,8)-RELU-FC-SOFTMAX	98.7%
CONV(8x8,8)-RELU-MP(4x4)-CONV(4x4,8)-RELU-FC-SOFTMAX	95.8%
CONV(8x8,24)-RELU-MP(4x4)-CONV(4x4,8)-RELU-FC-SOFTMAX	98.4%
CONV(8x8,24)-RELU-MP(4x4)-CONV(4x4,6)-RELU-FC-SOFTMAX	98.65%
CONV(8x8,16)-RELU-MP(4x4)-CONV(4x4,6)-RELU-FC-SOFTMAX	98.52%
CONV(8x8,16)-RELU-MP(4x4)-CONV(4x4,4)-RELU-FC-SOFTMAX	98.41%
CONV(8x8,12)-RELU-MP(4x4)-CONV(4x4,4)-RELU-FC-SOFTMAX	97.1%
CONV(8x8,20)-RELU-MP(6x6)-CONV(6x6,4)-RELU-FC-SOFTMAX	96.7%
CONV(8x8,20)-RELU-MP(3x3)-CONV(3x3,4)-RELU-FC-SOFTMAX	98.5%
CONV(8x8,20)-RELU-MP(4x4)-CONV(3x3,4)-RELU-FC-SOFTMAX	96.5%
CONV(8x8,20)-RELU-MP(3x3)-CONV(2x2,6)-RELU-FC-SOFTMAX	98.6%
CONV(8x8,24)-RELU-MP(4x4)-CONV(3x3,36)-RELU-FC-SOFTMAX	99.1%
CONV(8x8,24)-RELU-MP(4x4)-CONV(3x3,42)-RELU-FC-SOFTMAX	98.7%

Table 3.4: Series arc fault detection with CNNs

Except for the first network (CONV(7x7,20)-RELU-MP(3x3)-FC-SOFTMAX) which has very bad results because only one CONV layer has been used. The other networks are very close in terms of accuracy and they tend to learn the same feature in the first convolution layer. The most efficient network is the $CNNarc_{19}$(CONV(8x8,24)-RELU-MP(4x4)-CONV(3x3,36)-RELU-FC-SOFTMAX) which has the second-most filter on the second layer. It also takes longer to train than the other network because of its size. As we can see on the Figure 3.16 the network

$CNNarc_1$ and $CNNarc_2$ take about 300 iteration to train. The $CNNarc_{19}$ needs about 500 iterations.

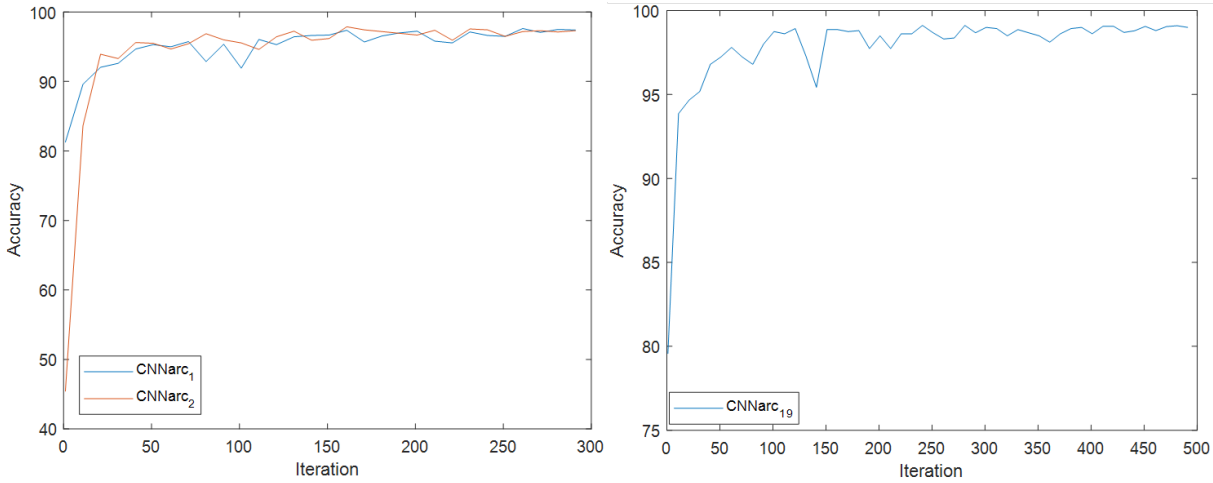

Figure 3.16: Training accuracy

Increase the number of filters on the second layer is not a guarantee to achieve better performance as we can see on the last listed network. In order to achieve better detection performance, we also tried multiple classifier systems (MCS). With the main objective of solving the detection problem by combining a set of classifiers. This idea was introduced by Chow [85], independent classifiers have been combined with defined weights. The performance of MCS has been confirmed early for a neural network with the work of Hansen and Salamon who used neural network ensembles [86], Tumer, and Ghosh showed that averaging outputs of several neural network classifiers can improve the accuracy [87].

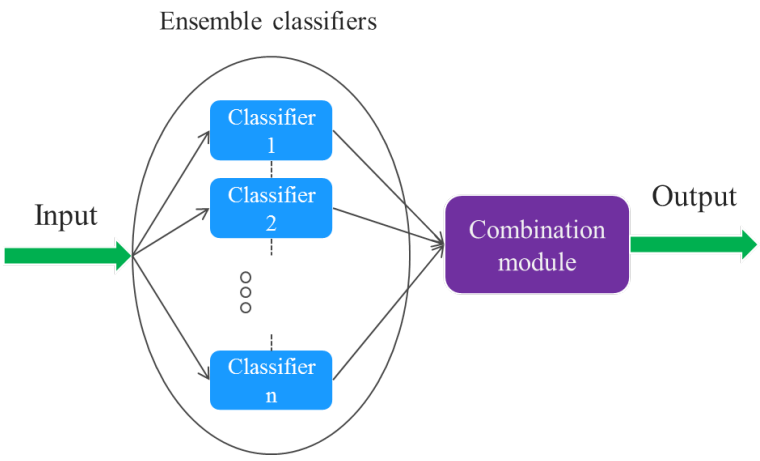

Figure 3.17: Overview multiple classifier systems.

Figure 3.17 shows the overview of MCS system. To design an MCS system we need to know how to make interconnections between classifiers, generate and select a pool of classifiers and build an efficient combination module [88].

The literature mentions several approaches to grouping classifiers, among which we can list:

- The stacking method (stacked generalization) uses multiple generalizers (generalizer can be a classifier) to improve the model performance. This method was presented by Wolpert [91], there are two layers of generalizer. The first layer solves the considered problem. The output of different models in the first layer will be collected and generate a new data set. The second layer resolves the new problem with the generated data set.

- The bootstrap aggregating or bagging method has been introduced by Breiman [89]. It works by generating multiple version of a classifier and using them to get final classifier by a voting system. Classifiers are trained with different training sets. For example, m training set S_i of size k are created from the original data pool S (size n). Each S_i is a set of k observations which are sampled from the S and with replacement. More specific, some observations may be duplicated in S_i. This method of generating the data set is called the bootstrap sample. With m training data set we can train m classifiers (Figure 3.18).

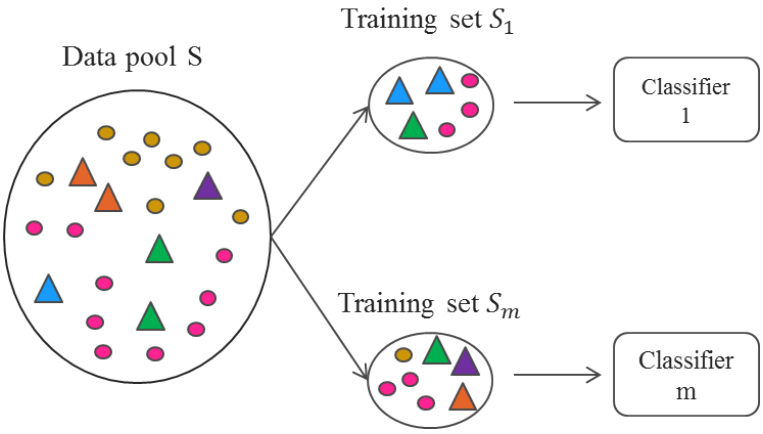

Figure 3.18: Example of boostrapping.

- The boosting technique (see Figure 3.19) combines weak classifiers to create a strong classifier with better performance by learning in the iteration. At first, a weak learner uses all the training data and pay equal attention to each observation. Second, if one or more observation has been misclassified by the first learner we give more attention to these observations and then apply another learner. We repeat this second step until we reached a predefined limit of the base learner. The final step consists of combining these weak learners in order to create a strong classifier. One well-known boosting algorithm is adaptive boosting [90] which achieved wide success in many application [92].

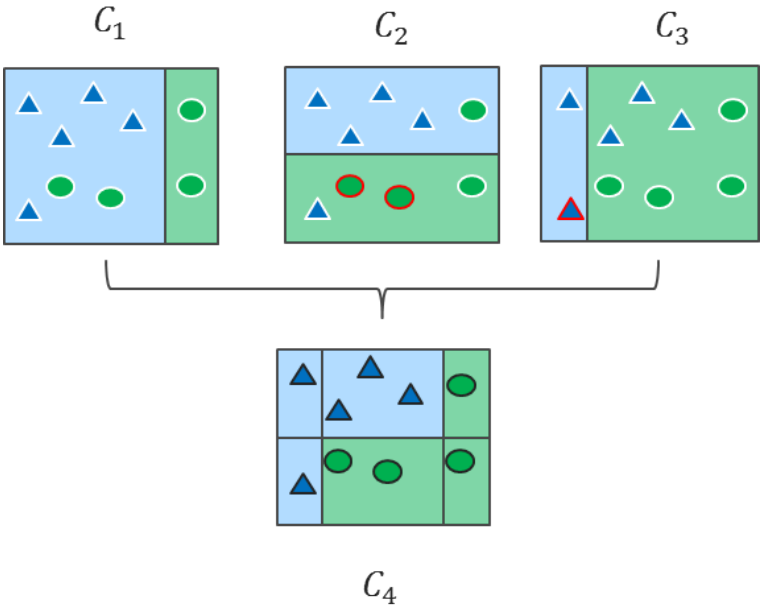

Figure 3.19: Example of boosting.

Red circle samples are misclassified by "weak" classifiers (C_1, C_2)that get more attention at training phase on the next step(C_2, C_3). C_4 the strong classifer is created by combinating the 3 "weak" classifiers.

Ensemble machine learning has been used for arc fault detection in direct current with support vector machine and decision tree [93]. There are also several works that incorporate CNN and MCS [94,96]. Because of the promising results obtained, we decided to investigate this technique with a neural network for the AC detection task. We used adaptive boosting to generate a better detection model, the training procedure is as following:

Step 1-Initialization: for each sample x_i, set label 1 for arc fault and -1 for normal. And weights of sample $w(x_i) = 1/N$ (N is the number of sample)

Step 2-Iteration:(from t = 1 to T)

Train the network $cnn_t(x_i)$ with data using weights distribution $w(x_i)$ Compute the error rate ϵ_t:

$$\epsilon_t = \sum_{i=1}^{N} w(x_i) * I_i$$

With I_i=0 if the given sample x_i is correctly classified by network and 1 otherwise.

Compute the coefficient α_t of network cnn_t:

$$\alpha_t = log(\frac{1-\epsilon_t}{\epsilon_t})$$

For each x_i we adjust the weight:

$$w(x_i) = w(x_i) * exp(\alpha_t * I_i)$$

And then the training data will be re-sampled to guarantee :

$$\sum_{i=1}^{N} w(x_i) = 1$$

To be more specific, we keep all misclassified samples and randomly remove correctly classified samples and then normalize.

Step 3-Combination module: The combination module can be simple as:

$$net_{final}(x_i) = sign(\sum_{t=1}^{T} \alpha_t * cnn_t(x_t))$$

We tested this training process with 12985 samples and 3246 samples for the validation task. The results are shown in table 3.5.

Configuration	Training accuracy	Validation accuracy	Step
CONV(8x8,8)-RELU-MP(4x4)-CONV(4x4,8)-RELU-FC-SOFTMAX	96%	95.8%	1
CONV(8x8,12)-RELU-MP(4x4)-CONV(4x4,4)-RELU-FC-SOFTMAX	96.4%	96%	2
CONV(8x8,16)-RELU-MP(4x4)-CONV(4x4,4)-RELU-FC-SOFTMAX	97.1%	96.7%	3
CONV(8x8,20)-RELU-MP(3x3)-CONV(3x3,4)-RELU-FC-SOFTMAX	97.3%	97%	4
CONV(8x8,20)-RELU-MP(3x3)-CONV(2x2,6)-RELU-FC-SOFTMAX	98%	98%	5
CONV(8x8,24)-RELU-MP(4x4)-CONV(3x3,36)-RELU-FC-SOFTMAX	98.7%	99.12%	6

Table 3.5: Adaptive boost CNN.

As we can see the boosting process may bring better performance for the last network however the gain is not significant. When we use all the CNN along with the combination equation the validation performance may go up to 99.15% which is still lower than the proposed method in Table 2.4. This result is consistent with the other researches because sometimes boosting methods may not make any significant improvement when used on strong classifier [97].

3.3 Conclusion

In this chapter, we investigated two widely used deep learning techniques for the arc fault detection task. The result with stacked auto-encoder is average, this method allows to bypass completely feature engineering task. However, nowadays these features can be found easily in the literature. Detection methods based on CNN bring better performance and we can still improve the prediction accuracy with more training data. The main disadvantage of CNN is computation cost, it's expensive to implement these proposed algorithms on an embedded platform.

