Università degli Studi di Milano
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di Laurea Magistrale in Informatica

Interpreting Raman spectroscopy towards diagnostic purposes : an explainable deep-learning based approach

Supervisor: Prof.ssa Vincenzina Messina
Co-supervisor: Dott. Dario Bertazioli
Dott. Cristiano Carlomagno

Master's Degree Thesis by:
Nina Singlan
Student number 867646

Academic Year 2020-2021

Acknowledgements

This first part is dedicated to all the people who allowed me to carry out this thesis, and, more generally, to complete my university studies.

First of all, I would like to express my sincere thanks to Professor Vicenzina Messina who allowed me to participate in this project. I would also like to thank her for her help and availability and for accepting to give a chance to a foreign student she did not know.

I would also like to thank from the bottom of my heart Dario Bertazioli without whom this thesis project would never have seen the light of day. Thanks for his help, his availability and his explanations.

I would also like to express my special thanks to the members of the Université Nice Côte d'Azur who allowed me to learn with them for 4 years before giving me the opportunity to finish my training at the Università degli Studi di Milano Bicocca.

On a more personal note, I would also like to thank my parents who, year after year, have always pushed me to do my best. Thank you for your support, both financial and emotional.

Finally, this section of acknowledgments would not be complete without addressing my most sincere thanks to Romain Michelucci, for his daily support, his help and his sincere encouragement. Thank you for everything.

Abstract

In the past few years, Raman spectroscopy allied with Machine Learning (ML) methods has shown great performances for diagnostic purposes. Raman spectroscopy is commonly used in chemistry to provide a structural fingerprint by which molecules can be identified. Then, ML methods are applied to decode resulting spectra. Thus, Raman spectroscopy allied with ML techniques promises a new, rapid and non-invasive way to diagnose patients. This ability has led researchers to highlight that Convolutional Neural Networks (CNNs) outperform other ML methods. The laboratory, where this study takes place, has already explored the CNNs configuration and has built one whose accuracy of around 89–92 % for diagnosing Covid-19 was achieved. However, in medical field, getting good results is not enough and the critical drawback of the CNN approach is the lack of interpretation. It is regarded as a black box. Thus, the purpose of this study is to suggest an interpretation of the CNN learning mechanism proposed in the previous work. This research hopes to implement a method in order to allow an intuitive understanding and further application.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

1.1 General context of the thesis

This work is part of a collaboration between the LABION (Laboratorio di Nanomedicina e Biofotonica Clinica) research group of the Don Gnocchi Foundation and the MIND (Models in decision making and data analysis) research laboratory of the University of Milan-Bicocca. Aims of this collaboration are the study and design of Machine Learning and Deep Learning algorithms for the analysis, characterization and classification of Raman spectra from saliva samples. The ultimate goal of this project is to allow the development of an innovative, rapid and non-invasive screening methodology for the diagnosis of neurodegenerative and COVID diseases.

The first thing that was done to achieve this goal was to create a deep learning model, called a classifier, which is able to tell whether a patient is infected with the disease or not, to classify him or her as healthy or sick. This classifier is the starting point for the work presented here. Indeed, having a model capable of telling whether a patient is ill or not, even with a very good accuracy, is not enough. Taking the Covid as a case study, the question this work aims to answer is: is the model reliable? To answer this question, the key idea is to understand the classifier: how does it classify? What part of the input data does it focus on to give its answer? These questions are at the heart of Explicable Artificial Intelligence. Thus, in simple terms, the aim of this thesis is to propose an Explicable Artificial Intelligence approach to understanding the classification, in the case of Covid, of Raman spectra.

1.2 Biological and medical context

1.2.1 Different kind of spectroscopy techniques

Spectroscopy is a term used to refer to the measurement of radiation intensity as a function of wavelength. One of the central concepts in this domain is a resonance and its corresponding resonant frequency. There are various implementations and techniques in spectroscopy, they can be classified in several manners. Amongst other ways, the types of spectroscopy can be distinguished by the nature of the interactions between the energy and the material.
There are three main categories that are used in medical fields:

- **Absorption spectroscopy:** Absorption phenomena occur when energy from the radiative source is absorbed by the material. It is often determined by measuring the fraction of energy transmitted through the material, with absorption decreasing the transmitted portion.

- **Emission spectroscopy:** In this spectroscopic technique the radiative energy is released by the material. There are different kind of emissions, some are spontaneous and others are induced.

- **Spectroscopy based on inelastic scattering:** The inelastic scattering phenomena involve an exchange of energy between the radiation and the matter that shifts the wavelength of the scattered radiation. Unlike elastic scattering, the kinetic energy of an incident particle is not conserved, in other words, some of the energy of the incident particle is lost or increased.

The latter type of spectroscopy is the one that this work focuses on. Indeed, the Raman Spectroscopy relies upon inelastic scattering of photons.

1.2.2 Introduction to Raman spectroscopy

Raman Spectroscopy is a non-destructive chemical analysis technique which provides detailed information about chemical structure, phase and polymorphy, crystallinity and molecular interactions of materials. It is based upon the interaction of light with the chemical bonds within a material. More precisely it is based on Raman Effect, that is to say, in certain circumstances, frequency of a small fraction of scattered radiation is different from frequency of monochromatic incident radiation. This effect is also called Raman scattering and is a type of inelastic scattering. As mentioned earlier, this effect is inelastic scattering on photons, which means that the incident particle is a photon. Absorption of a photon excites the molecule to the imaginary state and re-emission leads to Raman or Rayleigh scattering. As it is shown in Fig. 1.1, there are three different case, in all of them the final state has the same electronic energy as the starting state but is higher in vibrational energy in the case of Stokes Raman scattering (shown in orange in the figure), lower in the case of anti-Stokes Raman scattering (shown in blue in the figure) or the same in the case of Rayleigh scattering (shown in green in the figure).

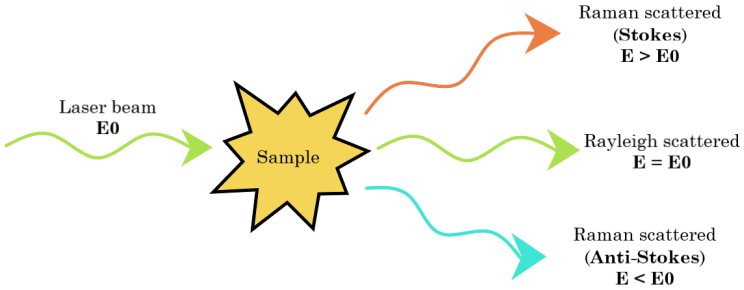

Figure 1.1: Basic scheme of the Raman scattering. The original vibrational energy is denoted by E0 while the resulting one is denoted by E.

The spectrum of the Raman-scattered light depends on the molecular constituents present and their state, allowing the spectrum to be used for material identification and analysis. Thus, Raman spectroscopy is a versatile method used to analyze a wide range of materials, including gases, liquids, and solids. Highly complex materials such as biological organisms and human tissue can also be analyzed by Raman spectroscopy. It resolves most of limitations of other spectroscopic techniques. It can be used for both qualitative as well as quantitative purpose. Qualitative analysis can be performed by measuring the frequency of scattered radiations while quantitative analysis can be performed by measuring the intensity of scattered radiations.

A Raman spectrum features a number of peaks, showing the intensity and wavelength position of the Raman scattered light. Each peak corresponds to a specific molecular

bond vibration.
The key features of a Raman spectra are :

- **The Raman shifts and relative intensities of all of the Raman bands of the material:** Basically, the Raman shift is the energy difference between the incident (laser) light and the scattered (detected) light. With these, a material with a particularly caracteristic Raman shift can be identified.
- **Individual band changes:** A band may shift, narrow or broaden, or vary in intensity. These changes can reveal information about stresses in the sample, variations in crystallinity, and the amount of material respectively.
- **Variations in spectra with position on the sample:** This will reveal changes in the uniformity (homogeneity) of the material. You can analyse at several arbitrary points, or systematically measure an array of points (enabling the production of images of composition, stress, crystallinity, etc.)

In practice, the spectra in this study are acquired using the Surface-enhanced Raman Spectroscopy (SERS) version of Raman spectroscopy which is characterised by the addition of a particular nano-structure (in this case, an aluminium substrate) which can enhance the Raman signal.

As mentioned earlier, the main aim of this thesis is to understand the classification of Raman spectra of salivary fingerprints. This classification is done according to three different classes: the patient is either Covid-positive, Covid-negative (meaning that he/she has already been infected, but is cured), or a control patient (he/she does not have Covid and has never been infected).Thus, for the sake of clarity, the Raman spectra shown as examples in Fig. 1.2 are taken directly from the Covid dataset which will be used throughout this work.

1.2.3 Medical application

The Raman spectroscopy technique is capable of providing details on chemical composition, molecular structure, and molecular interactions in cells and tissues. Thus, changes in the tissues leaded by a disease could be reflected in the spectra. If theses changes are typical of a certain disease, the Raman spectra can be used for diagnostic purposes.

In the scientific literature, there are already many studies concerning the application of Raman spectroscopy as a clinical diagnostic tool [1] [2] [3]. This technique can be based on tissue analysis or on bio-fluid analysis [4]. Thus, Raman spectroscopy offers a new method of non-invasive diagnosis. In this study, samples are taken from saliva, which is a complex biofluid, being composed by several different molecules, among which proteins, metabolites, carbohydrates, nucleic acids, and hormones, in an overall aqueous solution. Not only presence of these molecules can lead to potential hints for the identification of a pathological state, but also their concentrations and variations in concentration, as well as their modifications, interactions and environments, can give insights for the disease progression and response to specific therapies. Moreover, these molecules are represented in pathological state in specific patterns that involve several immunological and physiological biomarkers, with the clear difficulty of the overall detection of all the associated levels.

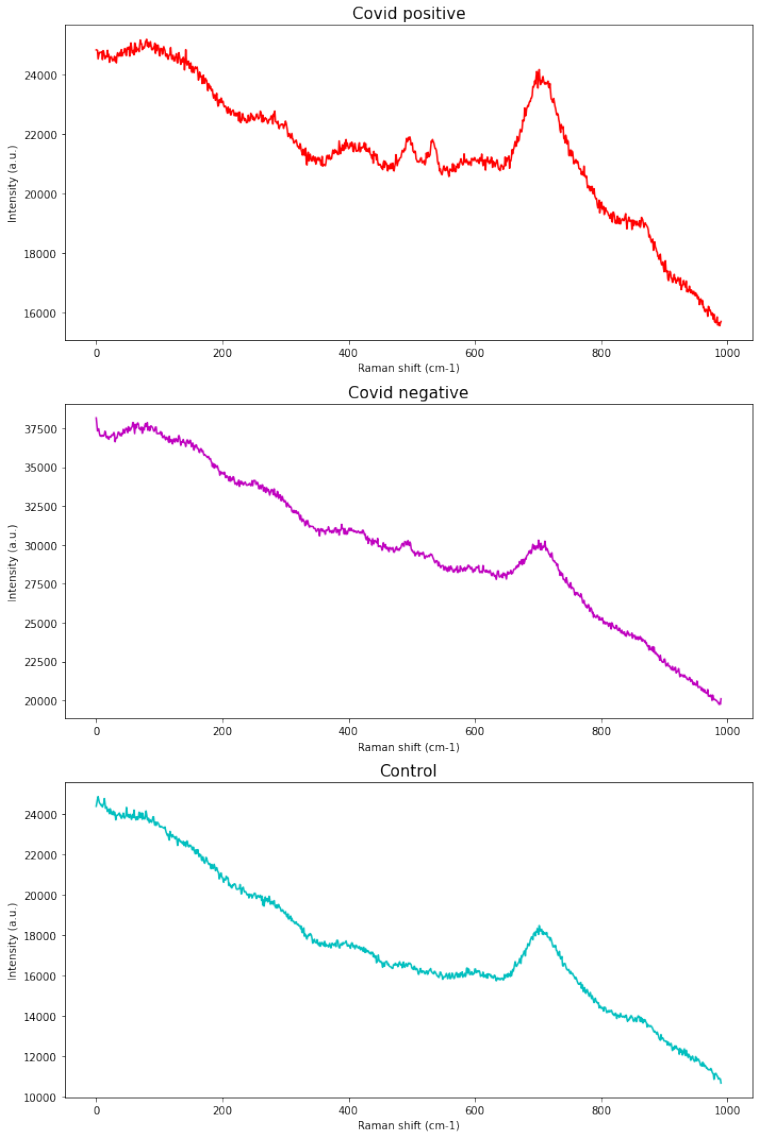

Figure 1.2: Example of Raman spectra. Each one is taken from a different classes. The first one come from the Covid Positive (Covid+) class, the second one came from the Covid Negative (Covid-) one and the third one is from the Control (CTRL) one.

1.3 Computer science context

1.3.1 Motivations

As can be noticed in Fig. 1.2, it is not easy to distinguish between a Covid-infected patient, a cured patient and a patient who has never been infected. Data analysis and computational techniques can be applied to the spectral data to unveil new insights in the sample characterization as well as to enable the discrimination of different classes and the identification of patterns and structures in the analyzed targets. Machine Learning (ML) techniques have already shown its ability to decode the Raman Spectra [5] [6]. Thus, Raman spectroscopy allied with ML techniques promises a new, rapid and non-invasive way to diagnose patients. This ability has led researchers to highlight that Convolutional Neural Networks (CNNs) outperform other ML methods.

The collaboration in which this work take place has already allow the construction of a promising model [7]. More precisely, a CNN (described in 4.2) which obtain an accuracy in the range 89-92%. In order to really benefit from this classifier, it is necessary to focus on understanding it.

1.3.2 Problem statement

In the past decades, Artificial Intelligence (AI), and more specifically, ML researchers have focus their effort on the results. The point was to obtain always better accuracy in record time. However, these days, obtaining good results is not enough anymore. This observation is even more true when ML is used for medical purposes. Indeed, a diagnostic method must be reliable, false positives and false negatives could lead to medical errors which, depending on the case, can be serious for the patient.

Even if the CNN constructed by the laboratory achieves good results, it does not allow the understanding and the interpretation of the classification. Indeed, one of the critical drawbacks of the CNN approach is the lack of interpretation, it is regarded as a black box. Thus, the purpose of this study is to suggest an interpretation of the CNN learning mechanism proposed in the previous work. This interpretation will be performed using Explainable Artificial Intelligence techniques and more specifically a technique originally designed for convolutional neural networks classifying images, called Class Activation Mapping.

2 Machine Learning

Since the objective of this work is to propose an explanation of a classification model, it is first important to present the theory of machine learning, and more precisely the sub-field of deep machine learning. This section is therefore devoted to this theoretical overview.

2.1 Machine Learning

Machine learning is a branch of artificial intelligence and computer science that focuses on using data and algorithms to mimic the way humans learn, gradually improving its accuracy. Since Deep Machine Learning is a sub-field of Machine Learning, it is first necessary to explain some of the basics of Machine Learning. The purpose of this section is thus to provide an understanding of the basics it theory in order to grasp the work being done here. For a wider and complete perspective on Machine Learning, the reader can refer to [8] [9] [10].

2.1.1 Machine Learning Paradigms

In simple words, an algorithm can be defined as a computer program that can produce an output from an input. Which can be mathematically writing as a function $f : X \longrightarrow Y$ that map from the input space X to the output space Y.

A Machine Learning algorithm is a special type of algorithm that can, by processing input data, produce a function as output. In other words, the output of a Machine Learning algorithm is the function $f : X \longrightarrow Y$ which is called the *model* and applied to unseen data.

Depending in the nature of the "signal" (or "feedback") available to the learning system, Machine Learning can be categorized into three main paradigms :

- **Supervised learning :** This learning paradigm is the most widely studied and can be summarised as follows: "*learning with labelled data*". In supervised learning, the data is a set of "*input-output*" pairs, that is to say the data is composed of n pairs (x_i, y_i) where $i \in [\![1, n]\!]$, $x \in X$ and $y \in Y$. Thus, the algorithm analyses the training data and produces an inferred function, which can be used to map unseen inputs $x_j \in X$.

- **Unsupervised learning :** In contrast to Supervised learning, the data is "*untagged*", that is to say the x is given but not the label y. In this paradigm, the Machine Learning algorithm is used to automatically find pattern from data.

- **Reinforcement learning :** This paradigm lies between the two previous ones. It deals with sequential decision making problems with limited feedback. In Reinforcement learning the focus is on finding a balance between exploration of uncharted territory and exploitation of current knowledge. A learning agent faced with a state s must choose a sequence of actions, interact with the environment, and adjust its behaviour according to the delayed reward provided by the environment. Thus, the

goal of the algorithm is to optimise its policy in order to maximise the accumulated reward.

The three paradigms presented here are the main ones, but they are not the only ones, there are others, such as *semi-supervised learning*. This work focuses on Supervised learning, so the following sections are mainly devoted to it.

2.1.2 Theoretical overview

Formally, a supervised learning algorithm can be defined as a computer program that aims to find a function f, from a function space F, that minimizes a loss function $l()$ over a training dataset D.

Definition 2.1.1 (Supervised Learning Algorithm)

$$f = \min_{f \in F} \frac{1}{|D|} \sum_{(x,y) \in D} l(f(x), y)$$

where $|D|$ is the number of data points in the training dataset D, F is the function space containing functions mapping from the input space X to the output space Y, and $l()$ is the loss function that evaluate the difference between the prediction of function f and the label y.

Thus, according to the above definition, a supervised learning algorithm consist of several typical components which are :

- **The training dataset D :** As aforementioned, training data points are pre-given in the format of input-output pairs.
- **The function space F :** This space, also known as the *hypothesis space*, determines the type of function to be learned from the data (i.e. the type of model). Typically, models can be linear functions or decision trees.
- **The loss function $l()$:** The loss function calculates a value indicating the quality of the prediction made by the model. To do this, it takes as input the prediction $f(x)$ and the actual label y. There are a large number of typical loss functions and there is no "hard and fast rule" for choosing one. Generally, the choice is made according to the type of task to be performed, the type of model and the particularities of the data set.
- **The optimizer needed to perform the *min* operation :** As in the previous point, many optimizers have been designed, each of which is tailored to the loss function and the assumption space.

The above definition is rather formal, but fortunately, in [11], Thomas Mitchell has given a rather simple and clear one, illustrated in Figure 2.1 :

Definition 2.1.2 (Learning Algorithm)
An algorithm is said capable of learning from a set of Experiences E with respect to a Task T and Performance Measure P, if P improves with E.