Chapter 4

Time variable arc fault detection

In this chapter, we introduce an algorithm of decision that is based on the truncated sequential probability test and an arc fault detection method based on long short term memory. Both approaches take into account the variable time aspect of arc fault detection in order to optimize detection performance. The detailed context can be found at paragraph 1.2.2.

4.1 Truncated Sequential probability ratio test decision

We present an approximate way to estimate the performance of an arc fault detection method which takes into account analysis time.

4.1.1 Fixed time detection method performance

An arc fault detection method can be evaluated by the detection rate and false alarm rate. To completely estimate these parameters we need to take into account the decision part along with the classification part (see Figure 1.14). Let FNR and TPR be the false-negative rate and true positive rate of a given detection method has a classifier C with a single analysis window of size t. FNR and TPR can be

calculated as follows:

$$FNR = \frac{FN}{FN + TP} \tag{4.1.1a}$$

$$TPR = \frac{TP}{FN + TP} \tag{4.1.1b}$$

Since the arc fault indicator calculated from classifier C for each analysis is independently calculated, we may describe the detection rate of the method used k successive event as (see Figure 1.19):

$$DR = TPR^k \tag{4.1.2a}$$

Similarly for the false alarm rate (FA):

$$FA = FNR^k \tag{4.1.3a}$$

with k is the number of successive positive analysis windows needed. DR is the probability of detecting a default after a fixed time k*t.

In case of decision technique based on counting k abnormal events occurred over n intervals we can deduct these variables:

$$DR = \sum_{x=k}^{n} \binom{k}{n} TPR^x (1 - TPR)^{n-x} \tag{4.1.4a}$$

$$FA = \sum_{x=k}^{n} \binom{k}{n} FNR^x (1 - FNR)^{n-x} \tag{4.1.4b}$$

For both case the number of false alarm for a given time T (NFA) can be estimated with the following formula:

$$NFA = \frac{FA}{t * k} * T \tag{4.1.5a}$$

For example a detection algorithm with a 1% FNR and 99% TPR of classification part, analysis window 20 ms. So without the statistics method, we can expect an unwanted trip every 2 seconds. With 4 successive abnormal event counting method for one year we may expect:

$$NFA = \frac{0.01^4}{0.02 * 4} * 3600 * 24 * 365 = 3.942 \tag{4.1.6a}$$

$$DR = 0.99^4 = 0.96 \tag{4.1.7a}$$

Which mean around 4 unwanted trip per years and a detection rate of 0.96% with 80 ms response time. We can see that statistic method may decrease the detection performance in some case, however, if the TPR tends to 1 the method can significantly decrease unwanted trip without losing much sensitivity.

4.1.2 Sequential probability ratio test

Consider the hypotheses for the arc fault detection for a series of n observation:

$H_i : O_1, O_2, O_3, ...O_n \overset{iid}{\sim} p_i$ with $i = \begin{cases} 1 \text{ arc fault} \\ 0 \text{ normal} \end{cases}$ iid stands for independent and identically distributed.

The method of Abraham Wald [72] is about gathering observations until a reliable decision can be made rather than a fixed number of n. The SPRT method uses the cumulative likelihood ratio as a function of observations.

$$\Lambda_n = \prod_k^i \frac{p_1(O_i)}{p_0(O_i)}, n = 1, 2, ... \tag{4.1.8a}$$

The log-likelihood can also be used to define the cumulative log-likelihood ratio:

$$ln(\Lambda_n) = \sum_k^i ln(\frac{p_1(O_i)}{p_0(O_i)}), n = 1, 2, ... \tag{4.1.9a}$$

If the probabilities of error type false negative and false positive of the detection method should be respectively lower than a fixed P_{FN} (probability of false-negative) and P_{FP} (probability of false positive). The SPRT helps to conclude which hypothesis is correct with the smallest number of observations needed.

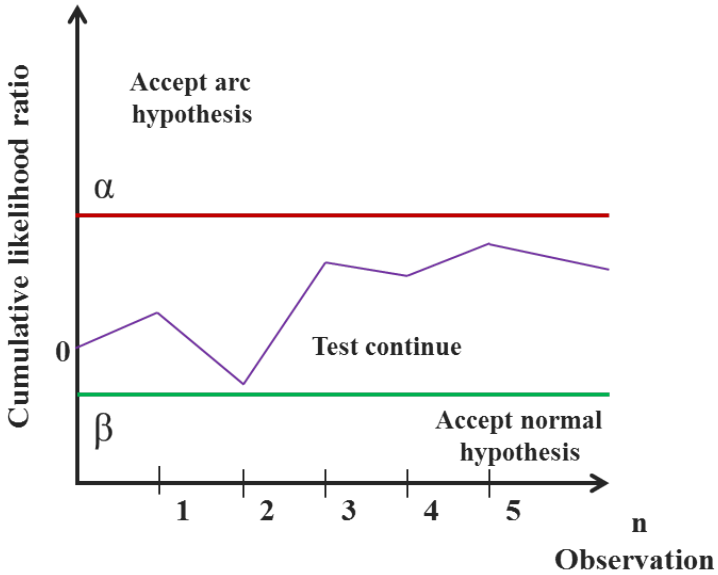

Figure 4.1: Sequential probability ratio test

Figure 4.1 describes the procedure of SPRT. For any observation (n), there are two basic stopping conditions of test: the threshold α and β:

$$\begin{cases} \Lambda_n \geq \alpha : \text{Accept the hypothesis } H_1 \text{ - with arc} \\ \Lambda_n \leq \beta : \text{ Accept the hypothesis } H_0 \text{ - normal} \\ \text{Otherwise : take another observation} \end{cases}$$

Threshold α and β statify the desired precision if they are calculated as following:

- When the hypothesis H_1 is correct

$$1 - P_{FN} = P_{TP} = \int_{O_1}^{O_n} p_1(O)dO = \int_{O_1}^{O_n} \Lambda_n p_0(O)dO \geq \alpha \int_{O_1}^{O_n} p_0(O)dO = \alpha P_{FP} \tag{4.1.10a}$$

$$\alpha = \frac{1 - P_{FN}}{P_{FP}} \tag{4.1.10b}$$

- When the hypothesis H_0 is correct

$$1 - P_{FP} = P_{TN} = \int_{O_1}^{O_n} p_0(O)dO = \int_{O_1}^{O_n} \frac{1}{\Lambda_n} p_1(O)dO \geq \frac{1}{\beta} \int_{O_1}^{O_n} p_1(O)dO = \frac{1}{\beta} P_{FN} \tag{4.1.11a}$$

$$\beta = \frac{P_{FN}}{1 - P_{FP}} \tag{4.1.11b}$$

The expected time of SPRT or the number of observations needed is a random variable which can be estimated. The expectation of stopping time (E_0 - hypothesis normal, E_1 - hypothesis with arc) with respect to $p_i, i = 0, 1$ are as following [72]:

$$E_0 \approx \frac{P_{FP} ln \frac{P_{FP}}{1-P_{FN}} + (1 - P_{FP}) ln \frac{1-P_{FP}}{P_{FN}}}{D(p_0||p_1)} \tag{4.1.12a}$$

$$E_1 \approx \frac{P_{FN} ln \frac{P_{FN}}{1-P_{FP}} + (1 - P_{FN}) ln \frac{1-P_{FN}}{P_{FP}}}{D(p_1||p_0)} \tag{4.1.12b}$$

Where $D(p_0||p_1)$ and $D(p_1||p_0)$ are the Kullback-Leibler divergences between p_0 and p_1 :

$$D(p_0||p_1) = \sum_i p_0(i) ln(\frac{p_0(i)}{p_1(i)}) \tag{4.1.13a}$$

$$D(p_1||p_0) = \sum_i p_1(i) ln(\frac{p_1(i)}{p_0(i)}) \tag{4.1.13b}$$

With i = 0,1 (normal and arcing).

We are only interested in cases where P_{FN} and $P_{FP} < 0.5$ so decreasing P_{FN} or P_{FP} increases the stopping time as we can see on the equations 4.1.12a and 4.1.12b.

4.1.3 Decision with Truncated SPRT

The problem when using directly SPRT for arc fault detection is that the stopping time is not fixed. Therefore, the decision method may not meet the time requirement in extreme cases. To avoid this, we add an upper limit to the number of observations, which is a function of the RMS value of the current.

Figure 4.2 presents our decision method based on SPRT. Let call the output of a given classifier C for arc fault detection as an abnormal event indicator (AEI). With $0 \leq AEI \leq 1$, the more AEI close to 1 the higher the probability of an arc fault occurring and vice versa. Let AEI_threshold is the lowest value of AEI which can

lead to an arcing event. This value can be found by analyses the used data. The calculated upper-bound α and lower-bound β depend on the desired accuracy.

At first, we compute the AEI, if AEI higher than the AEI_threshold the sequential test will begin. Then the likelihood ratio Λ_n will be considered, if Λ_n higher or lower than the defined thresholds α, β the test will stop. We obtain the corresponding result as an arc fault event or wait for another test.

if $\beta < \Lambda_n < \alpha$ we are unavailable to conclude the test . We should consider more observation if the time constraint permit. To verify the time constraint we compute the current RMS score (noted CRMS) the detailed formula for CRMS will be described later. If CRMS lower than calculated threshold (t_CRMS) we continue the test. Otherwise, we should conclude the test immediately (truncated test)with the threshold equal to mean value of α and β(if we don't prioritize hypothesis normal nor with arc fault).

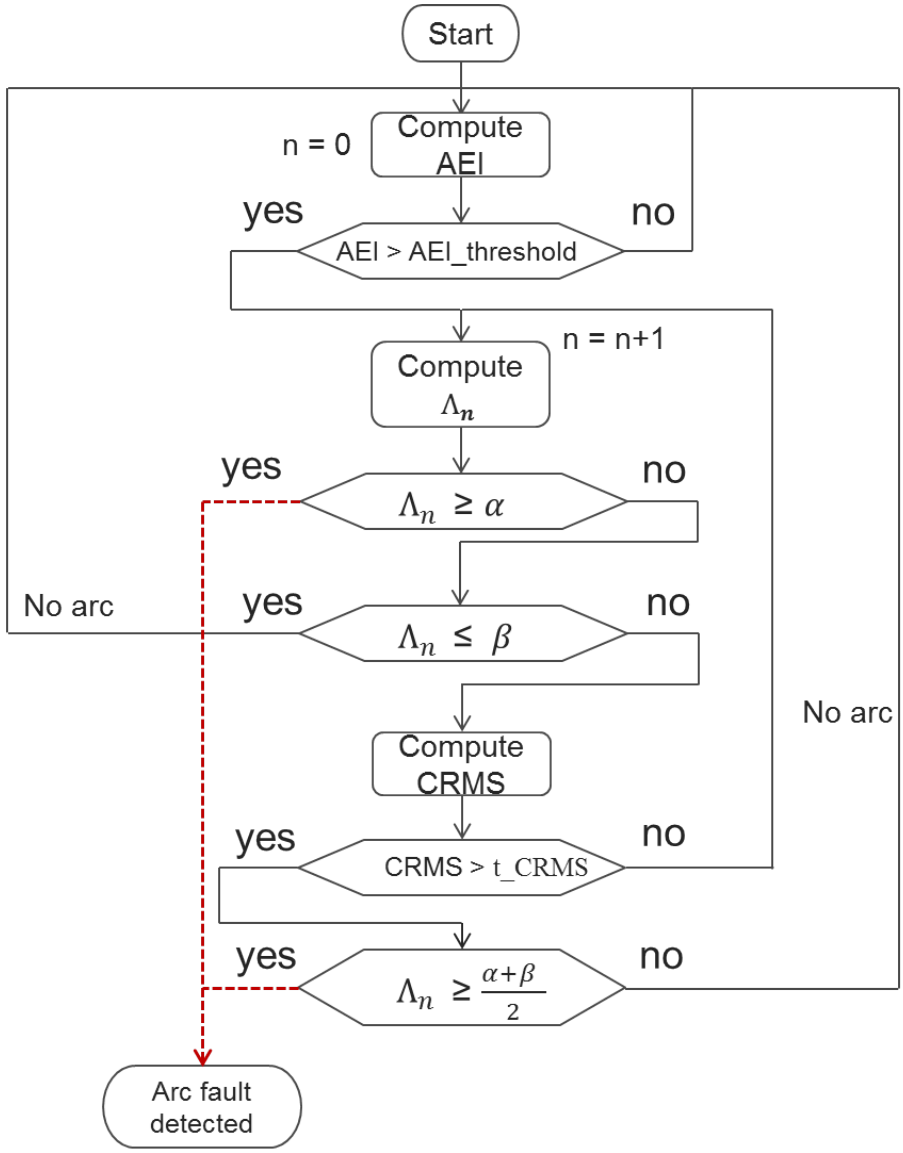

Figure 4.2: Sequential probability ratio test for arc fault detection

To verify the time constraint we calculate the CRMS. Taken into account that the current RMS may vary when arcing occurs (plugging new load when an arc fault is occurring or transient load) which may dramatically change the required response time. We use the following formula to compute the score.

$$CRMS = \sum_{i=1}^{n} RMS_i.t(RMS_i).k(RMS_i) \tag{4.1.14a}$$

With RMS_i is the value RMS of current for a given observation window i. n is the number of observation has been used. t is the maximum break time which

is based on the standard IEC 62606. k is a correction coefficient which will be determined later.

The value exact of t can be calculated which linear interpolation (Figure 4.3) from Table 1.1.

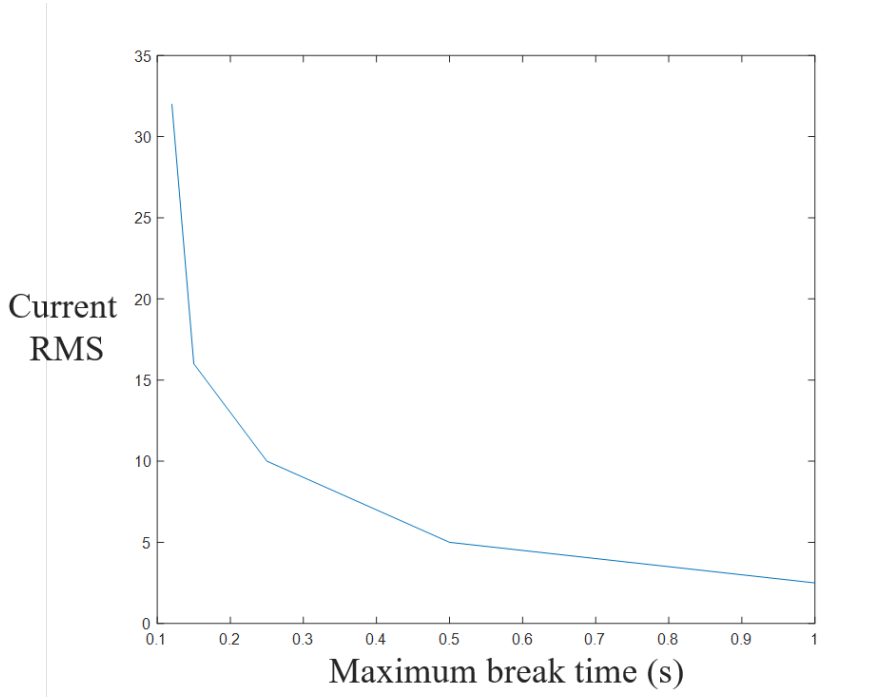

Figure 4.3: Maximum break time with linear interpolation.

As mentioned in the standard IEC 62606 at 2.5 A we have maximum 1 second to detect arc (Table 1.1). Let take $k = 1$ in this case so we obtain the threshold t_CRMS= 2.5*1*1 = 2.5. This value of k and t_CRMS satisfy the condition of maximum break time on the standard till the rms lower or equal to 10A (Table 4.1).

Current (RMS)	2.5A	5A	10A	16A	32A
Calculated response time	$\frac{2.5}{2.5}$=1s	$\frac{2.5}{5}$=0.5s	$\frac{2.5}{10}$=0.25s	$\frac{2.5}{16}$=0.15625s	$\frac{2.5}{32}$=0.078125s

Table 4.1: Calculated detection time for series arc.

More specific, 0.5s maximum break time for 5A and 0.25s maximum break time in the case of 10A.

In the case of 16A the time constraint will be exceeded, we have 0.15625s maximum break time instead of 0.15s. For the case of 32A, we have 0.078125s maximum

break time instead of 0.12s. To correct maximum breaking time errors, a simple linear adjustment based on lowest upper bound (LUP)of current RMS can be used.

$$k(RMS_i) = \frac{t_CRMS}{LUP(RMS_i).t(LUP(RMS_i))} \tag{4.1.15a}$$

With $LUP(RMS_i)$ equal to(2.5, 5, 10, 16 and 32), Table 4.2 shows the calculated coefficient k.

Current RMS (A)	2.5 to 10	10 to 16	16 to 32
k	1	1.042	0.652

Table 4.2: Normalization coefficient

Figure 4.4 shows an example of CRMS calculation, we can see that CRMS increases faster in the case of high RMS current (green step) and slower for lower RMS current (purple step). In the left case, the maximum break time is shorter than the right case knowing that the current level when we start the truncated SPRT is the same. The fixed threshold t_CMRS guarantees the time constraint is respected even in the case of current varying during a test.

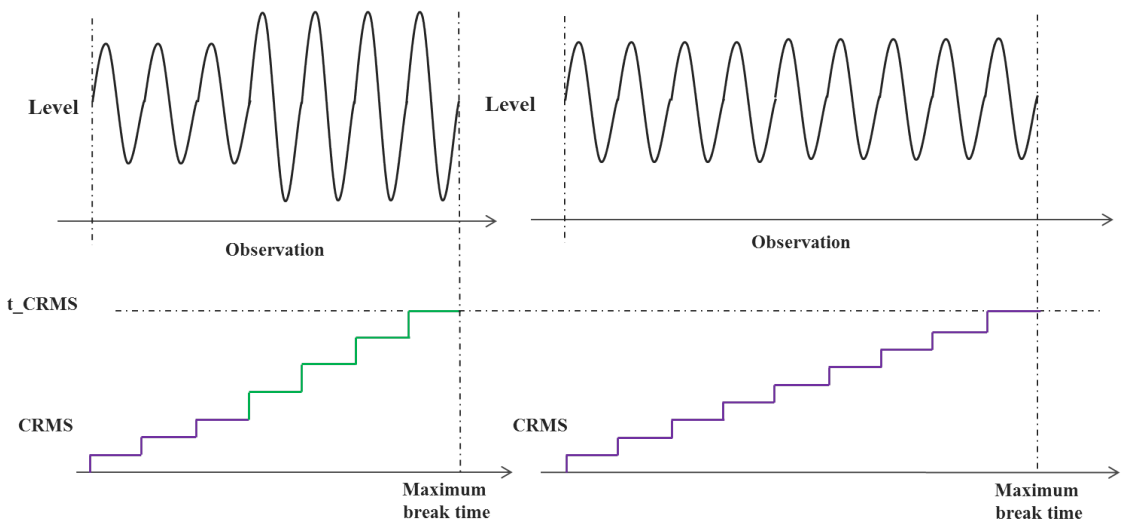

Figure 4.4: CRMS evaluation.

4.1.4 Experimental results

To verify the performance of the proposed decision method, let's take a detection algorithm with classifier C(FFT 1-10Khz and CZT 10-20 Khz) which has been

presented in the Chapter 2. The size of the window was fixed to one period 20 ms. Classifier C has 10 FN and 21 FP samples so

$$TPR = \frac{3405 - 10}{3405} = 0.9971$$

and

$$FNR = \frac{21}{12826} = 0.0016.$$

Without statics decision method we can expect an unwanted trip every $\frac{0.02}{0.0016}$=12.5s. With 4 successive abnormal events counting method for one year we may expect:

$$NFA = \frac{0.0016^4}{0.02.4}.3600.24.365 = 0.0026$$

$$DR = 0.9971^4 = 0.9885$$

That means almost no unwanted trip per year and a detection rate of 0.9885 with 80 ms response time.

With the proposed truncated sprt, if we define the same robustness again unwanted trip as the previous statistic method. We should achieve a

$$P_{FP} = \frac{0.0026}{3600.24.365.50} \approx 1.65.10^{-12}$$

Let say we want to have a DR equal to 0.999 which equivalent to $P_{FN} = 0.001$. To simplify the writing we can use $\ln(\Lambda_n)$ instead of Λ_n for the cumulative score. These following upper and lower bound should be respected (Equations 4.1.10b, 4.1.11b):

$$\alpha = ln(\frac{1 - 0.001}{1.64.10^{-12}}) = 27.12$$
$$\beta = ln(\frac{0.001}{1 - 1.64.10^{-12}}) = -6.9078$$

The Kullback-Leibler divergences between p_0 and p_1 are as following:

$$D(p_0||p_1) = 0.9984.ln(\frac{0.9984}{0.0029}) + 0.0016.ln(\frac{0.0016}{0.9971}) = 5.8218$$

$$D(p_1||p_0) = 0.0029.ln(\frac{0.0029}{0.9984}) + 0.9971.ln(\frac{0.9971}{0.0016}) = 6.3992$$

The expectation of stopping time E_0 and E_1 are:

$$E_0 \approx \frac{1.65.10^{-12}.ln(\frac{1.65.10^{-12}}{1-0.001}) + (1 - 1.65.10^{-}12).ln(\frac{1-1.65.10^{-12}}{0.001})}{5.8218} \approx 1.2$$

$$E_1 \approx \frac{0.001.ln(\frac{0.001}{1-1.65.10^{-12}}) + (1 - 0.001).ln(\frac{1-0.001}{1.65.10^{-12}})}{6.3992} \approx 4.23$$

So we can expect a response time for arc fault situation around 5 periods (100 ms) which are lower than the standard requirement. For the normal situation, we can expect a stopping time around 2 periods (40 ms).

An example of the calculation with the real signal is shown in Figure 4.5.

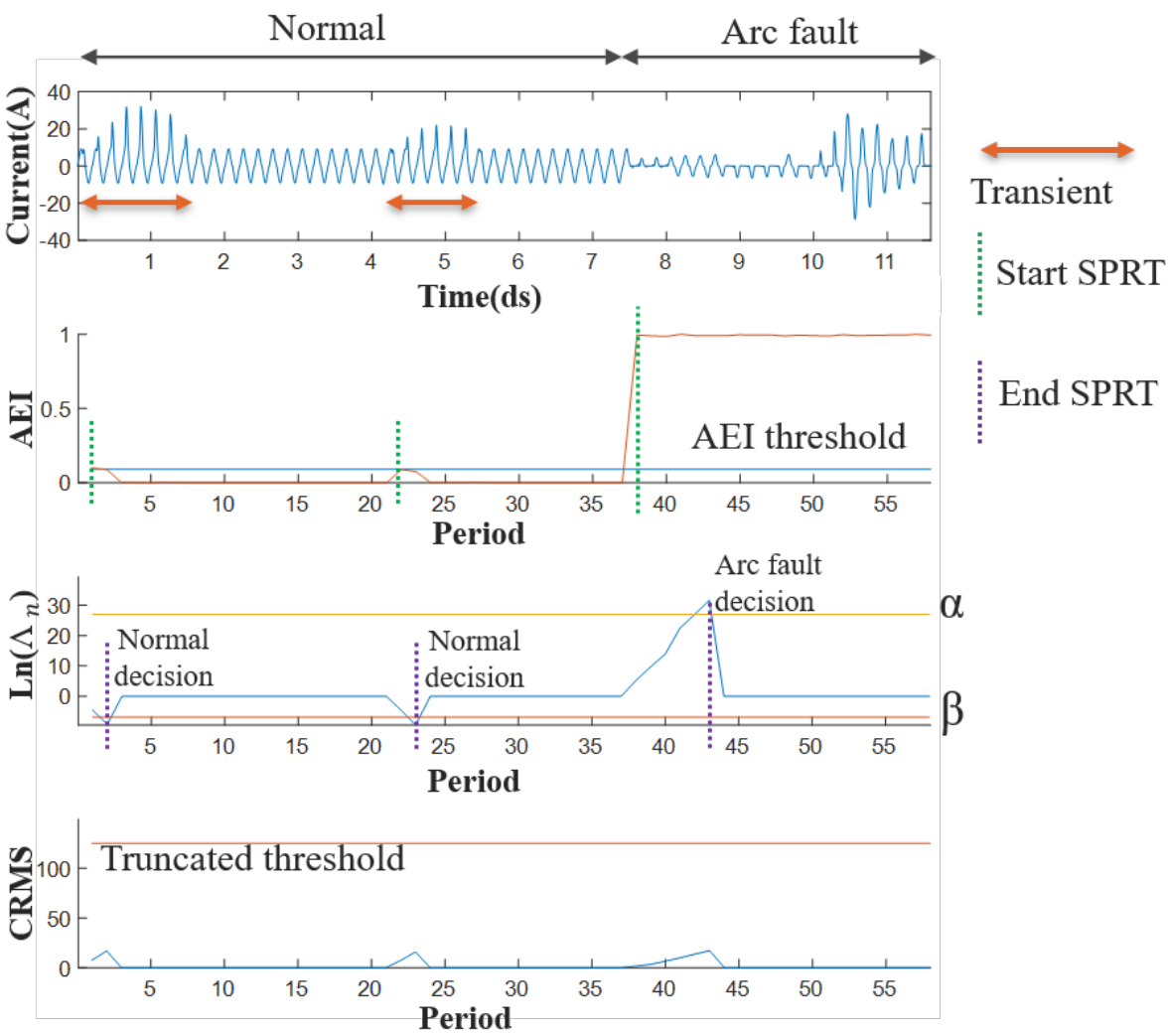

Figure 4.5: Example of sprt decision

As the classifier is not perfect, higher than zero results can be found at the output

when the electrical network is on the transient state. The SPRT starts because the output value of classifier is higher than the predefined threshold which is equal to 0.1 in this experiment (lowest AEI which can lead to an arc fault event). The green dash line indicates that the SPRT is starting and the purple dash line signifies the end of SPRT. The test normally stops when the threshold α or β is reached, and truncated stop when the CRMS goes higher than defined threshold t_CRMS (equal to 2.5*(1/0.02) = 125 in this case). The proposed decision method quickly eliminates the risk of an unwanted trip by correctly classify the current signal (within 2 periods, 1 to 3 and 22 to 24). When arc fault appears, SPRT needs 5 periods to detect the default (39 to 44) which is consistent with the expected stop time calculated above. In both cases, CRMS is very far from the truncated threshold because the used classifier has more than 99% in terms of accuracy. The proposed decision test is more efficient than a simple 4 successive event-based. Both have the same NFA but SPRT based has better decision rate 0.999 with 100 ms response time compare to 0.9885 with 80 ms response time. The results can be explained by the fact SPRT based method uses probability information (value between 0 and 1) rather than binary value (0 or 1) of event successive approach. In addition, the proposed method can use more information (multiple observation windows) in difficult case compare to fixed analysis time approach.

4.2 Long short time memory arc fault detection

As we can see in the previous section, using plural observation windows may increase the detection performance. The mentioned statics methods only exploit the result of the classifier part. In order to go further, we need to use directly the arc fault feature of many observation windows. Which leads to a length variable classification problem. A powerful tool to resolve this problem is recurrent neural network (RNN) as we can find in the literature. In practice, basic RNNs are hard to train and they seem not good at learning long term dependencies. Long short time memory network (LSTM) is a special type of RNN that can learn long-term dependencies with ease. In this section, a time variable detection algorithm based on Long short time memory

neural network will be presented.