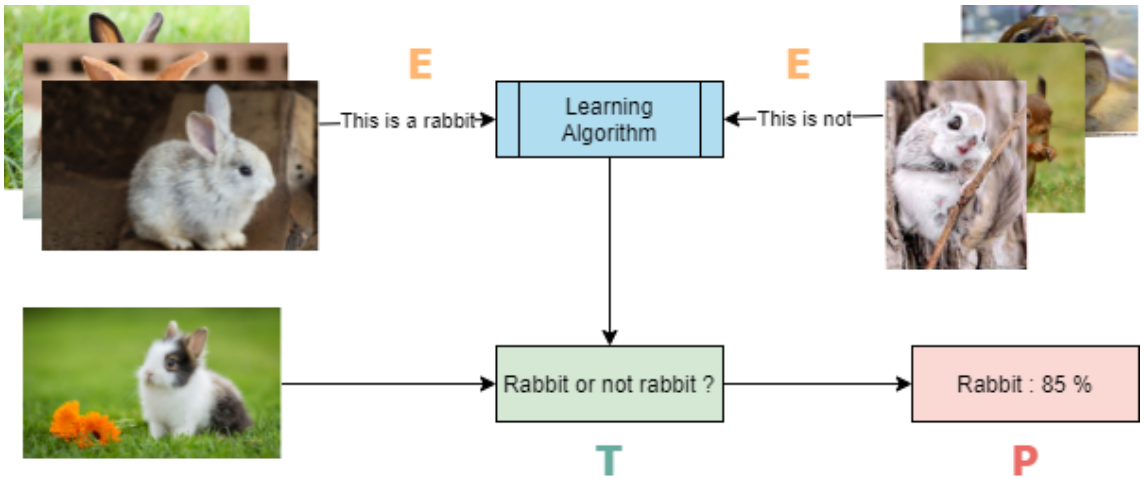

Figure 2.1: Schematic diagram of a learning algorithm, learning from experiences E to accomplish a task T.

The Task T

The learning process is not a task in itself, so the task T mentioned in the definition 2.1.2 must be an appropriate machine learning task, such as an A.I. assistant to detect a rabbit in an image. There are many Machine Learning tasks, among which three typical categories can be distinguished :

- **Prediction :** In prediction problems, the idea is to use data to predict a value. For example, the price of a house depends on its location, size, structure and other elements. Based on these elements, a model can predict the price of the house. In principle, this type of task is performed by a supervised learning algorithm.
- **Classification :** This task consists of classification into categories called classes. The case of the covid studied here is a classification task, it consists in categorising a spectrum, choosing between three classes: positive, negative or control.

As this is the subject of this work, let's dive a little deeper into the task of classification. To accomplish this, the learning algorithm must specify to which k category the input data belongs. Formally, this task can be defined as follows [12].

Definition 2.1.3 (Classification Task)
The learner is required to learn (produce) a function $f : \mathbb{R}^n \longrightarrow \{1, ..., k\}$ *which maps a vector* $\vec{X} \in \mathbb{R}^n$ *into a category identified by the label* $y \in \{1, ..., k\}$. *That is to say, the learner produces a function* f *such that* $y = f(\vec{X})$.

When there are only two classes, it is called **binary classification** and when $k > 2$, the task is called **multi-class classification**, which is the case for covid diagnosis.

The Performance Measure P

To determine the ability of a program to perform the task, it is necessary to carry out a performance evaluation, which is where the performance measure P comes in. This measure is closely related to the class of the specific task T. In classification tasks, the performance measure is commonly derived from the **confusion matrix**, shown in Figure

2.2, which "summarizes the classification performance of a classifier with respect to some test data" [13].

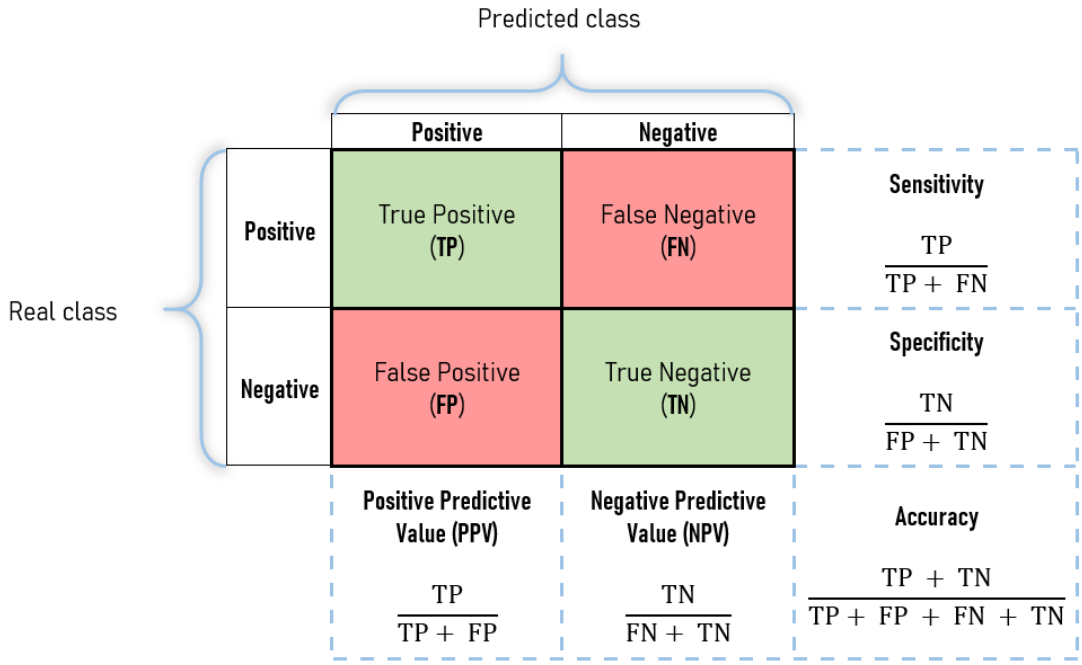

Figure 2.2: A general confusion matrix with formulas for the various measures extracted from it.

Basically, the terms presented in the confusion matrix can be defined as follows :

- **True Positive (TP):** The data is positive and is predicted to be positive.
- **False Negative (TP):** The data is positive but is predicted as negative. This is one of the error case.
- **True Negative (TN):** The data is negative and is predicted as such.
- **False Positive (FP):** The data is negative but is predicted as positive. This is the second case of error.

The most commonly used metric is **accuracy**, which is a measure of the proportion of correctly classified outputs (Def. 2.1.4).

Definition 2.1.4 (Accuracy)

$$\frac{TP + TN}{TP + FP + FN + TN} \in [0, 1]$$

In order to obtain more information on the performance of the model, this measure can be combined with other metrics or presented directly with the following matrix. Although both false positives and false negatives are misclassifications, they are not identical and do not have the same consequences. The case of medical diagnosis is a perfect illustration of this, in fact, both false positives and false negatives can have bad consequences but, depending on the disease one can be worst than the other. It is therefore interesting to know in detail what kind of errors a model makes.

The Experience E

The Experience is "the raw material that feeds the learning process", it is essentially the examples that form the dataset. Thus, the kind of learning algorithm chosen (i.e. Supervised or Unsupervised) is closely related to the kind of Experience that it can handle.

2.1.3 Application to Raman Spectroscopy

Different machine learning techniques were used to analyse various Raman spectra. This section is aimed to propose a quick overview of these application based on the work of [6].

Food

Machine learning algorithms associated with Raman spectroscopy have been used in the food industry. Indeed, these techniques can be used to identify the origin of a food product, thus avoiding certain frauds. In 2013, a study [14] tested butter adulteration using Raman spectroscopy and machine learning methods with good results. In 2018, another team [15] showed that by using a Random Forest algorithm on Raman spectra of milk, the species producing that milk sample could be identified. This collaboration between Machine Learning and Raman Spectroscopy has also been used in the luxury industry to test the purity of caviar [16].

Forensics

The combination of Raman Spectroscopy and Machine Learning has also proven to be very effective in the field of forensics. Firstly, in [17], this combination allows the detection of illegal drugs. And, in the same category, in [18], it has been used to ensure the identity of tablets, to allow quality control of the final product and detection of counterfeits.

Another type of application is blood analysis. In [19], the combination of Raman Spectroscopy and Machine Learning has yielded excellent results in differentiating human from animal blood. Even more impressive is that this combination can be used to determine gender [20] and age [21] from bloodstain.

Usually, when identifying explosives, the strategy is to place a physical barrier (such as glass) between the sample and the instrument (where the operator is located). But, thanks to Raman Spectroscopy and Machine Learning, in [22], a new strategy is proposed. The idea is to develop a hyper-spectral Raman imaging system using a telescope, making it possible to safely acquire Raman spectra of explosives at a distance of 15 metres. These spectra can then be classified using a Random Forest algorithm.

Bacteria and viruses

Raman micro-spectroscopy is a variant of Raman spectroscopy in which the diameter of the laser beam is of the order of 1 millimetre. This variant allows the analysis of individual bacteria and viruses.

In [23] and in [24], this technique is used to study Urinary Tract Infection (UTI) which is a very common infection. In [25], it is used to detect Clostridium Difficile Infections (CDI) more accurately. There are also some examples of application to dengue fever [26] [27] or to the identification of staphylococcal species [28] [29].

Medical diagnosis

In the same spirit as the previous application, Raman Spectroscopy and Machine Learning can be applied to provide a rapid and non-invasive diagnosis.

In [30], this combination is used for screening for diabetes mellitus, which is usually associated with a blood test (which is an invasive process). Another example is the diagnosis of kidney stones [31]. Indeed, the analysis of the type of it plays a crucial role in the follow-up of the patient's treatment. It is therefore very important to have an accurate method of diagnosis, which the combination of Raman Spectroscopy and Machine Learning offers.

One of the most documented applications is cancer diagnosis. In [32] a general discussion of this application is offered, showing a large number of more specific studies. But some particular cases are interesting. Ad example, in [33] an approach is proposed for thyroid cancer that also provides rich information on the biochemical differences between healthy and cancerous thyroid cells. Another study proposes a method for diagnosing cancer in urine [34].

2.2 Deep Machine Learning

Now that the foundations of machine learning have been laid, this chapter is devoted to the development of deep machine learning, which is the theory on which this study is based. For a more complete description of this theory, the reader may refer to [35].

2.2.1 Introduction

The beginning of deep learning can be found in the work of A. G. Ivakhnenko and V. G. Lapa (1967) [36] who were able to conceptualise a supervised, deep, feedforward learning algorithm known as "multi-layer perceptrons". A few years later, in 1971, Ivakhnenko succeeded in training a deep network with 8 hidden layers [37]. Although Deep Machine Learning was already conceived, the term was introduced some years later by Rina Dechter (1986) [38]. A more detailed timeline is shown in Figure 2.3.

Basically, deep learning can be summarised as the study of artificial neural networks that contain more than one hidden layer. These layers lie between the input and output layers and this is why they are called hidden, which implies that they are not visible to external systems. This complex architecture allows the network to generate complex hypotheses by itself, whereas with a simple machine learning network, these hypotheses must be written by the computer scientist. Thus, Deep Learning is a powerful tool to learn nonlinear relationships.

2.2.2 Deep Learning vs Machine Learning

The features can be described as appropriate representations of the information contained in the input data. More than the number of hidden layers, what really distinguishes deep learning algorithms from machine learning algorithms is the ability to learn these features (**Features Learning**). Typically, in machine learning models, features are extracted manually, for example by applying a principal component analysis (PCA), before applying the model. This application of PCA allows for more efficient training by reducing dimensionality, but also by highlighting important information in the data by reducing

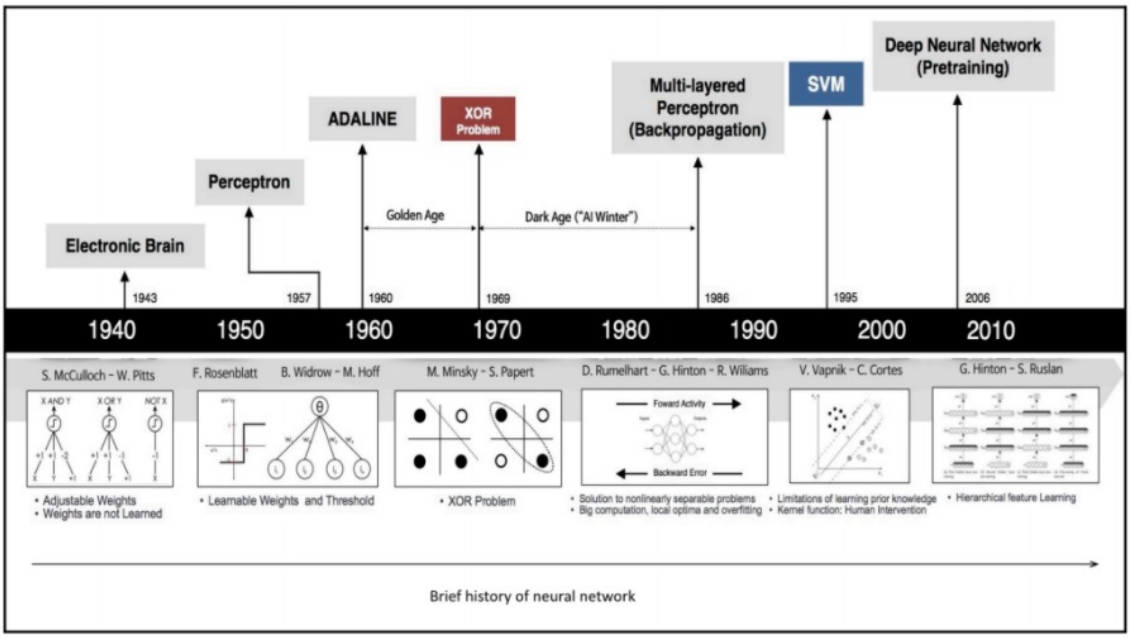

Figure 2.3: Timeline of Deep Machine Learning development. Figure taken from [39].

redundancy. In this sense, this operation can be considered as a **feature engineering** step.

Feature engineering

Formally, an example of feature engineering can be the following. If one considers a linear model, it is obviously affected by the inability to fit non-linear features well enough. Yet linear models can be extended to represent non-linear features $f(X)$ by transforming the input X via an appropriate map ϕ

$$y = \phi(X)^T \, W \tag{2.1}$$

with W a weight vector.

To choose the mapping ϕ several approaches exist :

1. **Use a generic ϕ :** This option relies on the fact that ϕ fits in a higher dimensional space, where the problem is more likely to be separable.This is the idea of the **Kernel Trick**. Even if this method allows a good adjustment capacity, the automatic improvement of the generalisation is not guaranteed.

2. **Manually engineer ϕ :** As mentioned earlier, this step is often carried out by the computer scientist himself. But this requires knowledge of the field and can be very time consuming.

3. **Learn it (Feature Learning) :** This is the option introduced by Deep Learning. The training process is not only devoted to the given task but also to ϕ learning. Considering the initial example, the model is

$$f(X; \Theta, W) = \phi(X, \Theta)^T \, W \tag{2.2}$$

Where parameters Θ are used to learn ϕ from a broad class of functions while parameters W map from $\phi(X)$ to the desired output.

2.2.3 Architectures of Deep Neural Network

This section will be devoted to the presentation of the structure of a deep neural network. The reader will also be familiarised with the terminology and the main components of a network. Finally, some classical architectures will be presented in more detail.

The schematic of a basic deep neural network is presented in figure 2.4. The most basic component of a network is called a neuron (or unit), each circle in the figure is a neuron. Each of them processes the input received in a different way depending on its type. It is also important to note that the input received depends on where in the network the specific neuron is located: if the neuron is in the input layer (i.e. at the very beginning of the model), the input will be a representation of raw data, otherwise the input will be the output of the previous neuron (i.e. the result of the computation performed by the previous neuron).

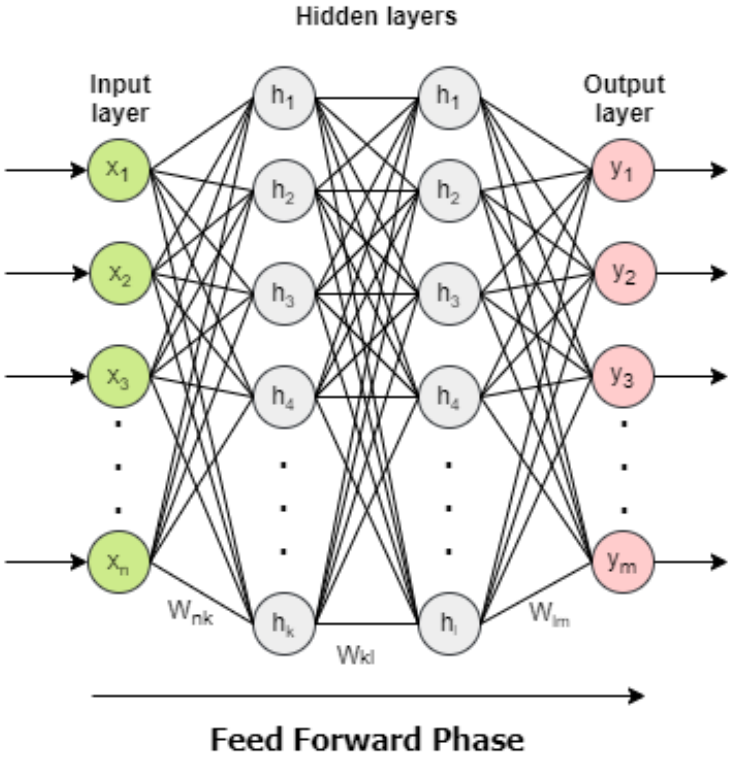

Figure 2.4: Diagram of a simple Deep Neural Network (DNN). The input layer (green) linearly maps the Input to prepare it for the first Hidden layer (grey). Then, the Hidden layers process the information in a non-linear way producing a hidden representation h_i. Finally, the Output layer (in red) transforms the output into a form appropriate for the given task (i.e. a class, a probability, a value, etc.).

Standard hidden units consist of an affine transformation $z = Wx + b$, where W is the weight and b is the bias (these are the two parameters associated with a neuron). To this affine transformation, a non-linear function (element by element) is applied, this is called **activation function**. Most hidden units are distinguished from each other by this specific function.

There are many different activation functions (for a more complete explanation and enumeration of the different activation functions, the reader may refer to [40]), but some of the more common ones can be distinguished:

1. **Sigmoid :** A very common activation function is the Logistic Sigmoid function, defined as :

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.3}$$

As shown in Figure 2.5, the Sigmoid function is characterized by saturation over most of its domain: on the one hand, when z is positive ($z >> 0$) then $\sigma(z) \to 1$, and on the other hand, $\sigma(z) \to 0$ when $z << 0$.