4.2.1 Recurrent neural network

RNN is a class of ANN which can work with temporal data thank to its internal connections. In feedforward ANN, the state of the network is lost after a given sample being processed. Therefore it's not optimal for process sequences of data related in time. RNN can pass the current state of the network to future processing. Theoretically, RNN can use all information of previous inputs and outputs which should be more adapted to arc fault detection application than regular ANN with a fixed observation window.

RNN can be describe with the following formula:

$$h_t = \sigma(W_{ih} I_t + W_r h_{t-1}) \tag{4.2.16a}$$

$$O_t = \sigma_o(W_{ho} h_t) \tag{4.2.16b}$$

Where σ is the activation function of hidden layer. σ_o is the activation function of output layer. W_{ih} denotes the weight of network from input layer to hidden layer. I_t is the input of network at a given time t. W_{ih} is the weight of recurrent, h_t is the value of hidden nodes at the given time t. O_t is the output value at t, W_{ho} denotes the weight of network from hidden to output layer.

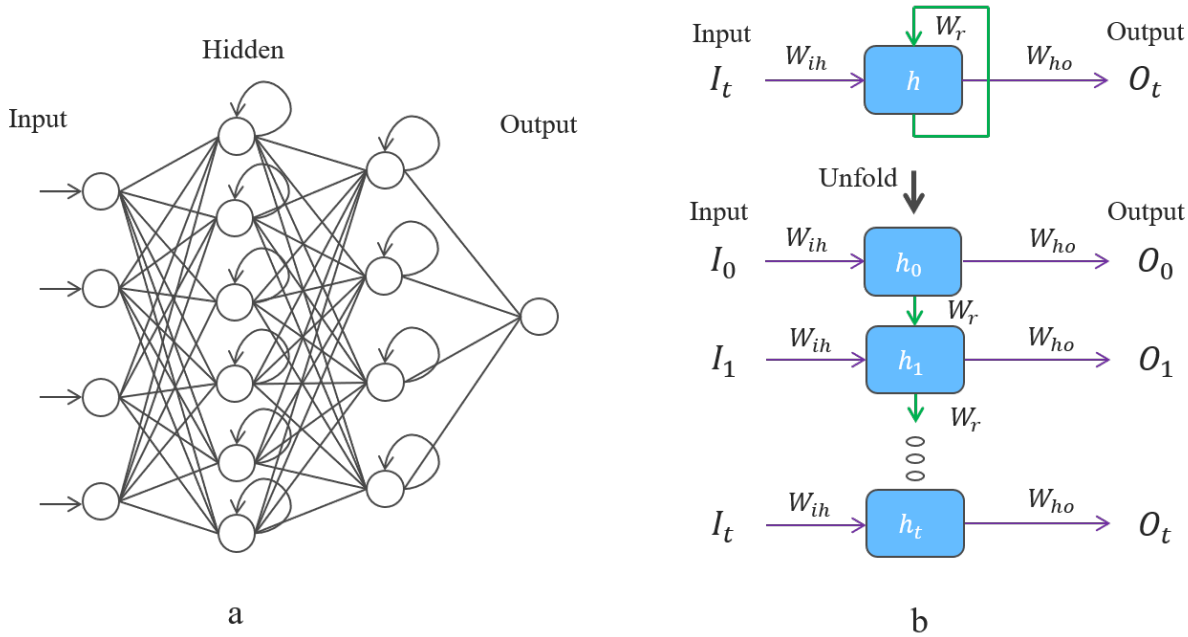

Figure 4.6: Recurrent neural network, a- Example of network; b-Unfold architecture

The method to train RNN is called back-propagation through time (BPTT). BPTT shares the same concept with back-propagation (BP). BP computes the needed gradient to adjust the weights of a feed-forward neural network [98].

It calculates the gradient of the loss function and distributes back output errors through the network layers.

For example, a feed-forward neural network has parameters θ. More specific, w_{ij}^k is the weight between node j of layer k and node i of layer $k-1$, b_i^{k-1} is the bias of node i. Let consider a data sample with input $\vec{x_m}$ and desired output $\vec{y_m}$ for the given input. The set of n sample can be noted as: $X = \{(\vec{x_1}, \vec{y_1}), ..., (\vec{x_n}, \vec{y_n})\}$. A cost function $J(X, \theta)$ defines the error between desired output $\vec{y_m}$ and the calculated output $\vec{yc_m}$ with the network of parameter θ from input $\vec{x_m}$. To train the network we should update the weight w_{ij}^k and biases b_i^{k-1} according to the gradient of the cost function $J(X, \theta)$. Each iteration we can adjust the network's parameters with the following formula:

$$\theta^{t+1} = \theta^t - r\frac{\partial J(X, \theta^t)}{\partial \theta} \tag{4.2.17a}$$

Where r is the learning rate , θ^t denotes the parameters of the network at iteration t.

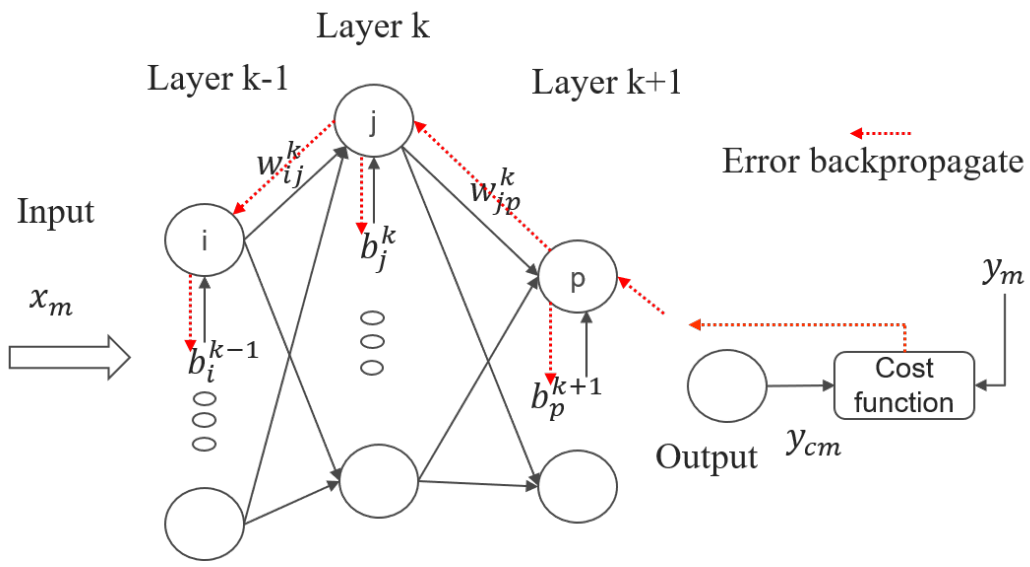

Figure 4.7: Backpropagation

BPTT can be understood as BP on an unfold RNN with the entire input sequence I_0 to I_t as the input layer. Output sequence O_0 to O_t as output of the unfold RNN and D_0 to D_t as desired output. They are all required for the training process. As the parameters of the network are shared across slices, each parameter is calculated at a time slice and then being averaged.

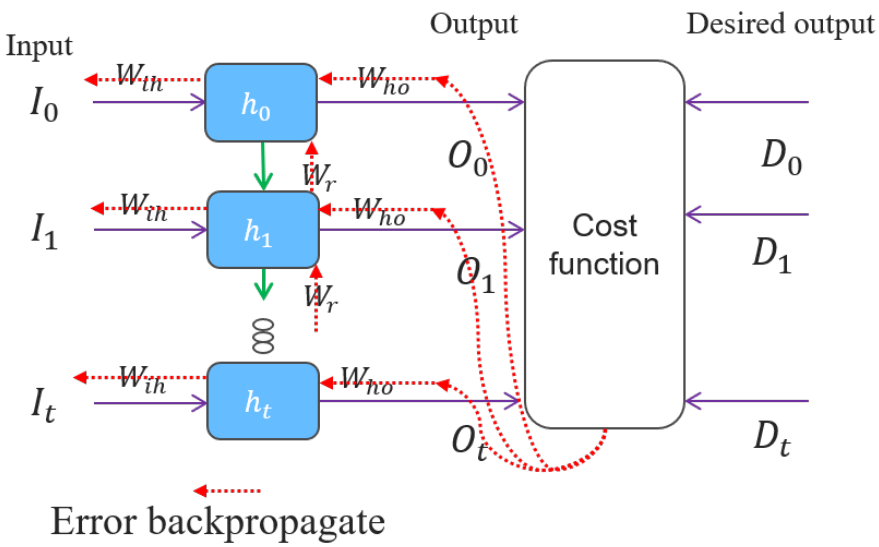

Figure 4.8: Backpropagation through time

However, learning with RNN is a difficult problem [99].

The gradients may vanish or explode when backpropagating error across many time steps [100].

Let's consider an I_{t_1} input pass through the network at a given time t_1 and error computed at time t_2. The contribution of input I_{t_1} at time t_2 will either tend to zero or grows very high exponentially as $t_2 - t_1$ increases. Therefore the gradient of error respected to the given input will vanish or explode. Truncated backpropagation through time (TBPTT) has been presented as a solution to limit the mentioned problem. TBPTT sets a maximum number of time steps that error can be backpropagated which means the ability to learn long-range dependencies is being sacrificed.

4.2.2 Long short term memory neural network

The most used RNN architectures for long sequence learning is long short term memory (LSTM). It was introduced by Hochreiter and Schmidhuber in 1997 [101] in order to overcome the problem of training RNN.

The main difference between basic RNN and LSTM networks is the LSTM cell. Ordinary nodes in hidden layers of RNN are replaced by LSTM cells. LSTM helped to pass the gradient across time steps without being vanished or exploded. A LSTM cell is composed of an input gate(i), an internal state(m), a forget gate(f) and an output gate(o)(Figure 4.9).

- Input gate takes information from the current data point I_t, and either the hidden layer h_{t-1} or internal state at previous time step m_{t-1}.
- The internal state is the memory of the LSTM cell, the internal state component has a recurrent connection with fixed unit weight. This constant weight helps the error to flow across many time steps without being vanished or exploded. It is frequently called the constant error carousel.
- Forget gate provides a method that helps the network to flush the information in the internal state.
- Output gate computes the output of LSTM cell from I_t , h_{t-1},m_{t-1}.

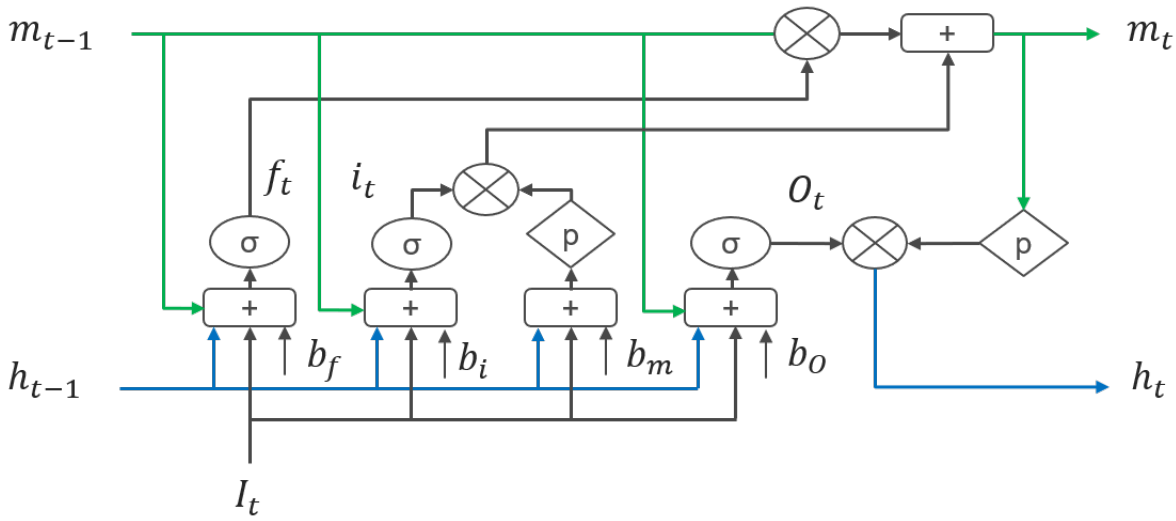

Figure 4.9: Long short term memory cell architecture

The relation between different elements of the network are presented with the following equations:

$$i_t = \sigma(W_{Ii}I_t + W_{hi}h_{t-1} + W_{mi} \odot m_{t-1} + b_i) \tag{4.2.18a}$$

$$f_t = \sigma(W_{If}I_t + W_{hf}h_{t-1} + W_{mf} \odot m_{t-1} + b_f) \tag{4.2.18b}$$

$$O_t = \sigma(W_{IO}I_t + W_{hO}h_{t-1} + W_{mO} \odot m_{t-1} + b_O) \tag{4.2.18c}$$

$$m_t = f_t m_{t-1} + i_t p(W_{Im}I_t + W_{hm}h_{t-1} + b_m) \tag{4.2.18d}$$

$$h_t = O_t p(m_t) \tag{4.2.18e}$$

Where I_t is the input vector at the given time t. i_t ,f_t ,O_t,m_t, h_t are the input gate, forget gate, output gate, internal state and hidden layer vector. W_{xx} denote weight matrices between different component of the network (for example W_{Ii} is the weight matrix between input vector and input gate etc.). σ is the action function (sigmoid function is frequently used). p is a hyperbolic function (tanh function for example). $\odot$ represents hadamard product.

4.2.3 Experimentation results

To adapt the LSTM for arc fault detection at first we need to define the input sequence data. The size of the observation window has been fixed at 20 ms (similar with previous researches). We decided to use the input feature from FFT1-10 kHz and CZT 10-20 kHz in order to make a comparison with the statistic decision method. Each observation gives 19 features (see Paragraph 2.4.2.2 and Table 2.1 for the details about features), the length of sequence depend on the value RMS of current (following the equation CRMS). We used 819 input sequences for this experiment. The size of the sequences varies between 10- 45 observation windows. The ratio of the arcing sequence and normal sequence is around 0.3/ 0.7. Each sequence has only a respected label of normal and arc fault situations.

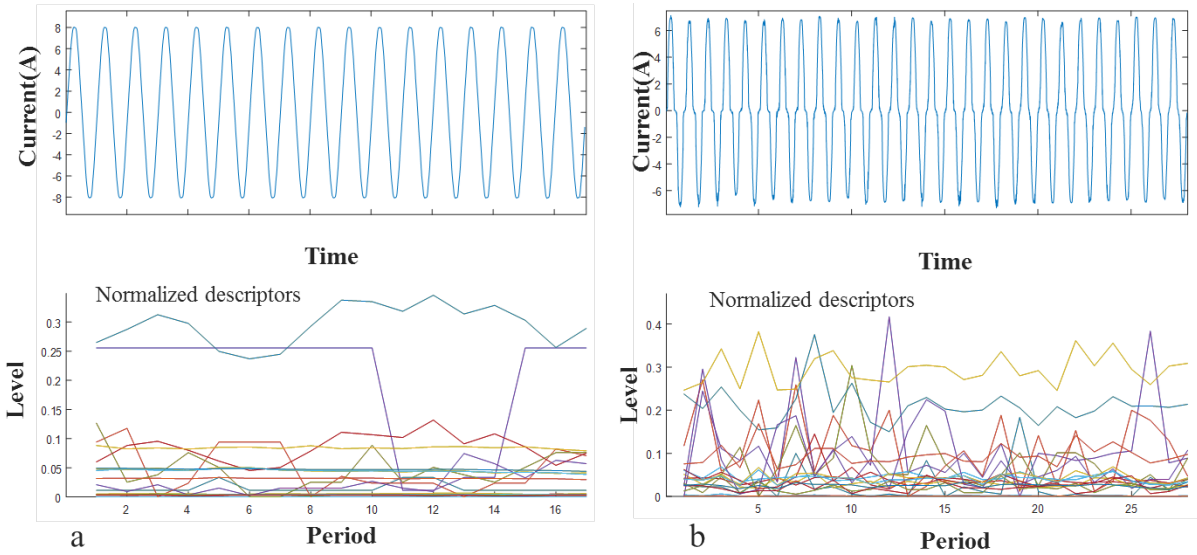

Figure 4.10: Example of input sequence in normal functioning and with arc

In the Figure 4.10-a (normal functioning), we can see the value RMS of current signal is around 5.66. According to the standard, for this RMS value, the algorithm should response within 350 ms if an arc fault occurs. The length of the sequence is 17 observations windows which satisfy the condition. Similarly, in the case of signal with arc (Figure 4.10)-b, the length of the input sequence (27 observation windows) is fixed according to the standard IEC62606 with interpolation method. The different features of input signals are normalized between 0 and 1. We can see

less variation of features in the case of normal functioning compare to with arc.

Secondly, we need to define the neural network for the detection task. According to the number of input features, the complexity of the problem and to speed up the training process, we mainly focused on the simple network which has up to three layers of LSTM cells. Also, as we saw in Chapter 3 complicated network tends to provide lower detection performance due to the difficulties in the learning process. The following architecture of network has been used for the experimentation (Figure 4.11):

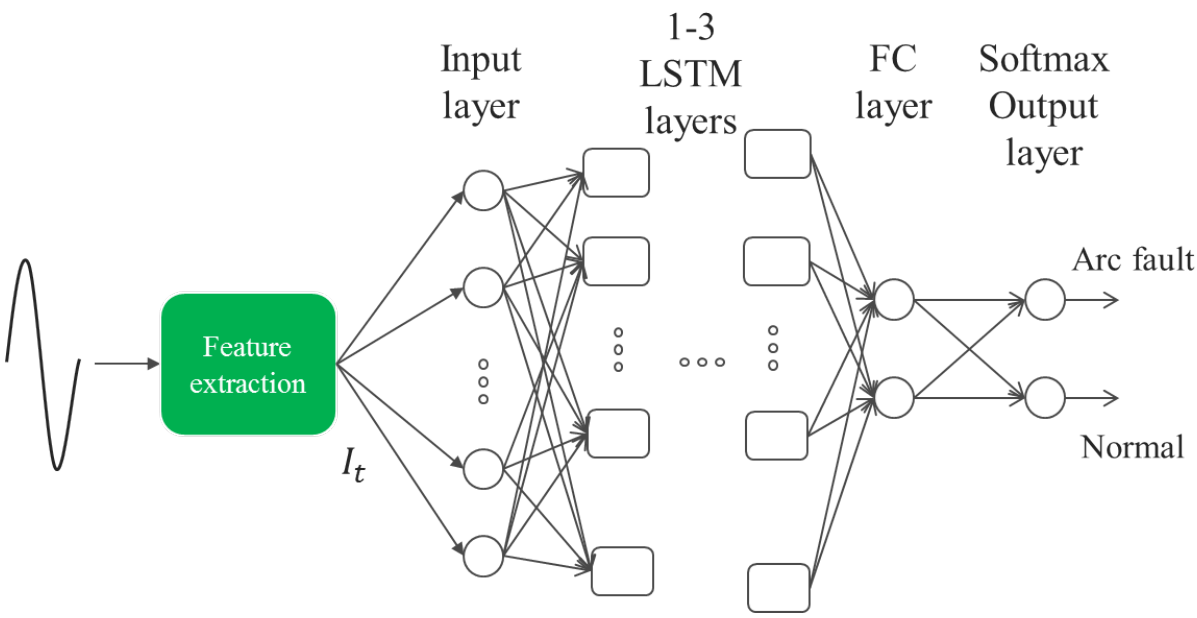

Figure 4.11: Neural network for arc fault detection with LSTM cells

Every tested network has one sequence input layer which consists of 19 nodes and 1 to 3 LSTM layers with a variable number of hidden nodes. They all have a fully connected layer and a softmax- classification layer. The size of the last layer is fixed to 2 because only two classes are needed: normal and arc fault.

These networks were validated and trained using 5-Fold Cross-Validation (5 FCV). This method consists of separating the database into 5 groups randomly and performing the actions. After dividing into groups (folds) we used the first four folds as the training set and the fifth as the validation set. The experiments performed with this configuration five times by changing the folds positions. The

averaged validation results from 5FCV are show at the Table 4.3.

Configuration	Validation accuracy	Configuration	Validation accuracy
1 Layer LSTM(5 cells)	97.5%	2 Layers LSTM(24-8 cells)	99.3%
1 Layer LSTM(8 cells)	98.2%	2 Layers LSTM(19-24 cells)	98.3%
1 Layer LSTM(10 cells)	98.6%	2 Layers LSTM(8-24 cells)	97%
1 Layer LSTM(12 cells)	98.6%	2 Layers LSTM(5-10 cells)	96%
1 Layer LSTM(24 cells)	99.2%	2 Layers LSTM(8-2 cells)	95.2%
1 Layer LSTM(36 cells)	99.4%	2 Layers LSTM(10-10 cells)	98.2%
1 Layer LSTM(50 cells)	99.6%	2 Layers LSTM(12-10 cells)	98.2%
1 Layer LSTM(100 cells)	99.6%	2 Layers LSTM(20-10 cells)	99.6%
2 Layers LSTM(8-5 cells)	97.5%	2 Layers LSTM(40-10 cells)	99.6%
2 Layers LSTM(10-5 cells)	97.9%	3 Layers LSTM(8-8-8 cells)	98.2%
2 Layers LSTM(24-5 cells)	99.2%	3 Layers LSTM(8-8-16 cells)	98.3%
2 Layers LSTM(8-8 cells)	97.8%	3 Layers LSTM(24-48-16 cells)	98.5%

Table 4.3: Series arc fault detection with LSTM

With one layer of LSTM, the network's performance scales with the number of using cells (97.5% to 99.6%). However, when the number of used cells is higher than 50 there is no improvement by adding more cells. The networks with two or three layers or LSTM share the same trend, the more cells using the better result is. There is no network that can achieve a classification rate higher than 99.6%. The LSTMnet_19_50 (1 Layer LSTM (50 cells)) and LSTMnet_19_50 (1 Layer LSTM (50 cells)) converge faster than LSTMnet_19_20_10 (2 Layers LSTM (20-10 cells)) LSTMnet_19_8_5 (2 Layers LSTM (10-5 cells)) (Figure 4.12). It can be explained by the fact that deeper networks are more difficult to train. In every case, the training and validation accuracy is close to each other (lower than 0.4% difference) which means LSTM based network has good generalization for the arc fault detection task.

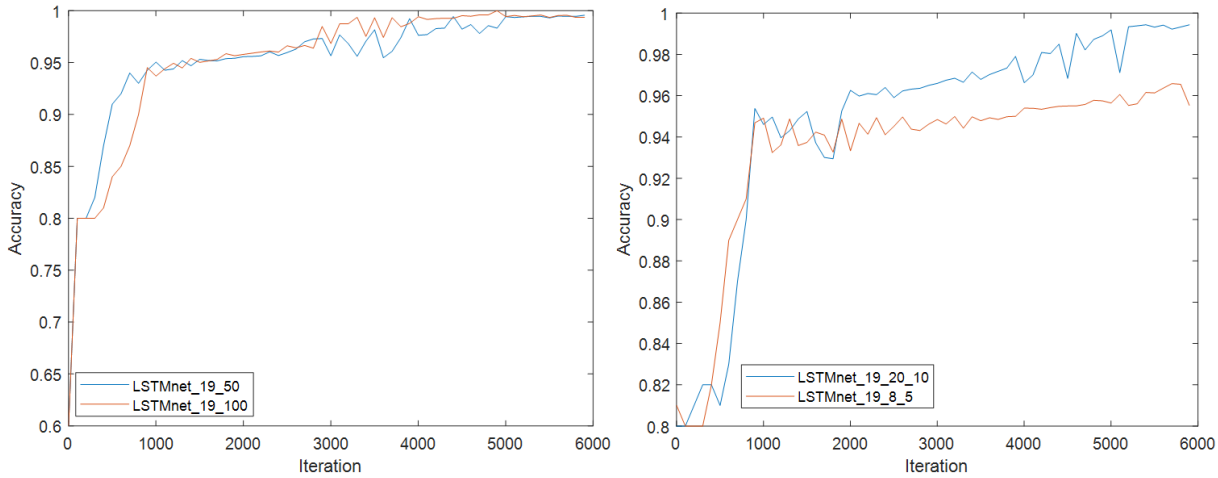

Figure 4.12: LSTMs trainning accuracy

The masking configuration (high energy consumption&noisy appliances and arcing with low energy consumption appliances) is problematical for LSTM based network. Many samples of this category have been misclassified. As we know the detection time should be calculated with the arcing current. In this considered configuration, we can't determinate correctly the detection time because we have only information of the line current. The difference between detection time based on line current and detection time based arcing current causes this problem (the input sequence may be shorter than it should be). For experimentation purposes, we can work around this problem by changing the size of the input current series according to the arcing current, however, this solution doesn't work in practice because we need to know exactly where is the arc fault before detecting it. To properly address this problem we train the LSTM network with a short sequence (80ms) and make the final classifier based on multiple outputs of the network. With this solution, most difficult samples can also be correctly classified. However more data with masking configuration is needed to confirm the efficiency proposed solution.

4.3 Conclusion

In this chapter, two arc fault detection methods which take into account the variable time aspect or arc fault detection have been presented. The SPRT based method adapts statistical analysis with classifier. This method permits to optimize the final

behavior (tripping or not) of the detection system correspond to different strategies (sensitivity or robustness). The second method based on LSTM uses directly the variable time series as input. The results of the LSTM network are very promising, and there are still some points that worth to be investigated further.

Chapter 5

Conclusions

The results in this thesis show that we can make efficient series arc fault detection algorithm for the household electrical network by using machine learning techniques. The first contribution of this research consists of creating a method that helps to find optimize arc fault features and make a detection algorithm from multiples features. The proposed method works very well with commonly used features that can be found in the literature and it increases significantly the detection performance. Worth to be mentioned that training time and amount of data needed for this method is very affordable. We can also easily add resources constraints if the final detection algorithm needs to be implemented into a low-cost platform.

The second contribution of this thesis shows that the deep learning algorithm can be used for arc fault detection. This kind of algorithm eliminates the need for finding optimal arc fault features, which is a time consuming and required field knowledge task. In exchange for this convenient, a performance drop can be noticed compared to algorithms based on highly optimized input features. This can be explained by the fact that the learning process is never an easy task for the complicated network. We may achieve better performance if we invested more time and effort in developing a good database and investigate further on architecture and training technics. However, there are many efficient arc fault features that have been found in the literature. Therefore deep learning-based solutions may require more effort than the first proposed method. In addition, a major issue with this kind of algorithm is the processing power required. It costs much more expensive than the first method in

the both term: training and implementation. Of course, it may work with a cloud-based system or so on but the cost of this system can be excessive and many more parameters should be considered such as: the connection, continuation of service, security problem.