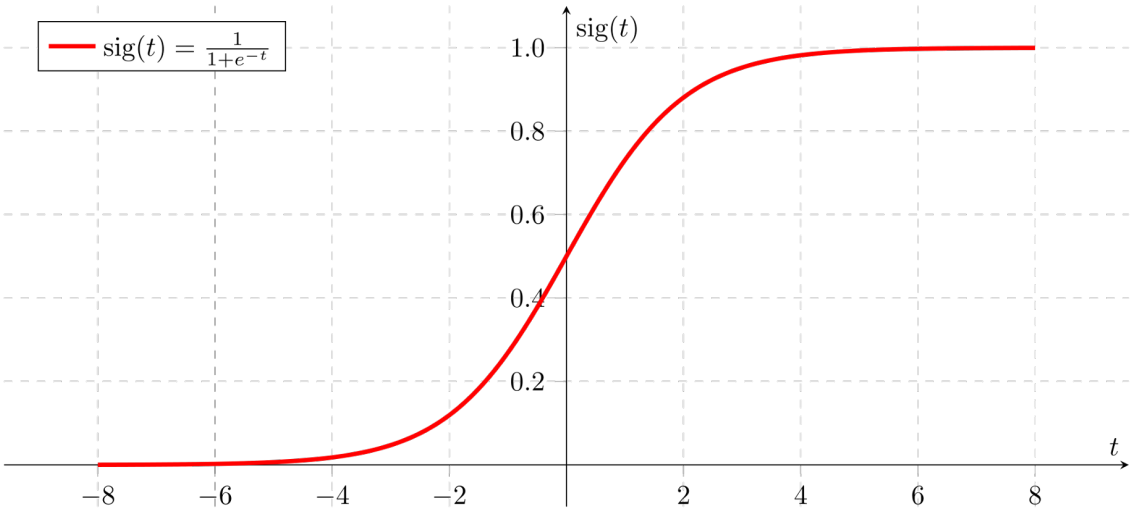

Figure 2.5: Sigmoid function.

2. **Hyperbolic Tangent :** This function is very similar, and related, to the previous one. It is defined as follows :

$$tanh(z) = 2\sigma(2z) - 1 \tag{2.4}$$

As shown in Figure 2.6, the Hyperbolic Tangent function is also saturated over most of its domain. Although it performs slightly better than the Sigmoid function, it still suffers from the same problems.

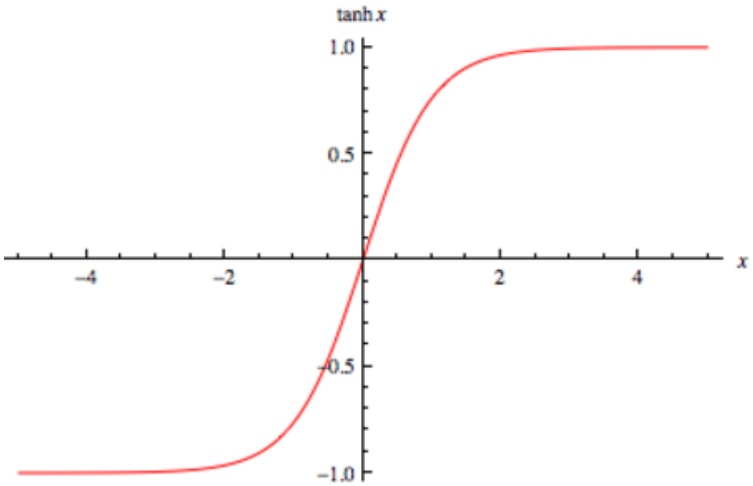

Figure 2.6: Hyperbolic Tangent function.

3. **ReLU : Rectified Lineat Units :** This activation function is a piece-wise linear function that overcomes the saturation problem introduced earlier. It is defined as follows :

$$g(z) = \max(0, z) \tag{2.5}$$

As shown in Figure 2.7, this function comes with several advantages :

- a piece-wise linear function is easy and very efficient to be optimized.
- its first derivative remains large when the units is active ($z > 0$).
- its second-order derivative is piece-wise constant.

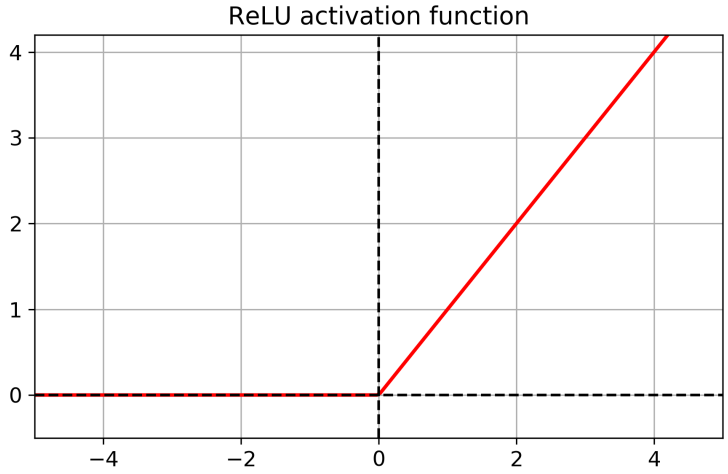

Figure 2.7: ReLU function.

There are many generalizations of ReLU, such as Leaky ReLU, Parametric ReLU or Maxout.

As can be seen in Figure 2.4, neurons are grouped, according to a property, in an organised structure called a layer. Each neuron in a layer is connected to (all, in the most basic case) the neurons in the previous and next layer. Whereas the neurons in a layer are not interconnected. As mentioned earlier, the first layer is called the input layer and its role is to linearly map the input data to the first hidden layer. The next layers are the hidden layers, which carry out the most important step, namely the processing of information. To do this, they exploit the non-linearity (given by the activation function) to learn appropriate features. Finally, at the end of the network is the output layer, which is responsible for ensuring that the shape of the result matches the given task. For this purpose, the activation function of the output units may be different from those of the hidden layers. The most common ones are :

1. **Linear Unit :** This is the simplest output unit. It simply produces an affine transformation of the input and can be used in a wide variety of tasks such as generating a conditional Gaussian distribution or in a regression task.

2. **Sigmoid :** This is the same activation function as above. It provides an appropriate output for a binary classification task.

3. **Softmax :** This activation function is used to output a probability distribution on a discrete variable. The Softmax function, defined as :

$$softmax(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{2.6}$$

outputs a Multinouilli distribution. This activation function is very often used to solve a multi-class classification problem.

The standard neural network model described so far and illustrated in Figure 2.4 is known as a **Fully Connected Neural Network**. It consists of several layers, where each neuron is connected to all neurons in the next layer. But there are other types of Neural Network architecture.

Convolutional Neural Network

This section is devoted to a detailed description of the convolutional neural network. As this is the architecture used in this work, the different components will be presented such as the convolution operation and the pooling layer.

Convolutional neural networks are based on the convolution operation. This operation, used in various fields such as engineering, physics or applied mathematics, can be defined as follows :

Definition 2.2.1 (Convolution Operation)
*A convolution is an operation, denoted *, defined over two functions f and g :*

$$\gamma(t) = (f * g)(t) = \int f(x)g(t-x)dx \tag{2.7}$$

Basically, the convolution operation combines two functions to produce another with certain properties. In particular, convolution can be seen as a description of how the shape of the first function is affected by the second.

In the particular domain of the Neural Network, the first term of the convolution (i.e. f in Eq. 2.7) is called the *Input* and the second (i.e. g in Eq. 2.7) is known as the *Kernel*. The output of this function is called ***Feature Map***. For the Neural Networks the convolution operation is defined as :

Definition 2.2.2 (Convolution Operation for Neural Networks)
The input X is a multidimensional array of data, and the kernel K is a multidimensional array of parameters that are learned by the network. The outputs of the convolution operation, called Feature Maps, are :

$$f(t) = (X * K)(t) = \sum_{x=-\infty}^{+\infty} X(x)K(t-x)dx \tag{2.8}$$

The reader has probably noticed that Eq. 2.7 is a discrete version of Eq. 2.8 which makes sense given that the data represented on a computer is discrete (e.g. an image is represented in pixels). Equation 2.7 is a 1D formula, used for data such as audio, but for 2D data such as images, this equation can be generalised as a 2D formula:

$$f(t,j) = (X * K)(t,j) = \sum_m \sum_n X(m,n)K(t-m, j-n) \tag{2.9}$$

In practice, the Convolutional Neural Network uses the convolution operation to learn feature maps through learning the kernel function (i.e. the model parameters). After a few convolutional layers (which perform the convolution operation), Fully Connected layers are added to perform the given task. A practical description can be found at [41], and Fig. 2.8 gives a diagram of this architecture.

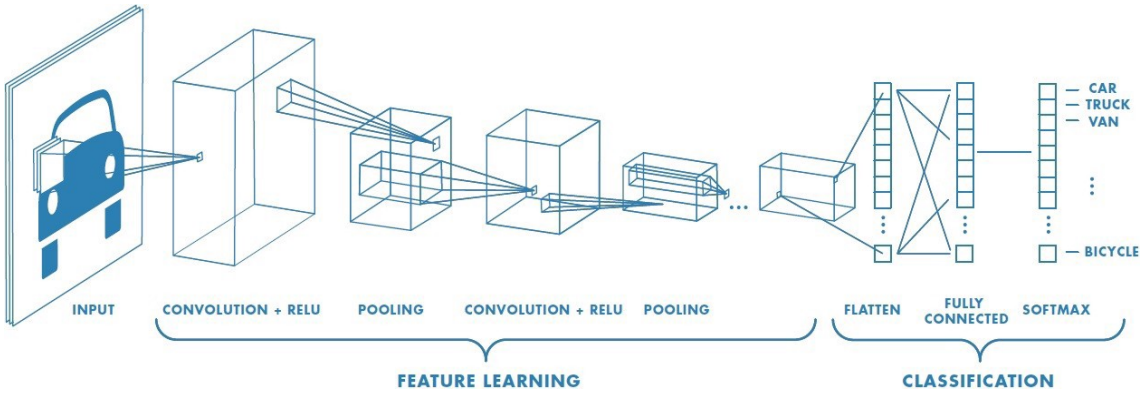

Figure 2.8: Example of a convolutional neural network scheme: the input image is processed by the convolutional part, which performs the **Feature Learning**. Then the fully connected component performs the **Classification** task. This figure is taken from [41].

As shown in Figure 2.8, common Convolutional Neural Network models are composed of a series of blocks containing a convolutional layer, a non-linear activation function and a pooling layer. The convolutional layer and the activation function have already been explained, so it is time to present the Pooling Layer. In fact, a pooling layer implements the pooling operation: it modifies the output of the previous layer by condensing some output values. Basically, there are two main pooling operations. Max-Pooling where a certain output region is replaced by the maximum value found in that region. Or Average-Pooling, where the replacement is based on the average of the values in the region under consideration. These two operations are illustrated in Fig. 2.9.

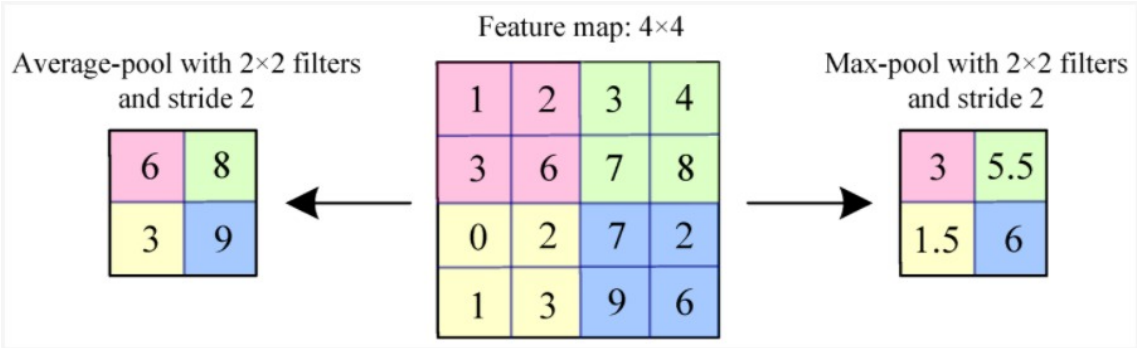

Figure 2.9: Example of a pooling operation. This figure is taken from [42]

Convolutional Neural Networks have many advantages. The first of these is given by the definition of the convolutional layer. Indeed, these layers are no longer fully connected, but have a **sparse connectivity** (described if Fig. 2.10). This characteristic results in a reduction in the total number of parameters to be learned, which saves memory and makes the learning process faster and more efficient.

Parameters can also be shared, which means that several functions can use the same set of parameters. In fact, each component of a kernel map is used at (almost) every

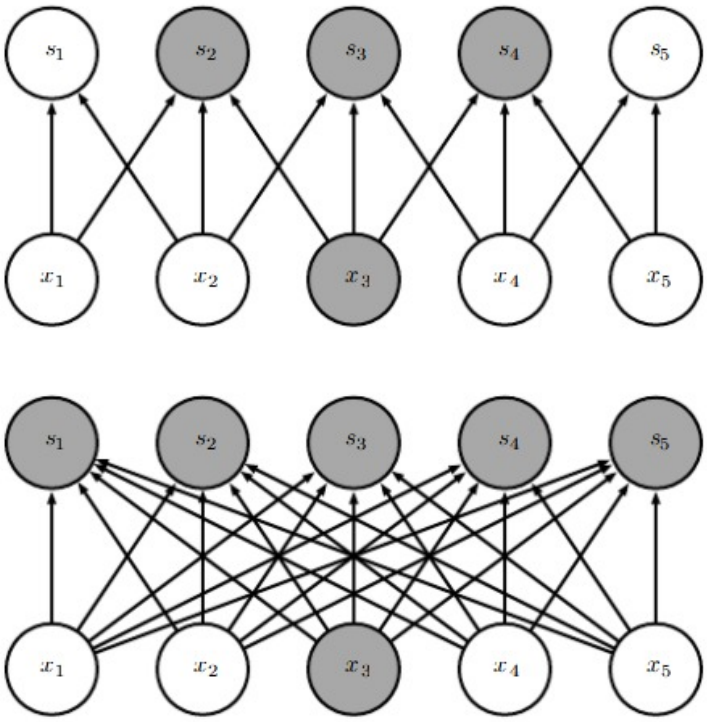

Figure 2.10: Illustration of sparse connectivity (top) compared to full connectivity (bottom). In grey, the output units affected by the input unit x_3 are shown at the top. On the fully connected part, all output units are affected by x_3 while only 3 are affected in the sparse version. This figure is taken from [43]

position of the input, so rather than having a separate and independent set of parameters for each location, only one set needs to be learned. This sharing reduces memory usage, increases efficiency and makes the convolutional layer equivalent to a translation (when the input changes, the output changes accordingly - in the same way).

Finally, another advantage that is important to note is given by the pooling operation. Indeed, this operation condenses the information which results in a reduction of the number of parameters. This operation creates weak translation invariance, that is to say if the input is translated by a small value, the value of the output of the pooling operation remains unchanged.

Other examples of architectures

This section is devoted to a brief presentation of other famous Deep Learning architectures. They will not be detailed, as they are not used in this study, but it is still interesting to mention them.

1. **Generative Models :** This type of model is an interesting variant of Deep Learning models, as it is a powerful way to model any kind of data distribution with an unsupervised learning approach. There are many generative models, of which two are particularly used: Variational Autoencoders (VAE) and Generative Adversarial Networks (GAN). Generally speaking, Autoencoders (AEs) are particular structures whose task is to reproduce the input supplied to them. Variational Autoencoders are still autoencoders, so they reproduce the input but apply a reparametrization

trick in the core layer, which generates an output similar, but not identical, to the input. Generative Adversarial Networks (GANs) have been created from Variational Autoencoders. Basically, they generate images by driving two subsystems called Generator and Discriminator. As the name suggests, the Generator generates an output based on examples. And the Discriminator discriminates by establishing whether the output is consistent with the training data.

2. **Recurrent Neural Networks (RNNs) :** Recurrent Neural Networks allow the specialisation of Neural Networks to process one-dimensional sequential data. These networks have characteristic connections that form directed graphs along a temporal sequence. This property makes it possible to process dynamic temporal data. These networks have an additional internal state, called memory, to process temporal inputs.

2.2.4 Optimization

So far, the description has focused on the feed-forward phase of the network 2.4. The input layer and the hidden layers produce learned features by processing information in a non-linear way, until the output layer provides the result in the desired form. But, a question remains: *for each hidden neuron, how are the famous parameters w and b chosen?*

Basically, these parameters are learned by Gradient Descent (GD) and Back-Propagation algorithms. This section is therefore devoted to the presentation of these algorithms and an overview of the main optimizers available for training Neural Networks.

Gradient Descent

Gradients are vector fields indicating the steepest directions in the functional landscape that drive the optimisation procedure towards a low loss value. The Gradient Descent (GD) method, introduced by Louis Augustin Cauchy in 1847 [44], is based on the observation that the gradient and the first-order Taylor expansion specify how a small change in the input influences the output. Formally, this observation can be written as follows :

$$f(x+\varepsilon) \approx f(x) + \varepsilon f'(x) \tag{2.10}$$

In simple terms, gradients are able to tell how to move among the values of x in order to minimise or maximise $y = f(x)$. From the directional derivative of a generic function f, differentiable (this method is based on differentiability) $f : \mathbb{R}^n \to \mathbb{R}$, there is the following limit :

$$\lim_{\alpha \to 0} \frac{\partial}{\partial \alpha} f(X + \alpha u) = u^T \ \Delta_x f(X) \tag{2.11}$$

where u is a unit vector and $\Delta_x f(x)$ denotes the gradient.

Therefore, f decreases when moving in the negative gradient position and increases when moving in the positive gradient direction. Thus, starting from a point, usually random, the Gradient Descent method proposes a new point :

$$X' = X - \varepsilon \Delta_x f(x) \tag{2.12}$$

Then the gradient descent method calculates the gradients in X' and iterates recursively. In Eq. 2.12, ε is a positive real number called **Learning Rate** and is a hyperparameter allowing the control of the new point update. This method is not guaranteed

to converge, but it does when the gradients tend to zero, i.e. when X' is updated by such a small amount that changes on f are no longer relevant.

In practice, the gradient descent method can be implemented in several forms. Here, the focus will be on the most famous of these, and its variants, Stochastic Gradient Descent (SGD).

- **Stochastic Gradient Descent [45]:** This method is based on a basic but crucial observation: calculating the exact expectation of the gradients of the cost function of a deep neural network can be very costly, as it would require an evaluation over the whole training process. Fortunately, it is possible to obtain a fair and unbiased estimate of the gradient by averaging the gradients over several mini-batches of m examples sampled from the training set. This gives the SGD method, which can be formally described as follows:

Algorithm 1 Stochastic Gradient Descent at the i-th iteration.