We also investigated in the time aspect of arc fault detection and two methods have been presented. The SPRT based method is potent when using a high accuracy classifier, it helps to optimize the tripping strategy. Also, this method is simple to implement on an embedded platform because it has very low computation cost(several operations per observation window). The LSTM based method shows very good performance, only slightly lower than the multi-criteria based method. However, masking load configurations are troublesome for the LSTM method because of the lack of information about arcing current. For all mentioned methods a large database with complicated situations has been used to confirm their performances.

5.1 Prototype Implementation

With the most efficient solution(Table 2.4)for every 20 milliseconds(one period), we need to calculate Wavelet transform, Fourier transform(FFT) and Chirp Zeta transform(CZT) of raw current signal. And then construct arc fault features with descriptors, finally classify with an artificial neural network. The most costly parts of the proposed solution are: Wavelet transform, CZT and FFT which have Θ(n log n) complexity - n is the length of input series. So if we use the low-frequency input signal, for example, n = 1024, we need about several tens of thousands of multiply-accumulate operations(MACs) for the whole processing and classify part. Let's say 40000 MACs per 20 ms is needed which is equivalent to 2 MFLOPs. We have tested and implemented an important part of the final detection algorithm(FFT and CZT) with SPRT decision into an embedded platform(the core is a Zynq-7010 SoC: Dual-core cortex A9, 28K logic programmable cells and 17.600 LUTs). With this platform, we have no real-time issues. The CPU part used mainly to compute the classifier with neural-network, descriptor, and SPRT. On the FPGA part, we implemented an ADC controller, FFT, and CZT. We didn't use a fully paralleled

architecture and DMA is used for transfer pre-processing data to CPU. If necessary, the proposed solution can be optimized further and implemented on a less powerful platform.

5.2 Perspective

The experimentation results of this thesis show that machine learning is well adapted to arc fault detection task, in addition, the developed prototype is working well on different electrical networks and appliances. This suggests the future development of work in these following directions. For CNN based method, multiple classifiers system is worth to be investigated further. The LSTM detection method needs more masking configuration data to confirm its efficiency. It could be very interesting to find a solution that give the LSTM method the ability to predict arcing current based on electrical network mapping(load recognition and localization). This solution permits to optimize the length of input sequences for LSTM method. In general, it's very important to see how these proposed methods work in practice. An environment auto-adapted system(adapting parameters depend on appliances frequently presents on the network) could be a worthwhile subject.

Bibliography

[1] **Hien Duc VU**, Edwin CALDERON, Patrick SCHWEITZER, Serge WEBER, Nicolas BRITSCH, *Multi criteria series arc fault detection based on supervised feature selection*, International Journal of Electrical Power & Energy Systems, Volume 113, pp. 23-34, 2019. **Journal paper**

[2] **Hien Duc VU**, Nicolas BRITSCH, *Convolutional Neural Networks for series arc-fault detection*, 19th Industrial Conference on Data Mining ICDM, Newyork, USA, 2019. **International conference paper**

[3] **Hien Duc VU**, Edwin CALDERON, Patrick SCHWEITZER, Serge WEBER, Nicolas BRITSCH, *AC Series arc fault detection with stacked autoencoders*, 45th Annual Conference of the IEEE Industrial Electronics Society (IES), IECON, Lisbon, Portugal, 2019. **International conference paper**

[4] **Hien Duc VU**, Patrick SCHWEITZER, Serge WEBER, Nicolas BRITSCH, *Optimisation de l'analyse multicritère pour la détection de défauts d'arc électrique*, 14ème Colloque sur les Arcs Electriques, Bourges, France, 2019. **National conference paper**

[5] **Hien Duc VU**, Patrick SCHWEITZER, Serge WEBER, Nicolas BRITSCH, *Impact of appliances and power network's conditions on the performance of arcing fault detection methods*, 13ème Colloque sur les Arcs Electriques, Nancy, France, 2017. **National conference paper**

[6] IEC62606, *General requirements for arc fault detection devices*, International Electrotechnical Commission , First Edition,2013.

[7] Underwriters Laboratories Inc., *UL standard for safety for Arc-Fault Circuit-Interrupters UL1699*, Second Edition,2008.

[8] O. Mayr, *Beitrage zur Theorie des statischen und des dynamischen Lichtbogens*, Archiv fur Elektrotechnik,vol. 37, pp. 588-608,1943.

[9] A. Cassie, *Theorie Nouvelle des Arcs de Rupture et de la Rigidité des Circuits*, Cigre,Report, vol. 102, pp.588-608, 1939.

[10] U. Habedank, *On the Mathematical Description of Arc Behaviour in the Vicinity of Current Zero*, etzArchiv, Vol. 10, No.11, pp. 339-343, 1988.

[11] B. Jieqiu, Z. Yi, D. Zhiqiang and Z. Hongqiang, *Arc fault identification method based on fractal theory and SVM*, 2014 International Conference on Power System Technology, pp. 1182-1187, 2014.

[12] K. Yang, R. Zhang, J. Yang, C. Liu, S. Chen and F. Zhang, *A Novel Arc Fault Detector for Early Detection of Electrical Fires*, Sensors, 2016.

[13] R. Zhang, K. Yang, Q. Wu and J. Yang, *Research on Low-voltage Arc Fault Detection Based on BP Neural Network*, Applied Mechanics and Materials, pp. 499-502, 2014.

[14] L. Yu-Wei, W. Chi-Jui and W. Yi-Chieh, *Detection of serial arc fault on low-voltage indoor power lines by using radial basis function neural network*, Electrical Power and Energy Systems, pp. 149-157, 2016.

[15] S. Jovanovic, A. Chahid, J. Lezama and P. Schweitzer, *Shunt active power filter-based approach for arc fault detection*, Electric Power Systems Research, vol. 141, pp. 11-21, 2016.

[16] C.-J. Wu, Y.-W. Liu and C.-S. Hung, *Intelligent Detection of Serial Arc Fault On Low Voltage Power Lines*, Journal of Marine Science and Technology, vol. 25, pp. 43-53, 2017.

[17] A. T. Renjini Raveeendran, *Series Arc Fault Detection Using Discrete Wavelet Transform*, International Journal of Science and Research, 2014.

[18] J. Lezama, P. Schweitzer, E. Tisserand, J.-B. Humbert, S. Weber and P. Joyeux, *An embedded system for AC series arc detection by inter-period correlations of current*, Electric Power Systems Research, vol. 129, pp. 227-234, 2015.

[19] N. Hadziefendic and Z. Radakovic, *Detection of series arcing in low-voltage electrical installations*, European Transactions on Electrical Power, pp. 423-432, 2009.

[20] C. Restrepo and J. Henson, *Systems, devices, and methods for detecting arcs*, US Patent US7110864B2, 8 3 2004.

[21] G. Artale, A. Cataliotti, V. Cosentino, D. D. Cara, S. Nuccio and G. Tine, *Arc Fault Detection Method Based on CZT Low-Frequency Harmonic Current Analysis*, IEEE Transactions on Instrumentation and Measurement, vol. 66, pp. 888-896, 2017.

[22] S. Zhang, F. Zhang, P. Liu and Z. Han, *Identification of Low Voltage AC Series Arc Faults by using Kalman Filtering Algorithm*, Elektronika ir Elektrotechnika , ISSN, vol. 5, 2014.

[23] H. Guan, B. Wang, Z. Zhao, S. Bimenyimana and Q. Wang, *Arc fault Current's Power Spectrum Characteristics and Diagnosis Based on Welch Algorithm*, International Journal of Engineering Science and Computing, vol. 6, 2016.

[24] T. A.Kawady, N. I.Elkalashy, A. E.Ibrahim and A.-M. I.Taalab, *Arcing Fault Identification using combined Gabor Transform-neural network for transmission lines*, Electrical Power and Energy System, pp. 248 - 258, 2014.

[25] Y. Wang, F. Zhang and S. Zhang, *A New Methodology for Identifying Arc Fault by Sparse Representation and Neural Network*, IEEE Transactions on Instrumentation and Measurement, vol. 67, pp. 2526-2537, 2018.

[26] K. Yang, R. Zhang, S. Chen, F. Zhang, J. Yang and X. Zhang, *Series Arc Fault Detection Algorithm Based on Autoregressive Bispectrum Analysis*, Algorithms, vol. 8, pp. 929-950, 2015.

[27] S. Lesecq and A. Barraud, *Arcing Fault Detection Using Wavelet Transform*, IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, vol. 36, pp. 345-350, 2003.

[28] C. R. Jr., *Arc fault detection using fuzzy logic*, US Patent 8 054 592 B2, 8 11 2011.

[29] Y. Liu, F. Guo, Z. Ren, P. Wang, T. N. Nguyen, JiaZheng and X. Zhang, *Feature Analysis in Time-domain and Fault Diagnosis of Series Arc Fault*, IEEE Holm Conference on Electrical Contacts, pp. 306-311, 2017.

[30] C. E.Restrepo, *Arc Fault Detection and Discrimination Methods*, Electrical Contacts - 2007 Proceedings of the 53rd IEEE Holm Conference on Electrical Contacts, pp. 115-122, 2007.

[31] P. Qi, S. Jovanovic, J. Lezama and P. Schweitzer, *Discrete Wavelet Transform Optimal Parameters Estimation for Arc Fault Detection in Low-voltage Residential Power Networks*, Electrical Power Systems Research, pp. 130-139, 2017.

[32] J. Tang, S. Alelyani and H. Liu, *Feature Selection for Classification: A Review*, Data Classification: Algorithms and Applications, 2014.

[33] G. Isabelle and E. Andre, *An Introduction to Variable and Feature Selection*, Journal of Machine Learning Research, pp. 1157-1182, 2003.

[34] J. C. Ang, A. Mirzal, H. Haron and H. N. A. Hamed, *Supervised, Unsupervised, and Semi-Supervised Feature Selection: A Review on Gene Selection*, IEEE Transactions on Computational Biology and Bioinformatics, vol. 13, pp. 971-989, 2016.

[35] M. M. Christiansen, K. R. Duffy, F. d. P. Calmon and M. Medard, *Brute force searching, the typical set and Guesswork*, IEEE International Symposium on Information Theory, pp. 1257-1261, 2013.

[36] P. Somol, P. Pudi and J. Kittler, *Fast Branch & Bound Algorithms for Optimal Feature Selection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, pp. 900-912, 2004.

[37] A. A. Naeini, M. Babadi, S. M. J. Mirzadeh and S. Amini, *Particle Swarm Optimization for Object-Based Feature Selection of VHSR Satellite Image*, IEEE Geoscience and Remote Sensing Letters, vol. 15, pp. 379 - 383, 2018.

[38] X.-Y. Liu, Y. Liang, S. Wang, Z.-Y. Yang and H.-S. Ye, *A Hybrid Genetic Algorithm with Wrapper-Embedded Approaches for Feature Selection*, IEEE Access, vol. 6, pp. 22863-22874, 2018.

[39] X.-Y. Liu, Y. Liang, S. Wang, Z.-Y. Yang and H.-S. Ye, *Useful AI tools-a review of heuristic search methods*, IEEE Potentials , pp. 51-54, 1991.

[40] L. J.-H. and O. S.-Y., *Feature selection based on geometric distance for high-dimensional*, Electronics Letters, vol. 52, pp. 473-475, 2016.

[41] J. Lezama, P. Schweitzer, S. Weber, E. Tisserand and P. Joyeux, *Arc Fault Detection Based on Temporal Analysis*, IEEE 60th Holm Conference on Electrical Contacts, pp. 1-5, 2014.

[42] L. Yu and H. Liu, *Efficient Feature Selection via Analysis of Relevance and Redundancy*, Journal of Machine Learning Research, pp. 1205-1224, 2004.

[43] Lal T.N., Chapelle O., Weston J., Elisseeff A, *Embedded Methods*, Feature Extraction. Studies in Fuzziness and Soft Computing, vol 207. Springer, 2006.

[44] Jinmi Lezama, Patrick Schweitzer, Serge Weber, Etienne Tisserand, Patrice Joyeux and Michael Rabla, *Frequency Analysis to Arcing Detection and Prototyping FPGA Approach*, Proc. IEEE 59th Holm Conf. Elect. Contacts, pp. 1-6,2013.

[45] Weiyan Zheng,Weilin Wu, *Detecting Low-Voltage Arc Fault Based on Lifting Multiwavelet*, Proceedings of the Asia-Pacific Conference on Computational Intelligence and Industrial Applications, pp. 254-257, 2009.

[46] Dae-won Park, Il-kwon Kim, Su-yeon Choi and Gyung-suk Kil, *Detection Algorithm of Series Arc for Electrical Fire Prediction*, 2008 International Conference on Condition Monitoring and Diagnosis, Beijing, China, 2008.

[47] Il-kwon Kim, Dae-won Park, Su-yeon Choi, Gyung-suk Kil, *Detection and Analysis of Series Arc Discharge in Indoor Wiring Systems*, Proceedings of the 7th WSEAS International Conference on Power Systems, Beijing, China, 2007.

[48] Manaf Atharparvez, Kedar R Purandare, *Series Arc Fault Detection Using Novel Signal Processing Technique*, 2018 IEEE Holm Conference on Electrical Contacts, 2018.

[49] Edwin Calderon-Mendoza, Patrick Schweitzer and Serge Weber, *Kalman filter and a fuzzy logic processor for series arcing fault detection in a home electrical network*, International Journal of Electrical Power & Energy Systems Volume 107, pp. 251-263, 2019.

[50] Etienne Tisserand, Jinmi Lezama, Patrick Schweitzer, Yves Berviller, *Series arcing detection by algebraic derivative of the current*, Electric Power Systems Research 119, pp. 91–99,2015.

[51] Wenjie Liu, Xiaobin Zhang, Yanjun Dong, Xinrong Huang, *Arc Fault Detection for AC SSPC Based on Hilbert-Huang Transform*, IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society, 2017.

[52] Wenjie Liu, Xiaobin Zhang, Yanjun Dong, Xinrong Huang, *A Stacked Autoencoder-Based Deep Neural Network for Achieving Gearbox Fault Diagnosis*, `https://www.hindawi.com/journals/mpe/2018/5105709/`, Mathematical Problems in Engineering, Article ID 5105709, 2018.

[53] Mohendra Roy, Sumon Kumar Bose, Bapi Kar, Pradeep Kumar Gopalakrishnan, *A Stacked Autoencoder Neural Network Based Automated Feature Extraction Method For Anomaly Detection in Online Condition Monitoring*, `https://arxiv.org/ftp/arxiv/papers/1810/1810.08609.pdf`,IEEE-SSCI 2018 Conference,2018.

[54] Qianli Ma, Lifeng Shen, Weibiao Chen, Jiabin Wang, Jia Wei, Zhiwen Yu, *Functional echo state network for time series classification*, Information Sciences, Volume 373, pp.1-20, 2016.

[55] Bianchi FM, Scardapane S, Løkse S, Jenssen R, *Reservoir computing approaches for representation and classification of multivariate time series*, `https://arxiv.org/abs/1803.07870`, 2018.

[56] Aswolinskiy W, Reinhart RF, Steil J, *Time series classification in reservoir- and model-space:A comparison*, Artificial Neural Networks in Pattern Recognition, pp. 197–208, 2016.

[57] Zhiguang Wang, Weizhong Yan, Tim Oates, *Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline*, `https://arxiv.org/abs/1611.06455`, 2016.

[58] C. Liu, W. Hsaio and Y. Tu, *Time Series Classification With Multivariate Convolutional Neural Network*, IEEE Transactions on Industrial Electronics, vol. 66, no. 6, pp. 4788-4797, 2019.

[59] G. Devineau, W. Xi, F. Moutarde, J. Yang, *Convolutional Neural Networks for Multivariate Time Series Classification using both Inter- & Intra- Channel Parallel Convolutions*, `https://rfiap2018.ign.fr/sites/default/files/ARTICLES/RFIAP_2018/RFIAP_2018_Devineau_Convolutional.pdf`,RFIAP, 2018.

[60] Arthur Le Guennec, Simon Malinowski, Romain Tavenard, *Data Augmentation for Time Series Classification using Convolutional Neural Networks*, ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data, Italy, 2016.

[61] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, Pierre-Alain Muller, *Deep learning for time series classification: a review*, Data Mining and Knowledge Discovery, Volume 33, Issue 4, pp. 917–963,2019.

[62] François Petitjean, Germain Forestier, Geoffrey Webb, Ann Nicholson, Yanping Chen, Eamonn Keogh, *Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm*, Knowledge and Information Systems (KAIS), Springer, pp.1 - 26, 2016.

[63] Young-Seon Jeong, Myong K.Jeong, Olufemi A.Omitaomu, *Weighted dynamic time warping for time series classification*, Pattern Recognition, Volume 44, Issue 9, pp. 2231-2240, 2011.

[64] B Ravi Kiran, Dilip Mathew Thomas, Ranjith Parakkal *An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos*, Computer Vision and Pattern Recognition, 2018.

[65] Rifai Salah, Vincent Pascal, Muller Xavier, Glorot Xavier, Y. Bengio, *Contractive Auto-Encoders: Explicit Invariance During Feature Extraction*, Proceedings of the 28th International Conference on Machine Learning, ICML, 2011.

[66] C. M. Bishop, *Pattern Recognition and Machine Learning*, Pattern Recognition and Machine Learning. Springer, New York, 2006.

[67] C. Garcia and M. Delakis, *Convolutional face finder: a neural architecture for fast and robust face detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 11, pp. 1408-1423, 2004.

[68] Rohit J. Kate, *Using dynamic time warping distances as features for improved time series classification*, Data Mining and Knowledge Discovery, Volume 30, Issue 2, pp. 283–312, 2016.

[69] Mit Shah, Josif Grabocka, Nicolas Schilling, Martin Wistuba, Lars Schmidt-Thieme, *Learning DTW-Shapelets for Time-Series Classification*, Proceeding CODS '16 Proceedings of the 3rd IKDD Conference on Data Science, 2016.

[70] A. I. Moustapha, R. R. Selmic, *Wireless Sensor Network Modeling Using Modified Recurrent Neural Networks: Application to Fault Detection*, IEEE Trans. on Inst. and Measurement, vol. 57, no. 5, pp. 981-988, 2008.

[71] A. Malhi, R. Yan, R. X. Gao, *Prognosis of Defect Propagation Based on Recurrent Neural Networks*, IEEE Trans. on Instr. and Measurement,vol. 60, no. 3, pp. 703-711, 2011.

[72] Wald. Abraham, *Sequential Tests of Statistical Hypothese*, Annals of Mathematical Statistics, 1945.

[73] Sawasd Tantaratana, John B.Thomas, *Truncated sequential probability ratio test*, Information Sciences, Volume 13, Issue 3, pp. 283-300, 1977.

[74] Alex Graves, Abdel-rahman Mohamed and Geoffrey Hinton, *Speech Recognition with Deep Recurrent Neural Networks*, `https://arxiv.org/abs/1303.5778`,ICASSP 2013.

[75] Haşim Sak, Andrew Senior, Kanishka Rao, Françoise Beaufays, *Speech Recognition with Deep Recurrent Neural Networks*, `https://arxiv.org/abs/1507.06947`, INTERSPEECH 2015 proceedings.

[76] Bruno Stuner, Clément Chatelain, Thierry Paquet, *Handwriting recognition using Cohort of LSTM and lexicon verification with extremely large lexicon*, `https://arxiv.org/abs/1612.07528`, 2016.

[77] Victor Carbune, Pedro Gonnet, Thomas Deselaers, Henry A. Rowley, Alexander Daryin, Marcos Calvo, Li-Lun Wang, Daniel Keysers, Sandro Feuz, Philippe Gervais, *Fast Multi-language LSTM-based Online Handwriting Recognition*, `https://arxiv.org/abs/1902.10525`, 2019.

[78] Venu Dasigi, Reinhold C.Mann, Vladimir A.Protopopescu, *Information fusion for text classification — an experimental comparison*, Pattern Recognition Volume 34, Issue 12, pp. 2413-2425, 2001.

[79] K.Mani, P.Kalpana , *A Review on Filter Based Feature Selection* , International Journal of Innovative Research in Computer and Communication Engineering,Vol. 4, 2019.

[80] Ronen Eldan, Ohad Shamir, *The Power of Depth for Feedforward Neural Networks* , `https://arxiv.org/pdf/1512.03965v4.pdf`,COLT, 2016.

[81] Nima Hatami, Yann Gavet, Johan Debayle, *Classification of Time-Series Images Using Deep Convolutional Neural Networks* , The 10th International Conference on Machine Vision (ICMV 2017).

[82] Yosinski J, Clune J, Bengio Y, and Lipson H, *How transferable are features in deep neural networks* , Advances in Neural Information Processing Systems 27 (NIPS '14), NIPS Foundation, 2014.

[83] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, *Going Deeper with Convolutions*, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1-9, 2015.

[84] Y. Freund and R. E. Schapire, *A decision-theoretic generalization of online learning and an application to boosting*, Proc. Eur. Conf. Comput. Learn. Theory, pp. 23–37, 1995.

[85] C. K. Chow, *Statistical Independence and Threshold Functions*, IEEE Transactions on Electronic Computers, EC-14(1),pp. 66–68, 1965.

[86] L. Hansen, P. Salamon, *Neural network ensembles*, Pattern Analysis and Machine Intelligence, IEEE Transactions on pattern analysis and machine intelligence, vol. 12, pp. 993 –1001, 1990.

[87] L. Hansen, P. Salamon, *Neural network ensembles*, Pattern Recognition Volume 29, Issue 2,pp. 341-348, 1996.

[88] Michał Woźniaka, Manuel Grañab, Emilio Corchadoc, *A survey of multiple classifier systems as hybrid systems*, Information Fusion Volume 16, pp. 3-17, 2014.

[89] L. Breiman, *Bagging predictors*, Machine Learning 24, pp. 123–140, 1996.

[90] R. Schapire, *The strength of weak learnability*, Machine Learning 5, pp. 197–227, 1990.

[91] D. Wolpert, *Stacked Generalization*, Neural Networks 5, pp. 241-259, 1992.

[92] Ying CAO, Qi-GuangMIAO, Jia-Chen LIU, Lin GAO, *Advance and Prospects of AdaBoost Algorithm*, Acta Automatica Sinica Volume 39, Issue 6, pp. 745-758, 2013.

[93] V. Le and X. Yao, *Ensemble Machine Learning Based Adaptive Arc Fault Detection for DC Distribution Systems*, 2019 IEEE Applied Power Electronics Conference and Exposition (APEC), pp. 1984-1989, 2019.

[94] Mohammad Moghimi, Mohammad Saberian, Jian Yang, Li-Jia Li, Nuno Vasconcelos, Serge Belongie, *Boosted Convolutional Neural Networks*, British Machine Vision Conference (BMVC), 2016.

[95] C. Wu, W. Gan, D. Lan and C. -. J. Kuo, *Boosted Convolutional Neural Networks (BCNN) for Pedestrian Detection*, 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 540-549, 2017.

[96] S. Lee, T. Chen, L. Yu and C. Lai, *Image Classification Based on the Boost Convolutional Neural Network*, IEEE Access, vol. 6, pp. 12755-12768, 2018.

[97] Rimah Amami, Dorra Ben Ayed, Noureddine Ellouze, *The challenges of SVM optimization using Adaboost on a phoneme recognition problem*, IEEE 4th International Conference on Cognitive Infocommunications, pp. 463-468, 2013.

[98] Ian Goodfellow, Yoshua Bengio, Aaaron Courville, *Deep Learning*, MIT Press, p. 196, 2016.

[99] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, *Learning long-term dependencies with gradient descent is difficult*, IEEE Transactions on Neural Networks, pp. 157-166, 1994.

[100] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jurgen Schmidhuber, *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*, A field guide to dynamical recurrent neural networks, 2001.

[101] Sepp Hochreiter and Jurgen Schmidhuber, *Long short-term memory*, Neural Computation, 9(8), pp. 1735-1780, 1997.

Résumé substantiel

Cette thèse a été mené dans le cadre d'une convention de collaboration entre la société Hager et l'Institut Jean Lamour. Le principal sujet abordé dans la thèse est la détection de défaut d'arc électrique pour les installations domestiques.

Un défaut d'arc est un arc électrique qui se produit accidentellement en série ou en parallèle avec des appareils. L'arc parallèle est produit entre deux tensions différentes (par exemple entre phase et neutre) et l'arc de série se produit dans la même phase. Fréquemment, des défauts d'arcs électriques sont causés par des câbles et des appareils électriques endommagés ou un mauvais contact électrique. Si un défaut d'arc est maintenu pendant une durée suffisamment longue, l'énergie produite par l'arc électrique peut conduire à départ d'un incendie.

Pour protéger à nouveau ce phénomène, des dispositifs de détection de défauts d'arcs électriques sont demandés dans plusieurs installations électriques depuis une vingtaine d'années. Les États-Unis et le Canada exigeaient qu'ils protègent la plupart des points des prises d'électrique résidentiels. En Europe, l'adoption des dispositifs de détection de défauts d'arc commence également, par exemple en Allemagne (2017) et au Royaume-Uni (2018) pour certains types d'installation.

Il existe sur le marché de nombreux dispositifs de détection des défauts d'arcs électriques, mais aucun d'entre eux ne peut garantir une protection absolue. Ils sont tous susceptibles d'avoir des faux négatif ou des faux positifs (ils ne détectent pas un défaut d'arc ou ne reconnaissent un état normale de réseau comme un défaut d'arc électrique).L'arc parallèle peut être facilement détecté en raison de son impact sur le niveau du courant, par contre le défaut d'arc en série est beaucoup plus compliqué à gérer. Jusqu'à présent, aucune solution ne peut garantir la détection de tous les défauts de l'arc sans jamais produire de déclenchement intempestif. C'est la raison

pour laquelle la détection des défauts d'arcs en série est un sujet de recherche actif. Une précision d'environ 98-99% peut être observé dans les résultats de l'état de l'art, cependant, ils ne sont souvent testés que sur quelques cas spécifiques, en outre, le pourcentage d'erreur parait faible mais peut être insuffisant.

On peut imaginer les conséquences d'un défaut de déclenchement dans le cas d'arc ou d'un déclenchement quotidien sans raison valable. La nature chaotique du défaut d'arc et le nombre de scénarios possibles rendent le modèle conventionnel peu pratique et imprécis. Avec l'augmentation de la puissance de calcul pour le système embarqué et les résultats prometteurs de l'apprentissage automatique dans les nombreux domaines, quelques travaux de recherche sur les méthodes de détection des défauts d'arc basées sur l'apprentissage machine se trouvent dans la littérature.

Motivés par le fait que nous pouvons obtenir des résultats d'état de l'art avec des algorithmes basés sur l'apprentissage automatique, nous nous concentrons dans cette thèse sur la résolution du problème de détection de défaut d'arc en série avec les différentes techniques. Les résultats ont été présentés dans nos publications [1,5]. Les principales contributions de la thèse sont les suivantes: Nous avons proposé une méthode permettant de choisir les caractéristiques optimales de défaut d'arc pour la tâche de détection. En outre, le procédé permet de créer un algorithme de détection basé sur plusieurs caractéristiques de défaut d'arc.