ε : the Learning Rate,
Θ : the initialized parameters,
L : the loss function,
while ! *stoppingcriterion* **do**
    Sample a mini-batch of m examples from the training set : $\{X_1, ..., X_m\}$ and the associated labels : $\{Y_1, ..., Y_m\}$
    Compute the gradient estimate : $\hat{g} \leftarrow \frac{1}{m} \Delta_\Theta \sum_k L(f(X_k, \Theta), Y_k)$
    Apply the update : $\Theta' \leftarrow \Theta - \varepsilon \hat{g}$

Thus, in this method, the Learning Rate becomes essential. Indeed, in order to compensate for the noise introduced by the random sampling of the mini-batches, the Learning Rate must be scheduled to decrease during the learning process. Otherwise, the algorithm would not converge.

- **Stochastic Gradient Descent with Momentum :** An improvement of the Gradient Descent based algorithms is the Momentum introduced by Boris T. Polyak in 1964 [46]. Since the convergence of the Stochastic Gradient Descent can be particularly slow, the addition of Momentum was proposed to speed up the learning process. In fact, the Momentum procedure keeps track of the past direction of the gradient and continues to move towards that direction. This procedure can be formally described as follows:

Algorithm 2 Stochastic Gradient Descent with Momentum at the i-th iteration.

ε : the Learning Rate,
Θ : the initialized parameters,
L : the loss function,
α : the momentum parameter,
while ! *stoppingcriterion* **do**
    Sample a mini-batch of m examples from the training set : $\{X_1, ..., X_m\}$ and the associated labels : $\{Y_1, ..., Y_m\}$
    Compute the gradient estimate : $\hat{g} \leftarrow \frac{1}{m} \Delta_\Theta \sum_k L(f(X_k, \Theta), Y_k)$
    Compute the velocity update : $v \leftarrow \alpha v - \varepsilon \hat{g}$
    Apply the update : $\Theta' \leftarrow \Theta + v$

- **Nesterov's accelerated gradient algorithm [47] :** The Momentum method can be further improved by this variant which further accelerates convergence.

Algorithm 3 Nesterov's accelerated gradient algorithm at the i-th iteration.

ε : the Learning Rate,
Θ : the initialized parameters,
L : the loss function,
α : the momentum parameter,
while ! *stoppingcriterion* **do**
    Sample a mini-batch of m examples from the training set : $\{X_1, ..., X_m\}$ and the associated labels : $\{Y_1, ..., Y_m\}$
    Compute the gradient estimate : $\hat{g} \leftarrow \frac{1}{m} \Delta_\Theta \sum_k L(f(X_k, \Theta + \alpha v), Y_k)$
    Compute the velocity update : $v \leftarrow \alpha v - \varepsilon \hat{g}$
    Apply the update : $\Theta' \leftarrow \Theta + v$

There are many alternatives to Gradient Descent based algorithms, but they will not be presented here, as for the time being Gradient Descent based algorithms remain one of the best options available.

Back-Propagation

In Fig. 2.4 the Forward Propagation phase is presented. Basically, the information passes through the input layer, then is elaborated in a non-linear way by the hidden layers and, in the output layer, produces a result whose quality corresponds to a scalar cost $L(\Theta)$ with L the loss function.

The reverse process is represented by Back-Propagation [48]. In this process, information flows from the loss function through the network to calculate the gradients. This will be essential for the iterative updating of the weights performed during the Gradient Descent optimisation. Back-propagation exploits simple mathematical tricks to perform an efficient and inexpensive gradient calculation.

One such mathematical trick is the concept of chain rule. Basically, when a function is a composition of other simpler functions $h(x) = f(g(x))$, this concept allows its derivatives to be calculated from the derivatives of the composite functions:

$$h'(x) = f'(g(x))g'(x) \tag{2.13}$$

By noting $y = f(u)$ and $u = g(x)$, the Eq. 2.13 can be rewritten in :

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} \tag{2.14}$$

This method is also generalizable beyond the scalar case and to a composition of more than two functions.

The Chain Rule is therefore a useful trick for calculating backpropagation since derivability and differentiability are key concepts in the calculation of gradients.

Formally, for a Deep Neural Network, the Feed Forward Phase can be described as in Algorithm 4.

Assuming the Feed Forward phase is already done and the fact that for each layer k, the gradients on the activation a are computed, the Back-Propagation can be defined as in algorithm 2.2.4.

Then, the gradients can be exploited by the Gradient Descent procedure to train the network.

Algorithm 4 Feed Forward Phase for a standard Deep Neural Network

$L(\hat{y}, y)$: the loss,
$\Omega(\Theta)$: a regularizer,
The input x : a single example,
The label y : label corresponding to the input,
l : the network depth,
$W^i, i \in \{1, ..., l\}$: the weights of the model,
$b^{(i)}, i \in \{1, ..., l\}$: the biases of the model

$h^{(0)} = x$
for $k = 1, ..., l$ **do**
    $a^{(k)} = b^{(k)} + W^{(k)} h^{(k-1)}$
    $h^{(k)} = f(a^{(k)})$
$\hat{y} = h^{(l)}$
$J = L(\hat{y}, y) + \lambda \Omega(\Theta)$

Algorithm 5 Back-Propagation for a standard Deep Neural Network

$L(\hat{y}, y)$: the loss,
$\Omega(\Theta)$: a regularizer,
The label y : label corresponding to the input,
l : the network depth,
$W^i, i \in \{1, ..., l\}$: the weights of the model,
$b^{(i)}, i \in \{1, ..., l\}$: the biases of the model

$g \leftarrow \Delta_{\hat{y}} J = \Delta_{\hat{y}} L(\hat{y}, y)$
for $k = 1, ..., l$ **do**
    Convert the gradient of the output layer into a gradient in the activation of the pre-nonlinearity :
    $g \leftarrow \Delta_{a^{(k)}} J = g \odot f'(a^{(k)})$
    Compute gradients on W and b :
    $\Delta_{b^{(k)}} J = g + \lambda \Delta_{b^{(k)}} \Omega(\Theta)$
    $\Delta_{W^{(k)}} J = g h^{(k-1)\mathbb{T}} + \lambda \Delta_{W^{(k)}} \Omega(\Theta)$
    Propagate the gradients with respect to the next lower level hidden layer's activation :
    $g \leftarrow \Delta_{h^{(k-1)}} J = W^{(k)\mathbb{T}} g$

Optimizers

Optimisation algorithms for Deep Neural Networks are generally called optimizers. The algorithms presented above are ones, however, in practice, the optimizers used are indeed based on Gradient Descent but with the particularity of adapting the Learning Rate during the training phase. The most famous and more used ones are :

1. **AdaGrad [49]:** it adapts the learning rates of all model parameters by scaling them, in inverse proportion, to the square root of the accumulation of all historical gradient values squared.

2. **RMSProp :** This is an adaptation of AdaGrad. The accumulation of the gradient becomes an exponentially weighted moving average. It is more efficient when dealing with highly non-convex functions, since only the last gradient traces are considered relevant for RMSProp, unlike the AdaGrad implementation where the relevance of the gradients is equal for the whole history.

3. **Adam [50] :** it is generally considered quite robust to its hyper-parameters choice, and it became one of the most used optimizers for Neural Networks. Compared to RMSProp, Adam adds the momentum term and implements bias corrections to take into account the initialization of first and second order moments.

2.2.5 Regularization

The optimal case of the learning process is to obtain a model that does not simply minimise the loss to the training set, but is able to handle the variability of the data. This is basically the role of regularization. Formally, regularization can be defined as [51]:

Definition 2.2.3 (Regularization)
Regularization is any modification to a learning process that is intended to reduce its generalization error but not (directly) affecting its training error.

Deep neural networks are complex and powerful tools. This makes them very prone to overfitting when no regularisation techniques are exploited. This is the reason why at least one regularizer is used when training deep neural networks.

Explicit Regularizers

Basically, a regularizer is defined as explicit when it directly affects the cost function to be optimised, typically by adding an extra term. In simple words, explicit regularizers impose constraints on the learning problems by directly influencing the loss function L. In the typical case, the loss becomes :

$$L = L + t_{reg} \tag{2.15}$$

where t_{reg} is the regularizing term. When minimising the loss function, the effect of the regularizer is already included. In general, t_{reg} is of the form :

$$t_{reg} = \lambda\Omega(\Theta) \tag{2.16}$$

where λ is the hyper-parameter responsible for the strength of the regularisation effect, and $\Omega(\Theta)$ is a function of the network parameters Θ.

In most cases, Ω only affects the network weights, leaving the biases unconstrained. This makes sense since bias fitting requires less data and effort than weight fitting. Therefore, leaving the biases unregularized does not introduce too much variance into the learning problem. This category of regularizer is called **Parameters Norm Penalties**. The two most common and widely used norm penalties are :

1. L^2 **norm penalty :** This is also known under the name of *Weight Decay*. This regularization strategy is driven by $\Omega(\Theta) = \frac{1}{2}\|w_2^2\|$. Thus, the loss function is :

$$L = L + \frac{1}{2}\lambda\|w_2^2\| \tag{2.17}$$

   The general effect of this regularisation is that the learning algorithm sees a higher variance in the input, causing it to reduce the weights on these features.

2. L^1 **norm penalty :** This regularisation technique is similar to the previous one, in fact they are special cases of a more general regularisation method called *p-norm*. Here, $\Omega(\Theta) = \frac{1}{2}\|w_1\|$. L^1 favours small weight norms and sparsity in the weight matrix.

Implicit Regularizers

Implicit regularization techniques do not directly affect the loss function. On the contrary, they indirectly influence the training phase in different ways. From the point of view of optimisation, these procedures are more training tricks than regularizers. These tricks act as regularizers because they induce certain effects that improve generalisation. Thus, in the sense of Def. 2.2.3, they can be fully considered as regularizers. It is interesting to note that sometimes implicit regularizers turn out to be equivalent to explicit ones, in terms of induced effects (e.g. the case of Dropout [52] [53]). In these cases, the theoretical boundary between explicit and implicit regularizers is blurred.

In any case, this section will ignore this vagueness and focus on the presentation of the most famous implicit regularizers :

1. **Dropout :** The Dropout technique [54] is one of the best known regularisation methods. It is very powerful in avoiding overfitting and is computationally cheap. This regularizer is based on a simple idea: during the training phase, each neuron is associated with a certain probability of being active or inactive. Technically speaking, a binary mask is randomly sampled for each mini-bath during training and applied to all input and hidden units of the network. Given this binary mask, the probability of being active is when it has a non-zero value and the probability of being inactive is when it has a zero value. Considering that the masks are sampled independently, these probabilities are independent for each neuron. An illustration of this technique is proposed in Fig. 2.11.

2. **Early stopping :** This regularization trick consists of monitoring the validation and learning curves, and stopping the learning when a sign of overfitting appears. Then, when the network is stopped, the previous weights (those present before the overfitting) are restored.

3. **Batch Normalization :** Typically, this trick is applied to hidden features and consists of normalising the features to a standard distribution. More than the regularisation effect, this technique is somewhat comparable to the effect of norm penalties.

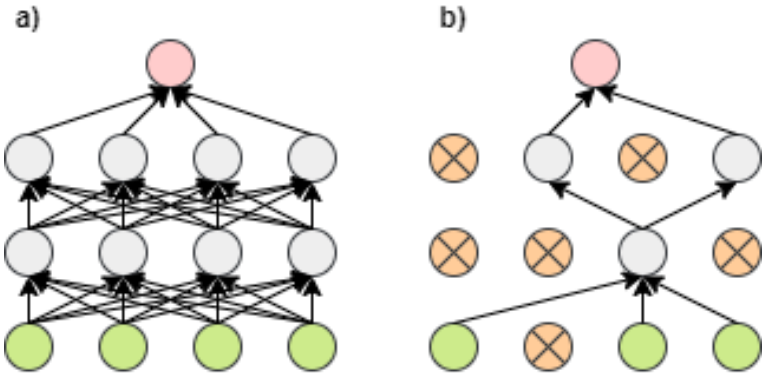

Figure 2.11: Illustration of the Dropout technique. In a), a simple Neural Network. In b), the same network after applying the Dropout mask.

Indeed, in [55], this technique is shown to speed up the learning process and make it more robust to variations in initializations and learning rates.

4. **Noise Injection :** In this technique, some random noise is injected into the input features, hence the input variances are increased. As with the previous trick, the effect of this is also comparable to that of norm penalties [56]. Some noise can also be added to the hidden features or weights themselves [57].

5. **Data augmentation :** As mentioned earlier, the best way to increase the generalisation performance of a model is to collect more data. However, this is not always possible and sometimes the data already available can be exploited further. Data Augmentation techniques are developed for this purpose. One of the best known examples of the success of data augmentation is image processing. For images, the techniques are based on modifying available images. The image is copied and then modified by randomly rotating angles, cropping, zooming, changing the resolution or applying filters.

2.2.6 Application to Raman Spectroscopy

Since few years, Deep Learning models are applied to Raman Spectroscopy. Up to now, the main works are on the **classification of minerals** using Convolutional Neural Networks [5] [58]. In [59] Deep Learning methods are used to identify **component of artificial chemical mixture**. Finally, in [60], a 1D ResNet is applied to classify **bacterial spectral data**.

However, the work presented here is based on [7], where Deep Learning methods are used in the COVID-19 dataset for diagnostic purposes. This work will be presented in more detail in the chapter 4.

3 Explainable Artificial Intelligence

Now that we know what a model (deep or not) is, this section is devoted to the explainability of these models. Indeed, the previous section showed that, even if the theory exists, there is no absolute rule for choosing the components of a model and, in some cases, there is no precise and obvious explanation to justify the success (or failure) of a component. In the medical context, more than in any other field, it is necessary to understand a model, without which it is impossible to have confidence in its choices. In this context, a decision can influence the life of the patient, so it is necessary to make sure that it is well founded.

3.1 Introduction

First of all, in order to understand the issues at stake in this field, it is interesting to look at its motivations and its history. To go even further, the reader can refer to [61].

3.1.1 Motivation

Artificial Intelligence has made great strides over the past decades, and Deep Learning methods have contributed greatly to this. However, they are generally treated as "black boxes" by users and developers.

In a world where Artificial Intelligence systems are intended to make decisions in complex areas, such as medical diagnosis, the people involved in these decisions need to trust the algorithm. To this end, it is necessary to make it more explainable and transparent. This is why "Explainable AI" field has received increasing attention in recent years. More than adding confidence and credibility to a model, explaining it "enables the non-expert user to learn from it" [62].

3.1.2 History

Although interest in this area is fairly recent, this does not mean that it is a new subject. Indeed, in the late seventies and early eighties, some experts already proposed a rule-based explanation of their results [62] [63]. This early mention proves that, since the beginning of artificial intelligence research, scientists have stressed the importance of the explainability of intelligent systems : "A computer program that models an expert in a given domain is more likely to be accepted by experts in that domain, and by non-experts seeking its advice, if the system can explain its actions" [62]. Early intelligent systems were based on rules and knowledge, formulated by human experts, and therefore had to be easy to interpret and understand. A typical "easy" structure to explain is the decision tree (illustrated in Fig. 3.1).

Despite this early interest, Explainable AI has become a research topic of its own in the context of Deep Learning. Indeed, the architectures presented in the previous section cannot be explained either by the network itself, or by an external explainer, or even by its developer. This is why Deep Neural Networks are generally considered as black boxes. In general, the explainability of a model is inverse to its prediction accuracy [64], that is

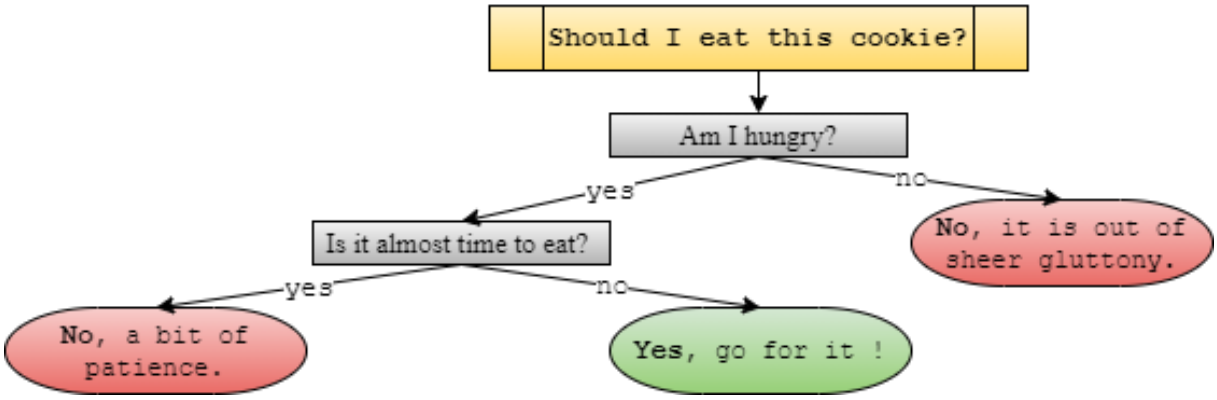

Figure 3.1: Illustration of a simple decision tree. Starting at the top and going down level by level, it gives an answer (in red or green) to the question (in yellow).

to say. the higher the prediction accuracy, the lower the explainability of the model. As expected, Fig. 3.2 shows that Decision Trees have a high degree of explainability but a lower prediction accuracy. In contrast, deep learning models have better predictive ability but low explainability. This inverse proportion explains why the field of explainable AI is growing at the same time as deep learning techniques.

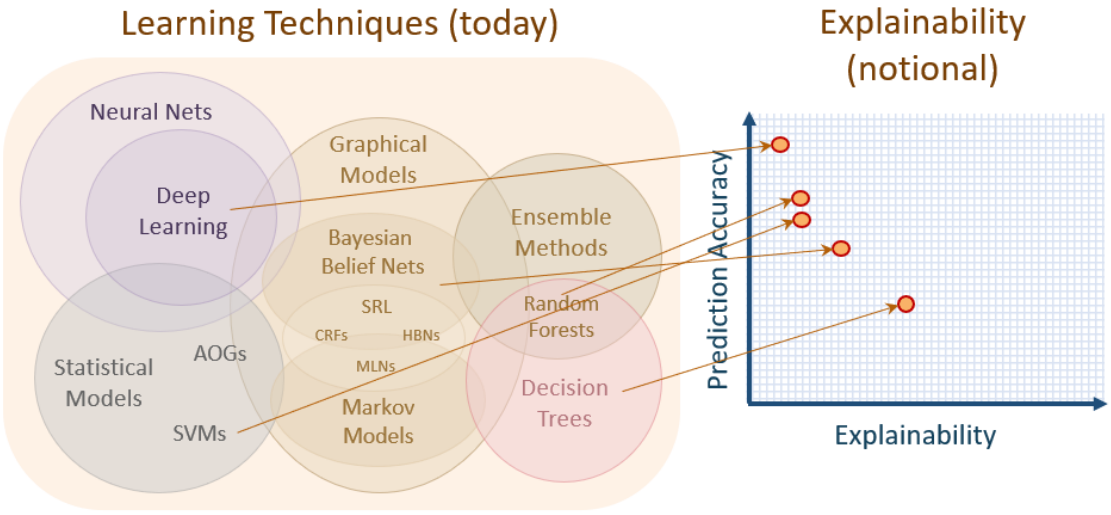

Figure 3.2: Illustration of the relationship between the accuracy of a model and its explainability. Image from [64].

3.2 Main current approaches

There are two main areas of work in Explainable AI, a detailed description of this categorisation can be found in [65]:

- **Transparency design :** The objective here is to reveal how a model works, from the developers' point of view. It tries to :

  1. understand model structure

2. understand single component
3. understand training algorithm

- **Post-hoc explanation :** It takes the user's point of view and tries to explain why and how a result is deduced. To do this, it tries to :

1. give analytic statement
2. give visualizations
3. give explanation by example

In the particular field of Deep Neural Networks, there are three main approaches (illustrated in the box) [61] :

1. **Transparency (a) :** The idea is to make the parts of the network transparent. Given an input, the network gives a corresponding output and the mapping function f. Thus, some techniques propose to use them to obtain information about how the network performs its task.

2. **Semantics from components (b) :** If a neuron is often activated by a certain part of the input data, this has a meaning. Based on this observation, the idea is to extract this information from the trained network and draw conclusions.

3. **Generation of explanation (c) :** This approach is the most easily understood by a user. Indeed, the idea is to use the network to produce a human-readable textual explanation of the underlying reason for a certain decision.

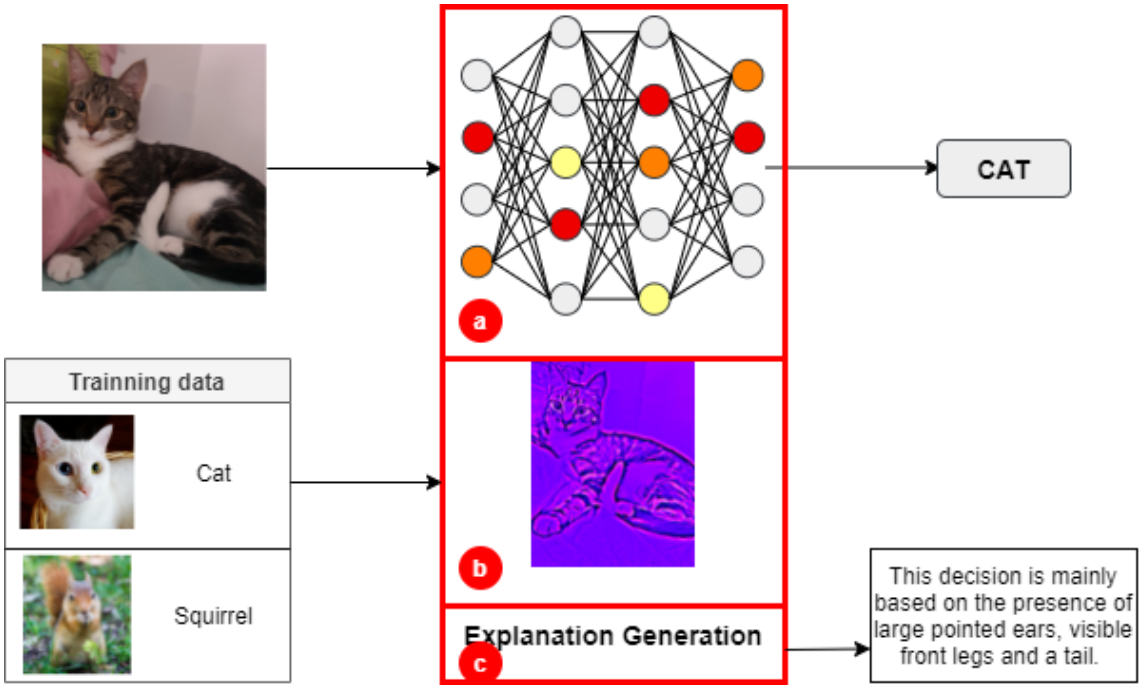

Figure 3.3: Illustration of the three main approaches to explaining a Deep Neural Network, indicated by red boxes where (a) represents the transparency approach, color of the neuron indicates the activation status, (b) represents component understanding and (c) explanation generation.

3.2.1 Transparency of Deep Neural Networks

To provide transparency in a Deep Neural Network, there are two popular techniques called Sensitivity Analysis (SA) [66] [67] and Layer-wise Relevance Propagation (LRP) [68] [67]. Both techniques aim to explain the individual prediction in terms of the input variable.

The **Sensitivity Analysis** method explains a prediction on the basis of the locally evaluated gradient of the model. The importance of each input variable i, calculated according to equation 3.1, is based on the fact that the most relevant input features are those to which the output is most sensitive. The Sensitivity Analysis does not explain the function $f(x)$ itself, but rather its variation.

$$R_i = \| \frac{\partial}{\partial x_i} f(x) \| \tag{3.1}$$

Basically, a heatmap computed with Sensitivity Analysis indicates which pixels need to be modified to make the image look more (or less) like the predicted class, from the perspective of the deep neural network. An example is presented in Fig. 3.4.

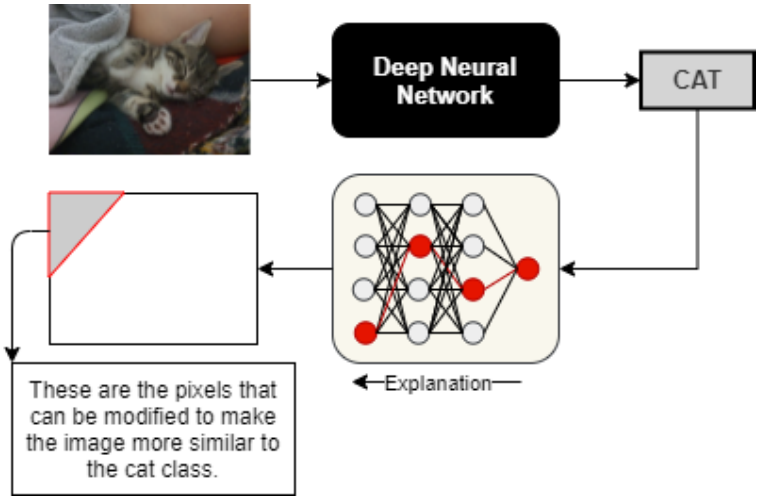

Figure 3.4: Diagram of the Sensitivity Analysis. The input image is correctly classified as a cat. Using Sensitivity Analysis, a heatmap is obtained where the modifiable pixels are highlighted.

Layer-wise Relevance Propagation decomposes Deep Neural Network predictions in terms of input variables. Compared to sensitivity analysis, this method explains the predictions in terms of the state of maximum uncertainty, which means that it identifies the input features that are critical to the prediction. An exemple is presented in Fig. 3.5.

Mathematically, this method redistributes the prediction $f(x)$ backwards, according to a local redistribution rule, until it assigns a relevance score R_i to each input variable. This relevance score determines the importance of the variable. The redistribution process is based on the property of *relevance conservation*, formally defined in Eq. 3.2. In simple terms, this property means that at each step of the redistribution process (in this method, this is equivalent to saying at each layer of a Deep Neural Network), the total amount of relevance (i.e. the prediction $f(x)$) is conserved.

$$\sum_i R_i = ... = \sum_j R_j = \sum_k R_k = ... = f(x) \tag{3.2}$$

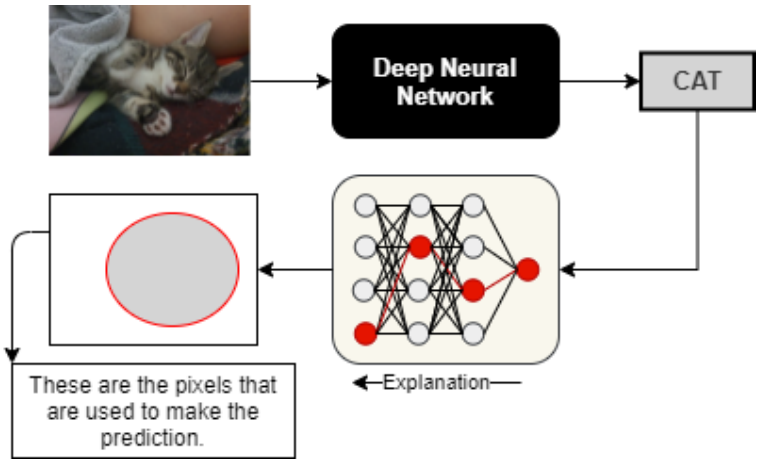

Figure 3.5: Diagram of the Layer-wise Relevance Propagation. The input image is correctly classified as a cat. Using Layer-wise Relevance Propagation, a heatmap is obtained where the pixels used for classification are highlighted.

Thus, in contrast to Sensitivity Analysis, the Layer-wise Relevance Propagation actually breaks down and explains the f function.

3.2.2 Semantic from Deep Neural Networks

To explain the idea of learning semantic from a Deep Neural Network, the "explanatory graph" method [69] is a perfect example.This technique proposes to learn a graphical model, the explanatory graph (represented in Fig. 3.6), which reveals the hierarchy of knowledge hidden in a pretrained Convolutional Neural Network.

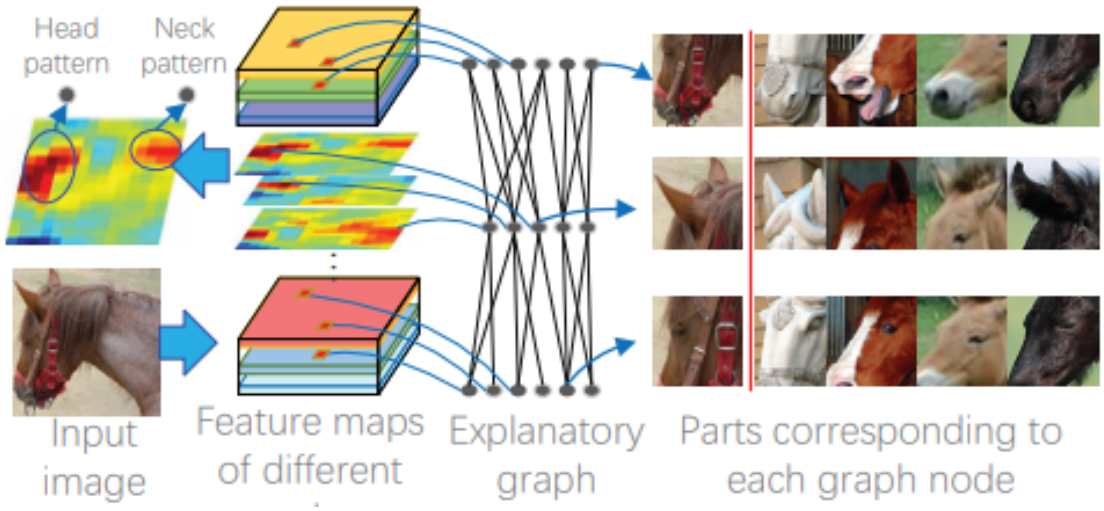

Figure 3.6: An explanatory graph represents the hierarchy of hidden knowledge in the convolution layers. Each filter of a pretrained CNN can be activated by different parts of objects. Here, the patterns of each filter are disentangled to clarify the knowledge representation. This figure is taken from [69].

The resulting explanatory graph is a coherent and meaningful representation that filters out noisy activations, disentangles reliable patterns from each filter, and encodes

co-activation logics and spatial relationships between patterns. However, the authors conclude their paper by pointing out that: "the explanatory graph is still a rough representation of the CNN, rather than an accurate reconstruction of the CNN knowledge" [69].

In a further work [70], the authors proposed a method to modify traditional Convolutional Neural Networks into interpretable networks, in order to further clarify the representation of hidden knowledge in the network. This technique works with a loss, designed specifically for this task, which pushes a filter in the high convolutional layers towards the representation of a part of an object. It is interesting to note that this method works without adding annotations to the data.

3.2.3 Generation of explanation

The techniques described above were intended for developers, but this one is clearly designed for the end user. Indeed, explaining a decision clearly to an end user can be as important as the decision itself, in the case of a medical diagnosis for example.

In [71], a technique is proposed to provide visual explanations of a visual classifier. It focuses on the discriminative properties of the visible object and jointly predicts a class label and explains why the predicted label is appropriate for the image. The assumption of this work is that visual explanations must be both class discriminative and image relevant. As shown in Fig. 3.7, the generated explanation is distinct from a description, which is a sentence based solely on the image description, and distinct from a definition, which provides a sentence based on the class information.

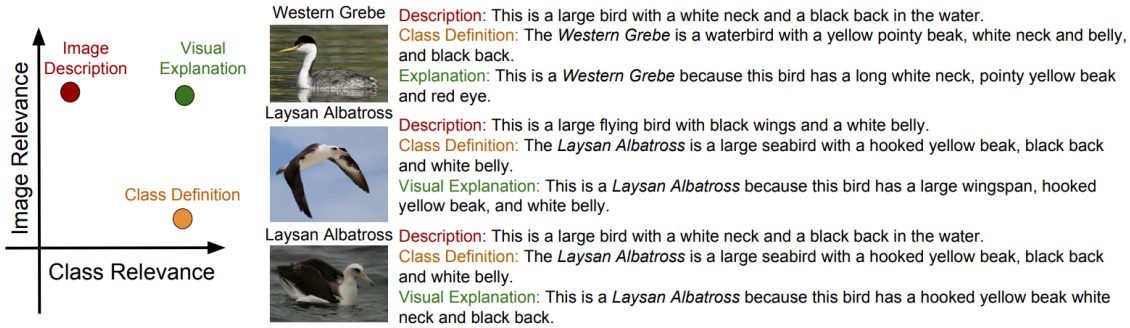

Figure 3.7: The visual explanations (green) are both relevant to the image and to the class. In contrast, the descriptions (in red) are mostly related to the image and the definitions (in yellow) are mainly related to the class. This figure is taken from [71].

On a more technical level, the input data is fed into a fine-grained, deep recognition pipeline to extract the nuanced details of the image and classify it. The features and label are then fed into a Long and Short Term Memory Model (LSTM - a recurrent network) to produce a sequence of words.

3.3 Application to Convolutional Neural Networks

Given that this work is carried out with a Convolutional Neural Network, this section is devoted to an overview of explainability techniques for this architecture. For a more in-depth look at these techniques, the reader can refer to [72].

Current studies on the understanding of convolutional neural networks can be divided into five main research directions:

1. **Visualization of representation in intermediate network layers :** These methods consist of synthesising the image that maximises the score of a given neuron or inverting the feature maps of a convolutional layer with respect to the input image.

2. **Diagnosis of representations :** This area of research is devoted to either diagnosing the feature space of a Convolutional Neural Network for different categories of objects, or to discovering potential representation flaw in the convolutional layers.

3. **Disentanglement of the "mixture of patterns" encoded in each filter :** The main objective of this area is to disentangle the complex representations in the convolutional layers and to transform the network representations into interpretable graphs.

4. **Building explainable models :** As the title says, these methods consist of building interpretable Convolutional Neural Networks. This method will not be discussed further as it has already been presented in section 3.2.2 - Semantic from Deep Neural Networks, with the work proposed in [70].

5. **Middle-to-end learning at the semantic level via human-computer interaction :** The semantic disentanglement of Convolutional Neural Network representations can enable "middle to end" learning with little supervision.

3.3.1 Visualization of representations

Visualization of filters in a Convolutional Neural Network is the most direct way to explore the visual patterns hidden in a neuron. For this reason, different types of visualisation methods were developed during the years :

- **Gradient-based methods :** These methods constitute the mainstream of network visualisation [73] [74] [75] [76]. They calculate the gradients of the score of a given neuron with respect to the input image. They then use the gradients to estimate the appearance of the image that maximises the neuron's score.

- **Up-convolutional net technique :** This method [77] inverts feature maps to images. This can be seen as a tool that indirectly illustrates the appearance of the image corresponding to a feature map, although, compared to gradient-based methods, up-convolutional nets cannot mathematically guarantee that the visualisation result exactly reflects the actual representation. In [78], an additional prior is introduced to control the semantic meaning of the synthesized image.

- **Compute the image-resolution receptive field :** This method [79] accurately compute the image-resolution receptive field of neural activations at in a feature map. The actual receptive field of neural activations is smaller than the theoretical field calculated using the filter size. This accurate estimation helps people to understand the representation of a filter.

3.3.2 Diagnosis of representations

Some methods go beyond visualization. By diagnosing Convolutional Neural Networks representations, it is possible to gain a deeper understanding of the features encoded in them. As in the previous point, different types of methods have emerged :

- **Analyze Convolutional Neural Network features from a global view :** This type of method may contain several different techniques. For example, in [80], the semantic meanings of each filter are explored. In [81], the trasferability of filter representations in intermediate convolution layers was analysed. Or, in [82] and [83], the distributions of features of different categories in the feature space of a pretrained network are calculated.

- **Extract relevant image regions for classification :** These methods are similar to visualisation in that they extract regions of the image that contribute directly to the network output for a label in order to explain the CNN representations of that label. Of course, different techniques are used to determine these regions. The methods presented in [84] and [85] propagate the gradients of the feature maps with respect to the final loss to the image plane to estimate the image regions. The LIME model in [86] extracts regions of the image that are highly sensitive to the network output. Some methods, proposed by [87], [88] and [89], visualise the regions of the input image that contribute most to the decision making process. Basically, these studies list the important objects detected in the images as the explanation for the output responses.

- **Estimation of vulnerable points in the feature space :** This is also a popular approach. Different techniques have been developed, for example in [90], [91] and [80], to calculate adversarial samples for a Convolutional Neural Network. In other words, these studies aim to estimate the minimum noisy perturbation of the input image that can change the final prediction.

- **Refine network representation :** As usual, different techniques can be found. For example, given a pretrained CNN for object classification, [92] proposes a method to discover the blind spots of knowledge (i.e. unknown patterns) in the network in a weakly supervised way. This method groups all sample points in the entire feature space into thousands of pseudo-categories. This method assumes that a well-learned network would use the subspace of each pseudo-category to exclusively represent a subset of a specific object class. In another case, [93] proposes to use the logical rules of natural languages to construct a distillation loss to supervise the distillation of neural network knowledge, in order to obtain more meaningful network representations.