Nous avons montré que différentes techniques d'apprentissage en profondeur peuvent fournir des bons résultats pour le problème de détection de défaut d'arc série. Nous avons présenté deux méthodes pour optimiser les performances de la détection en prenant compte de l'aspect variable dans le temps de la détection de défaut d'arc. Un algorithme de détection d'arc de défaut peut être divisé en trois parties principales: mesure - extraction de caractéristiques, classification entre situation normale et anormale et décision. Fréquemment, le courant ou la tension du réseau électrique sera surveillé, puis l'étape d'extraction des fonctions permet de rechercher une caractéristique de défaut d'arc (AFF) à partir du signal acquis. Pour obtenir de bonnes performances lors de la détection, des AFFs appropriées sont très importantes.

Il existe quelques d'algorithmes de détection ceux ont une partie d'extraction de caractéristiques directe telle que fractale, intégrale de courant, variation de courant

[11, 13], donc AFF peut être déduite sans étape supplémentaire. Dans les autres méthodes de détection, la partie extraction de caractéristiques peut être divisée en deux sous-parties: transformation et les descripteurs. Plusieurs transformations ont été utilisées pour la détection d'arc, telles que: transformation de Fourier [14,15] [47], transformation en ondelettes [14] [16, 17], filtrage [15] [46] [49] [50], corrélation [18].

De même, un certain descripteurs peuvent être listées: variation de l'énergie du sous-spectre entre deux cycles de puissance adjacents en tant que descripteur pour la transformée de Fourier discrète (DFT) [14], le rapport harmonique - DFT [19], la valeur moyenne de la différences [20] entre les spectres basse fréquence du courant, mesurées dans deux observations successifs avec transformée en chirp Zeta [21], valeur propre - filtrage de Kalman [22]. La partie classification peut être simple, comme un seuil fixe pour les AFFs [23], ou plus complexe, comme un réseau de neurones artificiels (ANN) [14] [24, 25], Machine à vecteurs de support (SVM) [11] [26]. La technique de comptage ou la logique floue sont utilisées comme stratégie de décision [21] [27, 28] [44] [49]. La principale difficulté de la détection des défauts d'arc en série est la distinction entre les situations d'arc et les situations normales pour tous les types de charges. La forme d'onde du courant ou de la tension peut varier de nombreuses manières en fonction des conditions du réseau et des chargés connectés.

En général, pour détecter une condition d'arc dangereuse, un ou plusieurs AFF peuvent être utilisés [13] [20]. Les AFF les plus utilisés sont: le passage à zéro, le bruit large bande, les caractères aléatoires de la variation du courant [18] [29,30] [48]. Cependant, ces fonctionnalités peuvent également être trouvées dans un réseau de fonctionnement normal.

Par exemple, un aspirateur peut produire de nombreuses caractéristiques d'arc aux passages à zéro. Les caractères aléatoires de la variation de courant peut également être imité en modifiant le mode de fonctionnement d'un appareil, allumer / éteindre rapidement un appareil. De plus, le bruit à large bande peut être généré par une interférence électromagnétique ou une charge bruyante. En raison de ce problème, chaque algorithme de détection propose un seuil permettant de discriminer les arcs dangereux des situations normales, autrement dit, un compromis

entre les déclenchements intempestifs (faux négatif) et l'échec de la détection d'un défaut d'arc (faux positif). Ils ont des conséquences différentes. Un déclenchement indésirable peut conduire à une perte de temps de travail et à un service désactivé. L'échec de la détection d'un défaut d'arc est plus problématique et peut entraîner une perte d'équipement, voire de vie. Par conséquent, ces erreurs doivent être minimisées autant que possible. On peut constater qu'une AFF peut donner de meilleures performances de détection qu'une autre AFF dans certaines situations et inversement pour le reste [31].

Le problème de détection de défaut d'arc peut être considéré comme une classification de série temporelle avec deux classes: arc et normal. La série temporelle correspond à l'acquisition du courant ou de la tension sur une durée définie. La durée peut être fixe ou variable. En effet, pour un niveau faible de courant, le système peut se permettre de prendre plus de temps pour confirmer le défaut qu'à un niveau élevé de courant. Au faible courant, une seconde du signal peut être utilisée comme entrée et descendre jusqu'à 0,12 seconde lorsque le niveau monte. La première et la plus populaire approche consiste à utiliser des séries de longueur fixe pour la détection d'un défaut d'arc. La taille de fenêtre d'analyse ne doit pas dépasser 6 périodes dans le cas d'une installation européenne (le temps de détection le plus court est de 120 ms). Nous pouvons lister plusieurs méthodes de détection qui utilisent un signal de longueur fixe comme entrée: [11] [45] [51].

Cette approche facilitée la conception des paramètres d'entrées et des classificateurs. Tenant compte de ces avantages, nous avons d'abord consacré du temps à la mise au point d'une méthode de détection reposant sur des paramètres d'entrées et des classificateurs élaborés. Dans le cas de charges simples telles que résistives ou inductives, la précision de prédiction de certaines méthodes peut atteindre 100%. Même avec des charges non linéaires (ordinateur, perceuse électrique ...) où la tâche de détection devient plus compliquée, certains algorithmes restent avec une précision très proche de 100%. Chaque méthode a l'avantage pour un ou plusieurs types de charge. Par conséquent, l'approche multicritères peut conduire à des résultats de pointe si toutes les méthodes disponibles peuvent être combinées correctement. L'idée d'utiliser multiples d'AFF pour améliorer les performances de détection a été

évoquée dans certaines publications et certains brevets: vérification de la présence de différentes AFF [20], en utilisant les caractéristiques temporelles ainsi que l'analyse de la fréquence [13]. Cependant, il manque encore des éléments très importants pour la détection de plusieurs AFF: le choix des AFFs à utiliser ensemble, l'algorithme de combinaison permettant de créer un algorithme de détection efficace à partir des AFFs choisies.

Pour répondre à ces questions, une méthodologie d'optimisation des performances de détection des défauts d'arc avec plusieurs AFF a été développée. La principale originalité de la méthode proposée est l'utilisation de la sélection supervisée des caractéristiques. La méthode consiste à créer un pool d'AFF de défaut d'arc et à rechercher une combinaison des AFFs répondant aux performances souhaitées. Des résultats expérimentaux avec des situations de base (norme IEC 62606) et des situations compliquées (transitoires, charges de masquage multiples ou perturbations sur le réseau électrique, etc.) ont démontré l'efficacité des méthodes proposées. Vingt et une transformées spécifiques associées à 10 descripteurs différents ont été évaluées, en termes de précision, une combinaison de trois AFF peut atteindre 99,85%.

Grâce à l'apprentissage en profondeur, de plus en plus d'approches pour classer les séries temporelles ont été présentées. Généralement, ces approches tendent à remplacer tout ou partie de l'étape d'extraction des caractéristiques par un apprentissage en profondeur. Ils visent à combiner le processus d'apprentissage des caractéristiques tout en optimisant le classificateur discriminant; ils réduisent le besoin d'ingénierie pour l'extraction des caractéristiques et des connaissances spécifiques au domaine. Parmi ces différentes techniques, les auto-encodeurs, les réseaux de neurones à convolution (CNN) et les réseaux de neurones à récurrences sont les plus utilisés. Par exemple, Guifang Liu et al. a utilisé la transformation de Fourier et des encodeurs automatiques empilés comme extraction de caractéristiques pour analyser la défaillance de boîte de vitesses [52]. Ou Mohendra Roy et al. a proposé une méthode de détection d'anomalie dans la surveillance des conditions basée sur des auto-encodeurs superposés en tant qu'extraction de caractéristiques et sur un réseau de machines d'apprentissage extrême séquentiel en ligne en tant que classificateur [53]. Ma Q et al .; Bianchi et al .; Aswolinskiy et al. utilisé des réseaux

de neurones de récurrence pour classer des séries temporelles [54, 56]. Z.Wang et W.Yan; C. Liu et al.; G.Devineau; Le Guennec A et al. proposées des méthodes de classification basées sur certaines variantes de CNN [57, 60]. Les résultats concernant la précision de la classification pour les séries temporelles avec apprentissage en profondeur sont parfois meilleurs que ceux obtenus avec les méthodes classiques [61].

Nous avons appliqué deux techniques d'apprentissage en profondeur pour la tâche de détection de défaut d'arc. Le résultat donné par les auto-encodeurs est moyen (96.2%). Les méthodes de détection basées sur CNN offrent de meilleures performances par rapport les auto-encodeurs (99.15%) et nous pouvons encore améliorer la précision des prévisions avec plus de données d'apprentissage. Le principal inconvénient de méthodes basées sur l'apprentissage profond est son coût de calcul. Il est coûteux de mettre en œuvre les méthodes proposées sur une plate-forme embarquée.

Dans le réseau domestique, il n'est pas nécessaire de détecter immédiatement quand une situation d'arc commence tout juste. Identifier le défaut d'arc avant tout dommage grave au système et limiter le nombre de fausses alarmes est le plus importantes. Logiquement, plus le temps d'attente est long, plus il est facile d'acquérir des informations et donc de prendre de meilleures décisions. Plusieurs méthodes peuvent être utilisées pour classifier les séries temporelles de longueur des variables pour la détection des défauts d'arc. Les deux approches les plus utilisées sont des caractéristiques indépendantes de la durée d'analyse, telles que la fréquence ou le domaine temps-fréquence, et des fenêtres d'analyse prédéfinies, combinées avec les méthodes statistiques. Par exemple, Yu-Wei Liu et al. ont utilisé un cycle d'alimentation (20ms) comme fenêtre temporelle et analysé 30 cycles d'alimentation avec la méthode de comptage pour détecter une défaillance d'arc en série [14]. Ou Artale et al. a développé un algorithme de détection basé sur l'analyse de fréquence utilisant différentes fenêtres d'observation de tailles différentes pour s'adapter au valeur RMS du courant [21] Edwin et al. ont présenté une méthode qui détecte un arc dans un temps variable (20-200ms), qui dépend du type de charge [49].

En plus de cela, l'adaptation temporelle dynamique a été largement utilisé pour la classification dans d'autres domaines [62, 69]. Cependant, dans cette application,

la longueur de la série d'entrée peut être considérablement modifiée et le motif principal de la forme d'onde du courant d'entrée est répété entre les périodes; il n'est donc pas pratique d'adapter les signaux d'entrée pour obtenir la même longueur. Une autre solution est proposée dans l'article [61]. Il consiste à utiliser un réseau de neurones récurrent (RNN). Par exemple, le RNN est utilisé pour détecter le nœud de défaillance dans un réseau de capteurs sans fil [70].Ou Arnaz Malhi et al. ont utilisé le RNN pour prédire l'état de l'état de la machine [71]. Dans cette thèse, nous présentons deux méthodes de détection de défaut d'arc à temps variable. La première méthode est basée sur l'observation temporelle prédéfinie et l'analyse statistique d'observations multiples. La deuxième méthode utilisait un réseau neuronal à mémoire de long et de court terme. Avec ces deux méthodes la précision obtenue approche les 100% dans la majorité des situations.

Résumé

La détection des arcs électriques se produisant dans un réseau électrique par des approches d'apprentissage automatique représente le cœur des travaux exposés dans cette thèse. Le problème a d'abord été vu comme la classification de séries temporelles à taille fixe en deux classes: normal et défaut. Cette première partie s'appuie sur les travaux de la littérature où les algorithmes de détection sont organisés principalement sur une étape de transformation des signaux acquis sur le réseau, suivie d'une étape d'extraction de caractéristiques descriptives et enfin d'une étape de décision. L'approche multicritères adoptée ici a pour objectif de répondre aux imprécisions systématiquement constatées. Une méthodologie de sélection des meilleures combinaisons et de transformation et de descripteurs a été proposée en exploitant des solutions d'apprentissage. La mise au point de descripteurs pertinents étant toujours difficile, l'évaluation des possibilités offertes par l'apprentissage profond a également été étudiée. Dans une seconde phase l'étude a porté sur les aspects variables dans le temps de la détection de défaut. Deux voies statistiques de décision ont été explorées l'une basée sur le test de probabilités séquentiel (SPRT) l'autre basée sur les réseaux de neurones artificiels LSTM (Long Short Time Memory Network) chacune de ces deux méthodes exploite à sa manière la durée d'une première étape de classification comprise entre 0 et 1 (normal, défaut). La décision par SPRT utilise une intégration de la classification initiale. LSTM apprend à classer des données à temps variable. Les résultats du réseau LSTM sont très prometteurs, il reste néanmoins quelques points à approfondir. L'ensemble de ces travaux s'appuie sur des expérimentations avec des données les plus complètes et les plus large possible sur le domaine des réseaux alternatifs 230V dans un contexte domestique et industriel. La précision obtenue approche les 100% dans la majorité des situations.

Abstract

The detection of electric arcs occurring in an electrical network by machine learning approaches represents the heart of the work presented in this thesis. The problem was first considered as a classification of fixed-size time series with two classes: normal and default. This first part is based on the work of the literature where the detection algorithms are organized mainly on a step of the transformation of the signals acquired on the network, followed by a step of extraction of descriptive characteristics and finally a step of decision. The multi-criteria approach adopted here aims to respond to systematic classification errors. A methodology for selecting the best combinations, transformation, and descriptors has been proposed by using learning solutions. As the development of relevant descriptors is always difficult, differents solutions offered by deep learning has also been studied. In a second phase, the study focused on the variable aspects in time of the fault detection. Two statistical decision paths have been explored, one based on the sequential probabilistic test (SPRT) and the other based on artificial neural networks LSTM (Long Short Time Memory Network). Each of these two methods exploits in its way the duration a first classification step between 0 and 1 (normal, default). The decision by SPRT uses an integration of the initial classification. LSTM learns to classify data with variable time. The results of the LSTM network are very promising, but there are a few things to explore. All of this work is based on experiments with the most complete and broadest possible data on the field of 230V alternative networks in a domestic and industrial context. The accuracy obtained is close to 100% in the majority of situations.