3.3.3 Disentanglement of representations

To disentangle the representations of a Convolutional Neural Network, there are two main strategies: disentanglement into explanatory graphs or into decision trees. Since the first option has already been presented in a previous section (3.2.2 - Semantic from Deep Neural Networks), only the second is presented here.

In [94], a decision tree is proposed to represent the decision modes in fully connected layers. Thus, the decision tree is used to quantitatively explain the logic of each prediction,

which means that, given an input image, the decision tree indicates which filters in a convolutional layer are used for prediction and to what extent they contribute to the prediction. This process, illustrated in Fig. 3.8, uses the [70] method (presented in the previous section) to disentangle the representations of the filters in the upper convolutional layers.Thus, this method allows, using the decision tree, to understand which parts of the object are used to make the prediction.

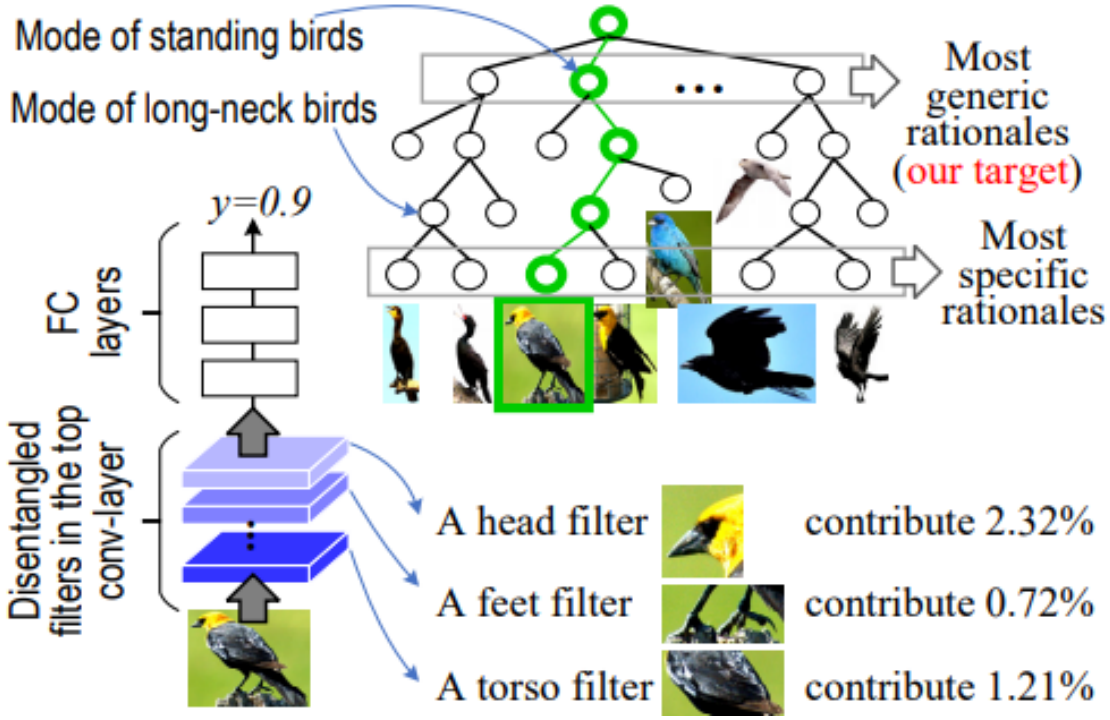

Figure 3.8: Illustration of the process of disentangling Convolutional Neural Network representations into decision trees. A CNN is learned for object classification with untangled representations in the top convolutional layer, where each filter represents a part of the object. Given an input image, a parse tree (green lines) is derived from the decision tree to semantically and quantitatively explain which parts of the object are used for prediction and to what extent they contribute to the prediction. This figure is taken from [94].

3.3.4 Middle-to-end learning

With an interpretable network representation, it is possible to enable middle-to-end model learning at the semantic level without heavy supervision. This is done in [95]. Based on a semantics defined in a previous paper, a method for using active question and answer to semantize the neural patterns in the convolutional layers of a pretrained model and to build a hierarchical object understanding model is proposed.

3.4 Application to Raman Spectroscopy

As this study is dedicated to the diagnosis of Covid using Raman spectra of saliva samples, this section is devoted to a more in-depth overview of the model-explainable

technique working with spectral data. In general, spectral data is processed using a Convolutional Neural Network, which was originally created to process images, so the convolutional layers are usually two-dimensional. But with spectral data, the convolutional layers used are uni-dimensional. Therefore, the techniques mentioned in the previous section are not necessarily applicable to the Covid case. Fortunately, some work has already been done in this area. For example, in [96], a Genetic Programming method is proposed to understand what features are important in Raman spectra. In [97], it is proposed to use the hidden information in the last Fully Connected (FC) layer of a Convolutional Neural Network to calculate the importance of each feature of a Raman spectrum.

Although, as in the previous section, there are many different techniques, it is the **Class Activation Mapping (CAM)** method and its variants that will be considered here. Indeed, this method is at the heart of the work presented in this thesis, so it is necessary to give a detailed description.

3.4.1 Class Activation Mapping (CAM)

First, it is necessary to define the CAM method, which was originally designed for Convolutional Neural Networks working with images [98].

This approach locates the discriminating image regions in the image analysis by the weighted sum of the feature maps in the last convolutional layer. In simple terms, the class activation map for a particular category indicates the image regions used by the network to identify that category. The idea is to perform global average pooling (GAP) on the convolutional feature maps and use them as features for an output layer. Then the importance of the image region is obtained by projecting back the weight of the output layer onto the convolutional feature maps. This process is illustrated in Fig. 3.9.

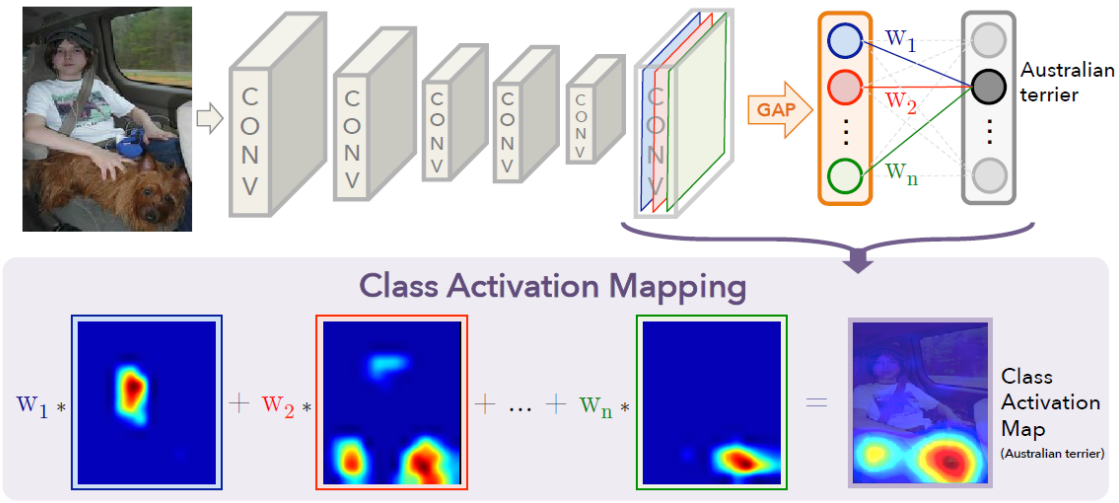

Figure 3.9: The predicted class is mapped to the previous convolutional layer to generate class activation maps, which highlight class-specific regions. This figure is taken from [98].

Mathematically, given an image, $f_k(x, y)$ is the activation of neuron k in the last convolution layer at spatial location (x, y) and the result of performing Global Average Pooling is :

$$F^k = \frac{1}{Z} \sum_{x,y} f_k(x, y) \tag{3.3}$$

where Z is the number of pixels in the feature map (or $Z = \sum_i \sum_j 1$). For simplicity, the output function is a softmax. Thus, for a given class c and denoting by w_k^c the weight corresponding to the class c for the neuron k, the input of the softmax is :

$$S_c = \sum_k w_k^c F^k \tag{3.4}$$

By adding the definition of F^k in Eq. 3.4, it becomes :

$$\begin{aligned} S_c &= \frac{1}{Z} \sum_k w_k^c \sum_{x,y} f_k(x,y) \\ &= \frac{1}{Z} \sum_{x,y} \sum_k w_k^c f_k(x,y) \end{aligned} \tag{3.5}$$

Thus, the class activation map for class c M_c is defined, each spatial element (x, y), by :

Definition 3.4.1 (Class Activation Map)

$$M_c(x,y) = \sum_k w_k^c f_k(x,y)$$

directly indicates the importance of the activation at the spatial point (x, y) which leads to the classification of an image in the class c.

Then, by upsampling the class activation map to the size of the input image, the image regions most relevant to the particular class can be identified.

3.4.2 Class Activation Mapping for spectral data

The Class Activation Mapping method has already been applied to spectral data in [99]. Logically, the formulas presented above must be modified in order to apply them to spectral data. Thus, with this type of data, the output of the Global Average Pooling F^k becomes :

$$F^k = \frac{1}{N} \sum_{x=1}^{N} f_k(x) \tag{3.6}$$

where $f_k(x)$ is the activation of neuron k in the last convolution layer at spatial location x and N is the length of the k-th feature map.
Using the same thinking as for the base Class Activation Mapping, the softmax input is, with M the total number of feature maps :

$$\begin{aligned} S_c = \sum_{k=1}^{M} w_k^c F^k &= \frac{1}{N} \sum_{k=1}^{M} w_k^c \sum_{x=1}^{N} f_k(x) \\ &= \frac{1}{N} \sum_{x=1}^{N} \sum_{k=1}^{M} w_k^c f_k(x) \end{aligned} \tag{3.7}$$

Using an oversampling approach called second-order spline interpolation, the feature maps can be back-transformed into the input spectral space, resulting in an oversampled vector P_k. This approach relies on different quadratic curves to fit every two adjacent

points with the same slope to the interior data points to obtain a curve that fits all data points. Thus, the vector P_k has the same dimension as the raw spectrum and indicates the activation at each variable in the spectrum.
Given the above, the softmax entry can be rewritten as follows:

$$S_c = \frac{1}{N} \sum_{x=1}^{N} \sum_{k=1}^{M} w_k^c P_k(x) \tag{3.8}$$

Thus, the Class Activation Map for the spectral data is defined as follows:

Definition 3.4.2 (Class Activation Map for Spectral data)

$$M_c(x) = \sum_{k=1}^{M} w_k^c P_k(x)$$

The difference between the different samples is therefore due to the change in the input spectrum. As there are individual differences, the magnitude of the variable $M(x)$ is normalised to a range of 0-1 by min-max scaling before visualisation. The min-max scaling approach does not alter the relative CAM peaks, allowing a fair comparison for different spectra, regardless of individual differences.

3.4.3 Variant of the Class Activation Mapping method

In [85], two variants of the Class Activation Mapping are proposed on the images : the **Gradient Class Activation Mapping (Grad-CAM)** and the **Guided Gradient Class Activation Mapping (Guided Grad-CAM)**. These two techniques are illustrated in Fig. 3.10.

Gradient Class Activation Mapping is based on a simple observation: the basic Class Activation Mapping method requires a particular architecture. Indeed, the output of a Global Average Pooling layer is fed into a classification layer, such as softmax, to obtain a result. Thus, for some networks, it is necessary to modify and re-train them, which can be very time consuming. To enable the technique to be applied to different architectures, the Gradient Class Activation Mapping method proposes to use the gradient information flowing through the last convolutional layer of the Convolutional Neural Network to understand the importance of each variable for a decision.
To obtain the Grad-CAM value $L^c_{Grad_CAM} \in \mathbb{R}^{u \times v}$ of width u and height v for any class c, it is necessary first to calculate the gradient of the score for class c, before the softmax, y^c with respect to the feature maps A^k of a convolutional layer, mathematically, it is needed to compute $\frac{\partial y^c}{\partial A^k}$. Then, these return gradients are global average pooled to obtain the importance weight for a neuron k:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{3.9}$$

This weight represents a partial linearisation of the deep network downstream of A, and captures the importance of the feature map k for a target class c. Then a weighted combination of the forward activation maps is followed by a ReLU operation to obtain the Grad-CAM value.

$$L^c_{Grad_CAM} = ReLU(\sum_k \alpha_k^c A^k) \tag{3.10}$$

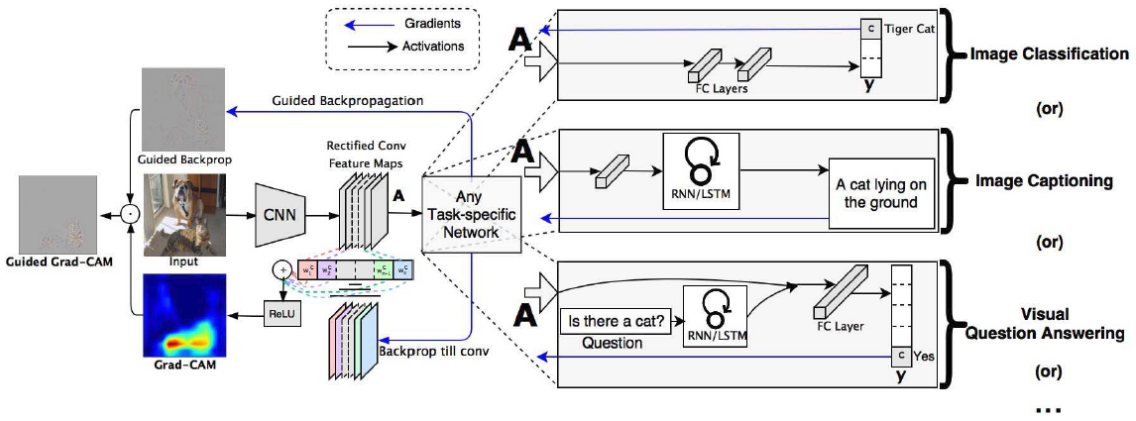

Figure 3.10: Overview of Grad-CAM and Guided Grad-CAM. Given an image and a class of interest as input, the image is propagated forward through the Convolutional Neural Network part of the model and then through taks-specific calculations to obtain a score for the class. The gradients are set to zero for all classes except the class under consideration, which is set to one.This signal is then back-propagated to the rectified convolutional feature maps of interest, which are combined to calculate the coarse Grad-CAM location (blue heatmap) that represents where the model looks to make the decision. Finally, the heatmap is multiplied point by point with guided backpropagation to obtain a guided Grad-CAM visualisation that is both high resolution and concept specific. This figure is taken from [85].

The ReLU operation is applied because the interest is in those features that have a positive influence on the class of interest, as features with a negative influence are likely to belong to other categories. It is important to note that Grad-CAM is a strict generalisation of CAM. Indeed, the application of Grad-CAM to the specific architecture needed for the CAM method gives $\alpha_k^c = w_k^c$.

Although Grad-CAM visualisations are discriminating and locate relevant regions of the image well, they do not have the ability to show fine-grained significance like gradient visualisation methods in pixel space (like Guided BackPropagation). The idea of guided Grad-CAM is to combine the best aspects of both. To do this, $L^c_{Grad-CAM}$ is first sampled at the resolution of the input image by bi-linear interpolation, and then the guided back-propagation and the oversampled Grad-CAM are multiplied point by point. Thus, this technique provides a visualisation that is both high resolution and class discriminating.

4 A Deep Learning model to diagnose Covid

This thesis is part of a larger study and therefore builds on the work previously carried out by members of the team. Thus, this part is devoted to the description and explanation of these in order to allow a better understanding of the situation in which this thesis begins. To do so, this section is based on [100] and on [101].

4.1 Dataset of Raman COVID-19 salivary fingerprint

4.1.1 Description of the dataset

Saliva samples, health records and clinical data were acquired at IRCCS Fondazione Don Carlo gnocchi ONLUS, Santa Maria Nascente Hospital, Milano, Italy and Centro Spalenza Hospital, Rovato, Italy between 16th April 2020 to July 2020. This dataset consists of 2408 spectra from 101 patients, described precisely in table A.1, divided into 3 classes :

- **Covid Positive :** This class groups together patients who are affected by Covid-19. It consists of 747 spectra from 30 patients.
- **Covid Negative :** This category is composed of patients cured of Covid-19. These patients were considered negativized after two consecutive tests with negative results. The data set shows 877 spectra from 37 patients in this case.
- **Healthy (CTRL) :** This classes is composed of 784 spectra belonging to 34 healthy patient.

Thus, the dataset used is pretty balanced.

4.1.2 Pre-processing of data

In the course of the work leading to the definition of the model, the team proposed a complete spectral data pre-processing pipe-line, based on [102]

This pipe-line, summarized in Figure 4.1, is composed of :

1. **Removal of the *outlier* spectra :** A spectrum is considered as *outlier* when is not informative for the purposes of classification. That is to say when during the extraction phase it undergoes some issues including saturation, laser misdirection, cosmic rays interference, and others: the resulting datum is not informative since the signal-to-noise ratio is typically too low. In the case of Covid, the team decided to use a standard outlier treatment, removing a spectrum when it has more than 10% null or completely saturated intensities. This led to the removal of 50 spectra.
2. **Realignment of the Raman shift axis :** Raman Spectra acquisition can happen in various moments and conditions: therefore, some acquisition errors and noise due to the calibration of the laser frequencies of the spectrometer can arise, resulting in

a misalignment of the Raman shift axis for distinct samples. In order to perform this realignment, a linear interpolation is applied to resample each spectrum.

3. **Substraction of the background signal :** The intensity of the aluminium substrate used for the SERS produces a background signal. As this signal is almost identical for each spectrum, it was simply subtracted from the original data.

4. **Removal of the background noise :** Molecular fluorescence is the optical emission from molecules that have been excited to higher energy levels by the absorption of electromagnetic radiation. Thus, during the acquisition of a Raman spectrum, the molecules centred by the laser beam produce molecular fluorescence, i.e. noise in the resulting spectra. The more complex the composition of the sample, the greater the fluorescence is likely to be. One of the most popular techniques for removing this noise is based on polynomial fitting: a polynomial function is fitted across a grid of points sampled from the spectrum, and the baseline effect is removed. For a simple molecular system, a low order, i.e. linear, fit may be sufficient. But for saliva samples, which are complex samples, a high-order polynomial fit is required. The team therefore proposed a baseline fitting with a sixth degree polynomial without smoothing the signal (in order to preserve the original signal and avoid loss of information).

5. **Despiking :** It is sometimes possible that the measurement process is influenced by cosmic rays: cosmic particles, such as highly energetic muons. These rays can interact with the instrument, producing anomalous peaks in a spectrum. As these anomalies can potentially negatively influence the subsequent analysis of the data, it is important to identify and remove them. To do this, the Whitaker-Hayes [103] algorithm is applied with a threshold set to 3.5 and a neighbourhood size set to 11.

6. **Intensity normalization :** In order to avoid numerical instability during the data analysis phase and to allow a homogeneous comparison between spectra acquired under different conditions (and possibly with different instruments), some normalisation techniques can be applied. In the scientific literature, there are many normalization techniques, but four of them are particularly used in Raman spectroscopy: Standard Normal Variate Normalization, Unit Vector Normalization (l_2), Min-Max Normalization and Max Normalization. To determine the best method, the team conducted experiments, the details of which can be found in [100] and [101], and finally decided to use the l_2 technique.

7. **Dimensionality reduction :** When data have a high dimensionality, it is usual to reduce it to avoid performance problems. To do this, the team applied principal component analysis (PCA), which consists of extracting a set of orthogonal components, and choose the first 100 components of the PCA to represent the data.

Although this pipeline is effective, it is not used in this work. Indeed, the preprocessing of the data takes some time and does not improve the result much [7].

4.1.3 Data Augmentation

Data augmentation, previously presented in Sec. 2.2.5, is a procedure that generates new synthetic data samples by applying variations and distortions to the original data,

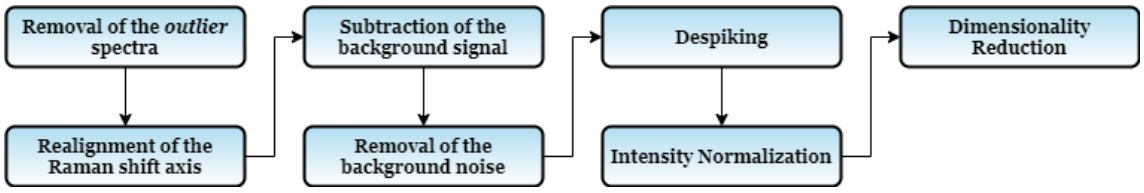

Figure 4.1: Summarized pre-processing pipe-line.

making the most of their intrinsic invariances. With data augmentation procedures, the generalization performance of models should improve while partially overcoming data sparsity. In [104], a technique is proposed to augment the spectral data. This technique consists of adding random offsets, random slope changes and random multiplications to the existing spectra in order to enlarge the data set. The various components of this augmentation are shown in Fig. 4.2.

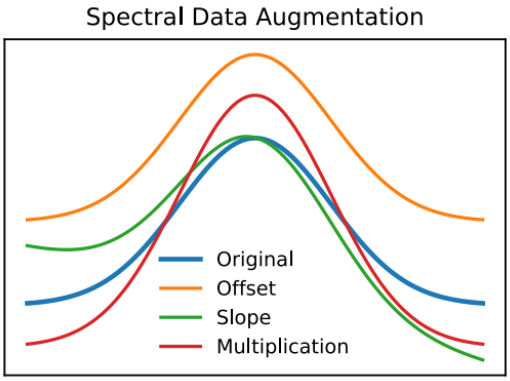

Figure 4.2: Presentation of the different components of the spectral data augmentation. The data augmentation is created with a randomly scaled contribution of offset, slope and multiplication. This figure is taken from [104].

It is important to note that the data augmentation is applied only to the training set.

4.1.4 Splitting the dataset

The dataset used in this study has two particularities that need to be taken into account. The first is the fact that it is a rather small dataset, consisting of 2408 spectra from only 101 patients, which represents an average of 24 spectra per patient. Even with the use of data augmentation, it is still quite small. The second is the fact that it is susceptible to data leakage, in that if some spectra belonging to a patient are in the training set and others in the test set, the classification is biased. Since two spectra belonging to the same patient are more similar to each other than to spectra from different individuals, the performance of the model tends to be more optimistic.

Leave-One-Patient-Out Cross-Validation is a technique that allows both of these particularities to be taken into account. Basically, the test set is composed of all the spectra belonging to a single patient (all the spectra belonging to the other patients are then in the training set). It is important to note that using this method results in 101 different divisions of the dataset, so it is necessary to train 101 different models, one for each division.

4.2 Model

4.2.1 Description of model

The model used in this study has been fully developed and optimised in a previous work done by the team, the details of its development can be found in [100]. This part is only dedicated to the description of the final optimised model obtained.

This model, illustrated in Fig. 4.3, is a Convolutional Neural Network designed for classification. It consists of three 1D convolutional layers with a ReLU activation function. The last two are followed by a MaxPooling operation, and after the first MaxPooling operation, there is a batch normalization. After the convolutional part of this model, there are three linear layers using LeakyReLU as the activation function, with heavy use of Dropout. Finally, a softmax is used as a classification function.

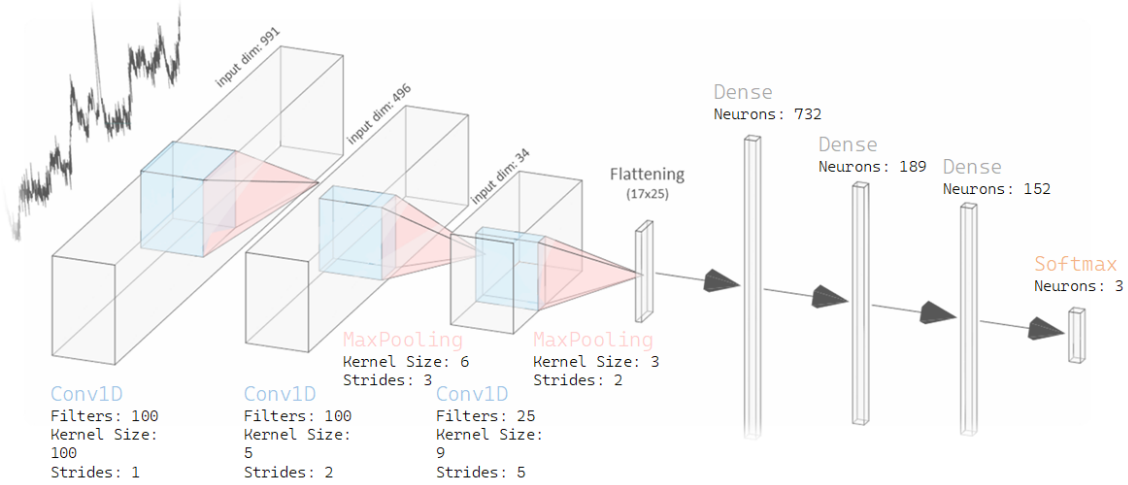

Figure 4.3: Description of each layer of the CNN used for this work. This figure is taken from [100].

This model is trained using an Adam optimiser and a cross-entropy loss. The original model was written using Keras, but to allow for more possibilities and to simplify the explainable part, it has been rewritten using the Pytorch library.

4.2.2 General remarks

As mentioned earlier, using the Leave-One-Patient-Out cross-validation technique, 101 train/test pairs are obtained. So, in practice, 101 models are created, each is trained with one of these pairs and then, when a new piece of data is tested, a completely new spectrum, each of the 101 models predicts a class, and the one that comes up most often is selected as the true one. For example, if 32 models say that this spectrum is a control spectrum, 15 say it is a Negative Covid and 54 predict that it is a Positive Covid, the final prediction is the Positive Covid.

Another interesting point to mention is that this prediction is done at the spectra level. Therefore, for a real-world diagnostic application, the goal is to have a prediction for a patient. As there are several spectra associated with each patient, each patient is classified according to the majority label assigned to his or her spectra.

4.2.3 Result of model

At the spectra level, the average accuracy obtained on the 101 models is 0.83, at the patient level it is 0.89. These results are shown in Fig. 4.4.
The confusion matrix also shows that Covid-negative patients are generally well classified, but Covid-positive and control patients seem more likely to be misclassified, especially by being confused with Covid-negative patients. The detailed results are presented in table A.2.

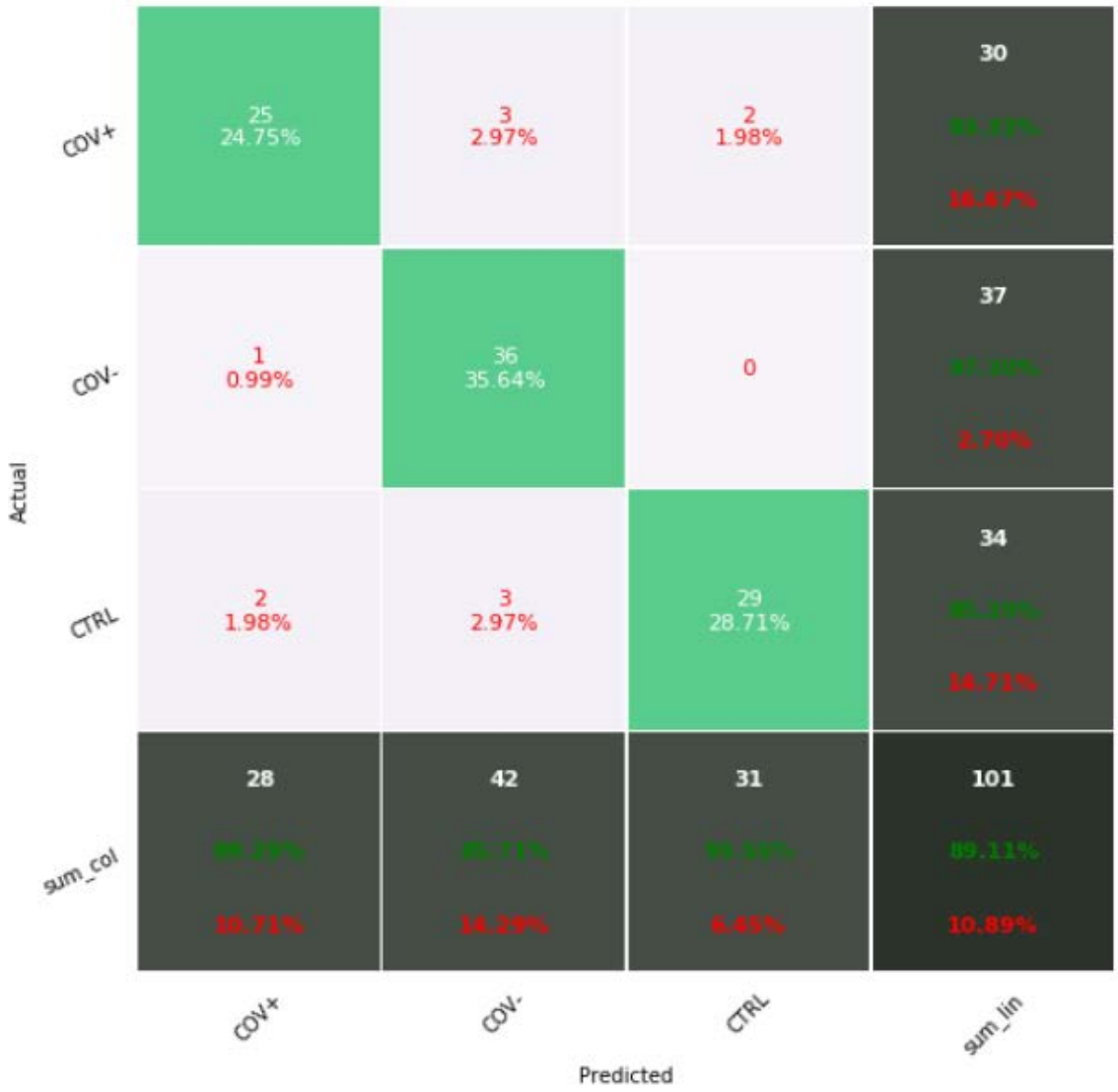

Figure 4.4: Confusion matrix of the results obtained (at patient-level) from the classification of the 101 patients of the Covid classification with the original Convolutional Neural Network. This figure is taken from [100].

After the transformation of the original model into a Pytorch model, the average accuracy, both at the spectra and at the patient level, is the same. But, as shown in Fig. 4.5 the confusion matrix is slightly different.

The previous observation on Covid Negative patients is still valid. However, the confusion matrix shows that, although Covid Positive and control patients are still more

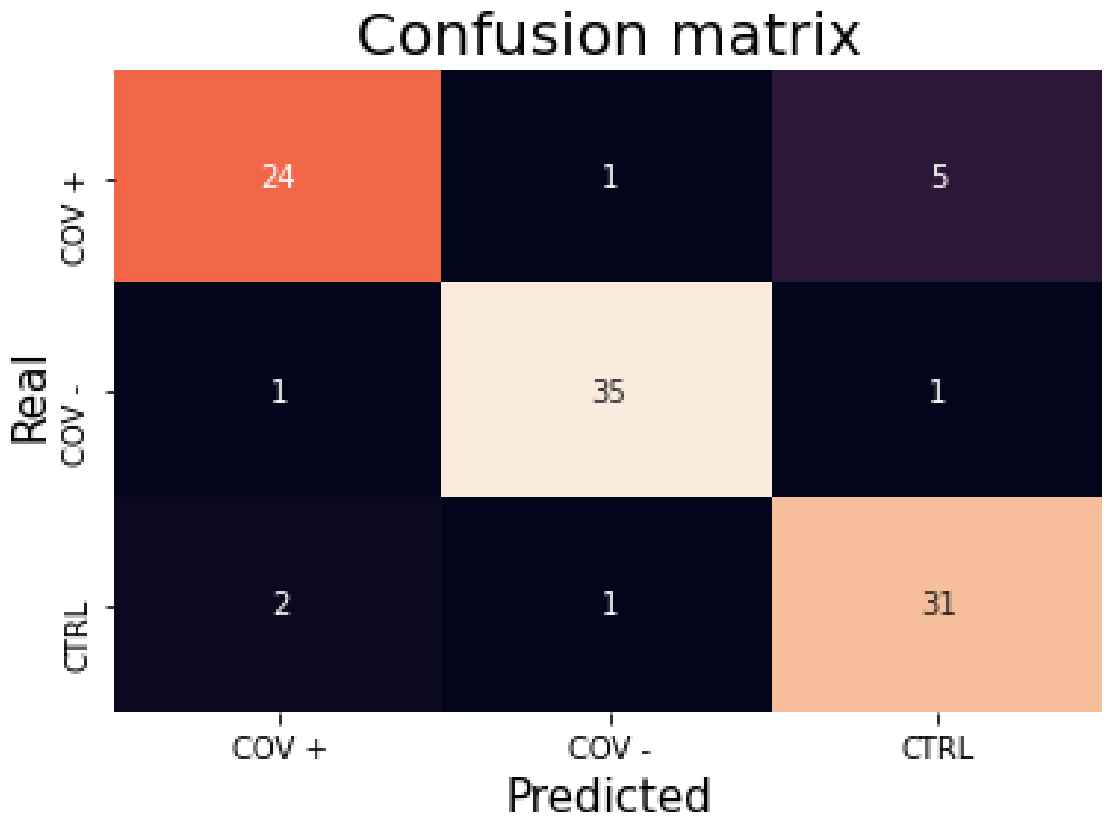

Figure 4.5: Confusion matrix of the results obtained (at patient-level) from the classification of the 101 patients of the Covid classification with the Pytorch model.

prone to misclassification, there are some differences. First of all, Covid Positive patients are more often misclassified than control patients. And, this time, the two classes are more likely to be confused with each other. This subtle instability in the model prediction confirms the need of a robust model explanation to address a proper use in clinical settings.

5 Stages of reflection

Now that all the key concepts are defined, this chapter is devoted to furthering the methodology and the reasoning used to complete this thesis. The final objective is to propose an efficient method to understand the classification performed by the Convolutional Neural Network on the Covid diagnostic task.

In the chapter 3, several methods are presented to explain such a network, but here the aim is to explain which of these methods is used, why and how it was chosen and how it is applied to the specific case study of Covid diagnosis.

5.1 Guideline

As the reader has already understood, the method chosen to achieve explainability of the model is called Class Activation Mapping [98]. A previous section (3.4) has fulfilled the role of describing and explaining this method, so it will not be the focus of this one. The purpose of this section is to complete the description by allowing the reader to better understand the thinking that went into bringing this method to light.

To be a good candidate, the chosen method must have certain characteristics. Indeed, considering that the overall project aims at developing an innovative, rapid and non-invasive screening methodology for the diagnosis of various diseases, the approach presented in this thesis must go beyond the understanding of the specific model created for the diagnosis of covid, it must be applicable to other models created for other diseases. In fact, the chosen method must be :

1. **Relevant for Convolutional Neural Networks :** As explained earlier, the convolutional architecture is specific and, therefore, not all explicable artificial intelligence techniques are applicable to it, or at least not yet. Thus, in order to find a good candidate, the search was carried out in the restricted domain of the explicable convolutional neural network. But, as shown in Chapter 3, there are many different techniques applicable to these networks. It is therefore necessary to restrict the search domain again to find a good approach.

2. **Usable on Raman spectra :** Of course, the model used is a Convolutional Neural Network but it is slightly different from the majority of them. Indeed, generally, these networks are two-dimensional and designed to process images. In the case of Raman spectra classification, the network is one-dimensional and works on spectral data. This is the second particularity that must be taken into account. Some techniques are not usable on spectral data because they use the specificity of an image, others are not easily transposable on spectral data. Thus, the search area for a good candidate should be narrowed down to techniques that have already been used on spectral data or that do not use the inherent specificities of image processing.

3. **Easily portable :** As mentioned a few lines above, the final aim of this work is to propose a method that can be used on other case studies. When one says other cases of study, one probably means slightly different models. Indeed, in [100],

where the model used for the Covid is defined, other cases are proposed such as Parkinson's disease, Amyotrophic Lateral Sclerosis or Alzheimer's disease. In each of these cases, the proposed network is slightly different, it is always a convolutional neural network, but the parameters and even sometimes the number of layers are different. These differences mean that the proposed method must be "portable". In computing, **portability** is defined as the ability to use the same software in different environments. Saying that the method must be portable means that the method must be usable on different models with a minimum of changes. Thus, a (very) good candidate must also satisfy this property.

4. **Preserve model :** In an ideal scenario, the method proposed in this thesis should preserve the performance of the model and be applicable on a pre-trained one. Basically, this means that the approach should not require any modification of the model. Indeed, making a modification to make it more explainable is problematic, it induces a re-training, which is time consuming, and, moreover, it may be irrelevant. Indeed, the objective is to explain the results obtained by a model, but by modifying it, the risk is to obtain different results and, even if the results remain unchanged, the explanation may not be appropriate. If, for example, the modification involves the removal of certain layers, their roles will not be taken into account in the explanation. Yet they must have a role. Thus, to be a "perfect" method, it must satisfy this property.

These characteristics form the guideline for this thesis, and this is what needs to be kept in mind in order to find a good approach.

5.2 Class Activation Mapping

By restricting the research area to a method suitable for Convolutional Neural Networks and spectral data, several methods were found. However, the one that seemed to be the most interesting was the Class Activation Mapping [98]. Indeed, this method has a number of advantages.

First of all, it provides a method to visualise the most important variables for classification. This is particularly interesting because this visual representation can allow a specialist to analyse the results and thus provide a better level of understanding of the classification. To achieve this visualisation, this method uses the last convolutional layer, which is also an advantage because, although not all Convolutional Neural Networks have exactly the same architecture and layers, they all have convolutional layers by definition. But one of the main strengths of this method is that it has already been applied to spectral data, and more particularly to Raman spectra [99]. This existing application confirms that this method is suitable for Raman spectra. But the spectra used are quite simple, such as the spectra of a bacterium. The question is therefore whether this method is effective for Raman spectra of more complex samples, such as saliva. In addition, the results of this application can be used as a basis to confirm the methodology used on Covid. Indeed, if it is correct, it will give the same results on the bacteria dataset as those presented in [99].

However, even if this method satisfies a number of the characteristics specified above, it has a major flaw. Indeed, this technique requires a particular architecture presented in Fig. 3.9. After the last convolutional layer, a Global Average Pooling layer is expected.

Unfortunately, the model used in this study is not built with this architecture. Thus, to apply this method, the network must be modified and re-trained.

5.3 Gradient Class Activation Mapping

This weakness is the reason why a variant of Class Activation Mapping, called Gradient Class Activation Mapping [85] has been introduced. Indeed, this method corrects this problem by being usable on all Convolutional Neural Networks.

This variant retains most of the advantages of Class Activation Mapping. Unfortunately, however, it has never been applied to spectral data of any kind. To address this new challenge, basic Class Activation Mapping is first applied to a set of bacterial data to obtain baseline results. Then, Gradient Class Activation Mapping is applied on the same dataset. By comparing the results obtained in each case, the method can be verified on spectral data.

One of the main advantages of this method is that it can be applied to very different networks working on very different tasks. For example, in [85], the method is applied to image classification, but also to image captioning and visual question answering.

6 Experiments

After giving a theoretical explanation of the methodology in the previous chapter, this one aims to dive into the actual experiments conducted during this thesis.

6.1 Presentation of the bacterial case

In [99], where the Class Activation Mapping method is applied to Raman spectra, the results presented are obtained on two different datasets of Raman spectra, the first is composed of the Raman spectra of *E. coli* bacteria (divided into *E. coli* 498 and *E. coli* 1116) and the second is composed of the Mid-infared spectra of chicken, turkey and pork samples [105]. But in the case of Covid, the spectra are more complex. Indeed, the Raman spectrum of an *E. coli* bacterium is only the spectrum of that bacterium, it is not necessary to detect it in a more complex sample. Therefore, logically, the results of the class activation mapping on the Covid dataset will most likely be more difficult to interpret than those obtained on *E. coli* bacteria. For this reason, the approach proposed in this thesis is confirmed by using a bacterial database similar to the one used in the reference article.

For the sake of simplicity, it is not exactly the same dataset and model that are used in this thesis and in the reference document. Indeed, the dataset and the model code used in [99] are not available. However, in [60] a model is proposed to classify the different Raman spectra of bacteria. The dataset, as well as the model used, are available. Thus, to avoid the creation of a new model and the training of the latter, the case of bacteria is carried out by using the pre-trained model proposed in [60] and the associated bacterial dataset.

6.1.1 Dataset

The dataset consists of spectra from 30 bacterial and yeast isolates detailed in the Table 6.1. The training part of the dataset, on which the model is trained, consists of 2,000 spectra for each of the 30 isolates. Subsequently, another 100 spectra for each class are used as a validation set.

The test set for this dataset, on which the Class Activation Mapping and Gradient Class Activation Mapping are processed, consists of 100 spectra for each class.

6.1.2 Model

The model created in [60] is a convolutional neural network adapted from the well-known ResNet architecture. As shown in Fig. 6.1, it is essentially composed of an initial convolutional layer followed by 6 residual layers and a final fully connected classification layer. The initial convolutional layer has 64 convolutional filters, while each of the hidden layers has 100 filters. The residual layers contain shortened connections between the input and output of each residual block, allowing for better gradient propagation and stable formation. Each residual layer contains 4 convolutional layers, so the total depth of the network is 26 layers.

Species	Label	Class label
Candida albicans	C. albicans	0
Candida glabrata	C. glabrata	1
Klebsiella aerogenes	K. aerogenes	2
Escherichia coli	E. coli 1	3
Escherichia coli	E. coli 2	4
Enterococcus faecium	E. faecium	5
Enterococcus faecalis	E. faecalis 1	6
Enterococcus faecalis	E. faecalis 2	7
Enterobacter cloacae	E. cloacae	8
Klebsiella pneumoniae	K. pneumoniae 1	9
Klebsiella pneumoniae	K. pneumoniae 2	10
Proteus mirabilis	P. mirabilis	11
Pseudomonas aeruginosa	P. aeruginosa 1	12
Pseudomonas aeruginosa	P. aeruginosa 2	13
Staphylococcus aureus	MSSA 1	14
Staphylococcus aureus	MSSA 3	15
Staphylococcus aureus	MRSA 1 (isogenic)	16
Staphylococcus aureus	MRSA 2	17
Staphylococcus aureus	MSSA 2	18
Salmonella enterica	S. enterica	19
Staphylococcus epidermidis	S. epidermidis	20
Staphylococcus lugdunensis	S. lugdunensis	21
Serratia marcescens	S. marcescens	22
Streptococcus pneumoniae	S. pneumoniae 2	23
Streptococcus pneumoniae	S. pneumoniae 1	24
Streptococcus sanguinis	S. sanguinis	25
Streptococcus pyogenes	Group A Strep.	26
Streptococcus agalactiae	Group B Strep.	27
Streptococcus dysgalactiae	Group C Strep.	28
Streptococcus dysgalactiae	Group G Strep.	29

Table 6.1: Description of the bacteria contained in the dataset.

This model was trained with an Adam Optimizer on the training set described above. It finally obtained a final accuracy of 82.2% at the isolate level. The detailed result can be found in [60].

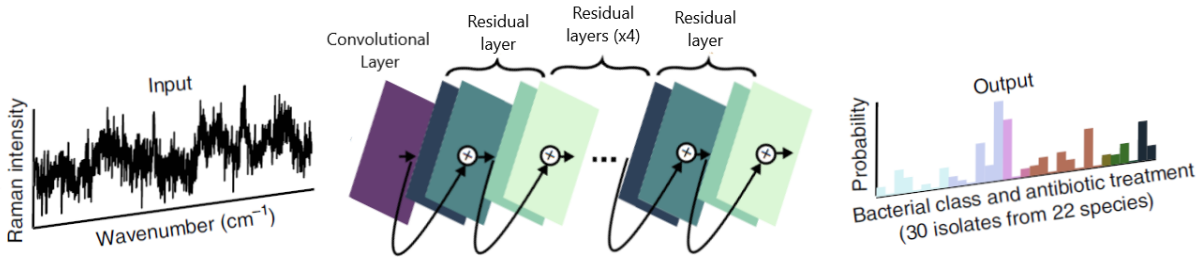

Figure 6.1: Diagram of the convolutional neural network used to classify bacteria. This figure is taken from [60].

For the experiments on the bacterial dataset, this model is directly taken, pre-trained and fine tuned, from the GitHub repository made available by the authors of the reference article.

6.2 Class Activation Mapping experiments

After giving a theoretical overview of the Class Activation Mapping method in the section 3.4 and explaining the thinking behind the implementation of this method in the chapter 5, it is time to give the reader a more technical idea of what has been done in this thesis.

6.2.1 General process

In the following sections, the specifics of the implementation for each case will be detailed, but first, this section is devoted to describing the general steps of the process. These are nothing more or less than an implementation of those described in [99].

Thus, in order to carry out Class Activation Mapping, it is necessary to :

1. **Adapt the original model :** As explained earlier, the first step in using the Class Activation Mapping method is to modify the model and re-train it. The details of this modification for the cases of bacteria and Covid will be detailed in the sections 6.2.2 and 6.2.3 respectively. But, globally, the idea is to "cut" the model just after the last convolutional layer and then add a Global Average Pooling layer. Finally, the prediction is made using a Softmax activation on a linear layer. Obviously, this step takes a lot of time and can lead to a loss of performance.

2. **Make a prediction :** Once the transformed model is trained, it is used to make a prediction about the spectra on which Class Activation Mapping is desired.

3. **Extract the feature maps and weights :** Once the prediction is done, the feature maps of the last convolutional layer and the model weights of the classification layer are extracted. Using the Pytorch library, this is quite easy.

4. **Back-transform the feature maps to the input spectral space :** To obtain the importance of each variable, it is necessary to resample the feature maps in the input spectral space. According to [99], this can be done using an upsampling method called second-order spline interpolation. This is a form of interpolation where the interpolant is a special type of piecewise polynomial called a spline. In other words, instead of fitting a single high degree polynomial to all values at once, spline interpolation fits low degree polynomials to small subsets of values. The "order" of the spline is the order of these "piecewise" polynomials. Thus, to say that upsampling can be performed with a spline interpolation of second order is to say that these "piecewise" polynomials are of order 2.

   Usually spline interpolation is used with a third order, called **cubic spline interpolation**, as this avoids the mathematical problems that can arise with interpolating higher or lower order polynomials. Although these problems are not relevant to the work done here, the fact is that because of these advantages, the cubic spline version is more often implemented. After some testing, on the cases where Class Activation Mapping is performed, the second and third order spline interpolation gives very

similar results. Thus, in order to save time by using a pre-implemented spline interpolation with Pytorch, the cubic version is used to perform the oversampling. For logical

5. **Compute Class Activation Mapping :** Finally, the importance for classification of each variable in the input spectra is obtained by applying the formula presented in Def. 3.4.2.

The main adaptation that is made in each case study to the general process described above is in the first step, namely the modification of the model.

6.2.2 Bacterial case

In the case of the bacterial study, the modifications to the model are quite slight. Indeed, the architecture of the model is very close to the desired one. Thus, the modifications to the model are :

1. **Remove the last batch normalization**
2. **Add a Global Average Pooling layer between the last convolutional layer and the classification layer**

Although these modifications are fairly simple to implement and at first sight do not seem to change the model too much, they do have an influence on the classification results. Indeed, the accuracy of the model drops from 82.2% to 81.2%.This drop is still acceptable, but it is very likely that by starting with a more complex model on which the modifications would be more drastic, the drop would be less negligible.

6.2.3 Covid case

The adaptation of the Convolutional Neural Network to the Covid case is not as simple as the previous one. Indeed, in this model, the convolutional part is followed by a classification part which is not negligible. However, to adapt this model to the Class Activation Map technique, it is necessary to remove it.

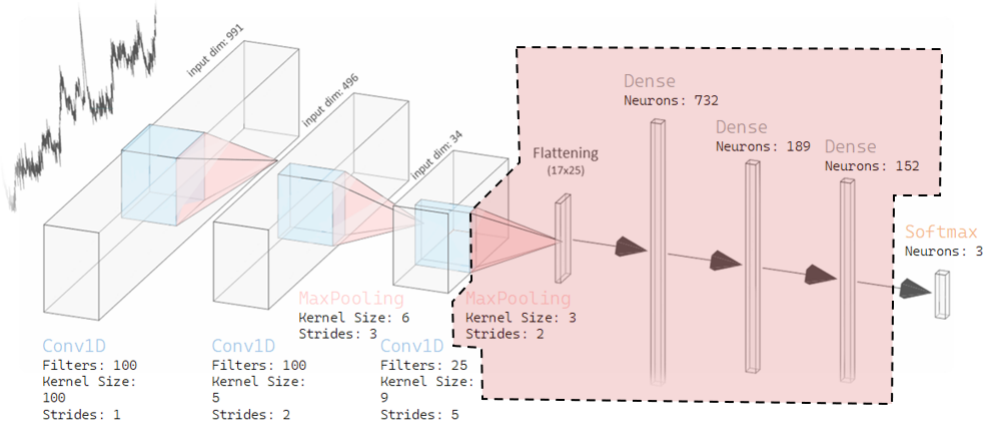

Figure 6.2: Diagram of the modification carried out on the Convolutional Neural Network.

Referring to Fig. 6.2, the steps to fit the model in the Covid case are as follows:

1. **Cut the model after the last convolutional layer :** Basically, this means removing the Max Pooling layer and the whole task part, in red on the figure. That's four layers removed, and more importantly, the whole Dropout part which is the only form of regularisation used in this model.

2. **Add a Global Average Pooling layer :** Then, the removed layers are replaced by a simple Global Average Pooling layer.

As expected, these modifications to the original model have significant performance implications. Accuracy at the spectra level drops by more than 5% (from 83% to 77.8%) and at the patient level by more than 7% (from 89% to 81.7%). The confusion matrix presented in Fig. 6.3 shows that there is a non-negligible amount of additional misclassified patients.

This decrease in performance can be explained by the fact that the new model has fewer learnable parameters and no regularisation. This clearly demonstrates the importance of regularisers such as Dropout and of the classification part which has been removed here.

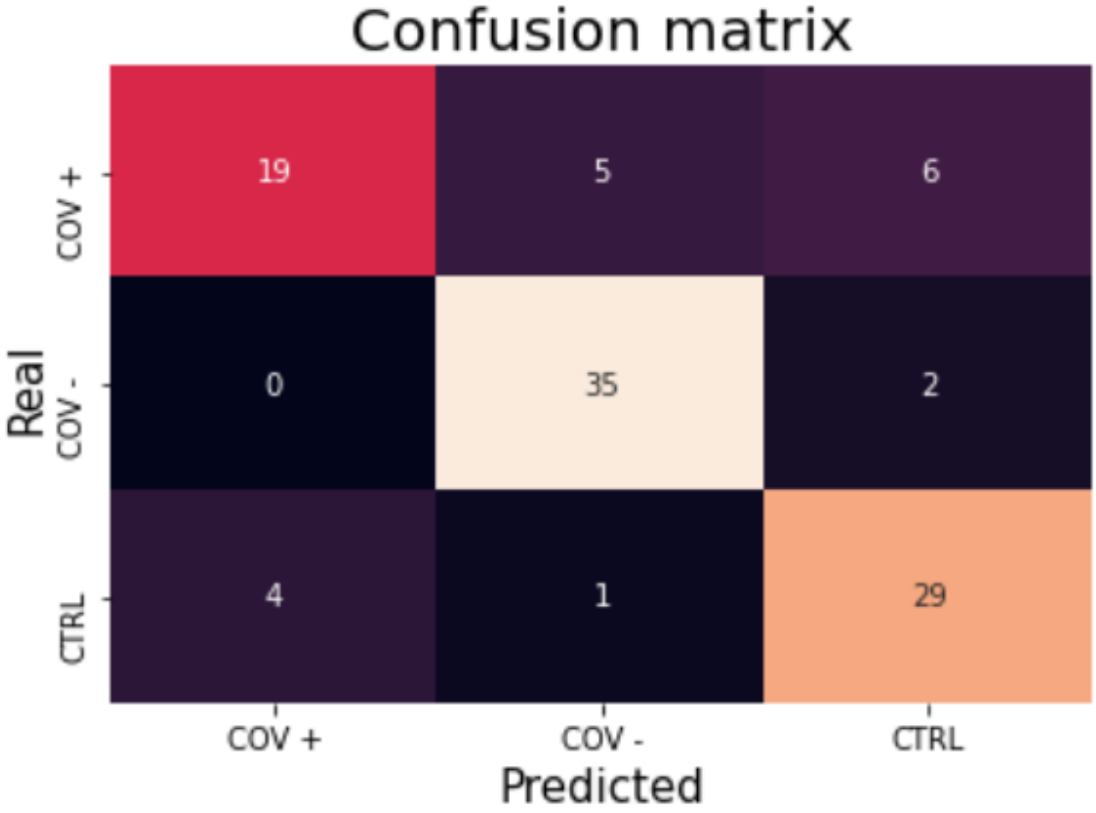

Figure 6.3: Confusion matrix of the results obtained (at patient-level) from the classification of the 101 patients of the Covid classification with the modified model.

However, it is interesting to note that the misclassification itself seems to follow the same patterns as the original model as described in section 4.2.3.

6.3 Guided Class Activation Mapping experiments

The time spent modifying and re-training each of the models associated with each of the case studies and the decrease in performance in both cases shows that, as expected,

whatever the results of the Class Activation Mapping method it is not a perfect approach. Therefore, the Gradient Class Activation Mapping method is also implemented in this thesis.

6.3.1 General Process

Compared to the implementation of the Class Activation Mapping method, this is simpler. Indeed, it is not necessary to modify the model. It is essentially a question of applying the process defined in [85] by adapting it to the spectral data.

The general process consists of the following steps:

1. **Make a prediction :** This time there is no need to change the model and the prediction is made with the original trained model.
2. **Get the feature maps from the last convolutional layer :** As for the basic method, it is very easy to do with the Pytorch library.
3. **Get the gradient of the class with respect to the feature maps :** As with the previous step, there are a few tricks in Pytorch that allow to do this quite easily.
4. **Pool the gradients**
5. **Obtain the Grad-CAM value :** According to Eq. 3.10, the map is weighted by the corresponding pooled gradients.
6. **Interpolate the values obtained to the input spectral space :** This is the only modification to the original method proposed in the reference document. For this purpose, the same method as for the basic CAM is used, namely cubic spline interpolation.

6.3.2 Study cases

Unlike the simple Class Activation Mapping method, it is not necessary to distinguish between the implementation of bacterial and covid cases. Indeed, this method simply uses the original model and the information it contains, and is therefore applicable to any type of Convolutional Neural Network whatever its architecture and the task it is trained on.

7 Results and discussion

After presenting the experiments carried out in the course of this thesis, this chapter is devoted to the presentation of the results obtained and the discussions arising from them. The results are presented following the same logic as the previous chapters.

7.1 Results of Class Activation Mapping

Even if it has been proven that the Class Activation Mapping method is not an ideal solution, it is still interesting to take a closer look at the results obtained. Indeed, the results of this method on the bacterial case will serve as reference results to validate the proposed method.

7.1.1 Bacterial case

For the sake of simplicity, the method was applied to the entire data set, but only the results for the bacterium *E. coli* are presented here.

In Fig. 7.1, an example of the results produced by the Class Activation Mapping method is presented. The second graph is a case of misclassification, but if we look at the shape of the curve representing the importance of each variable (the middle curves), it is quite clear that they are quite similar.

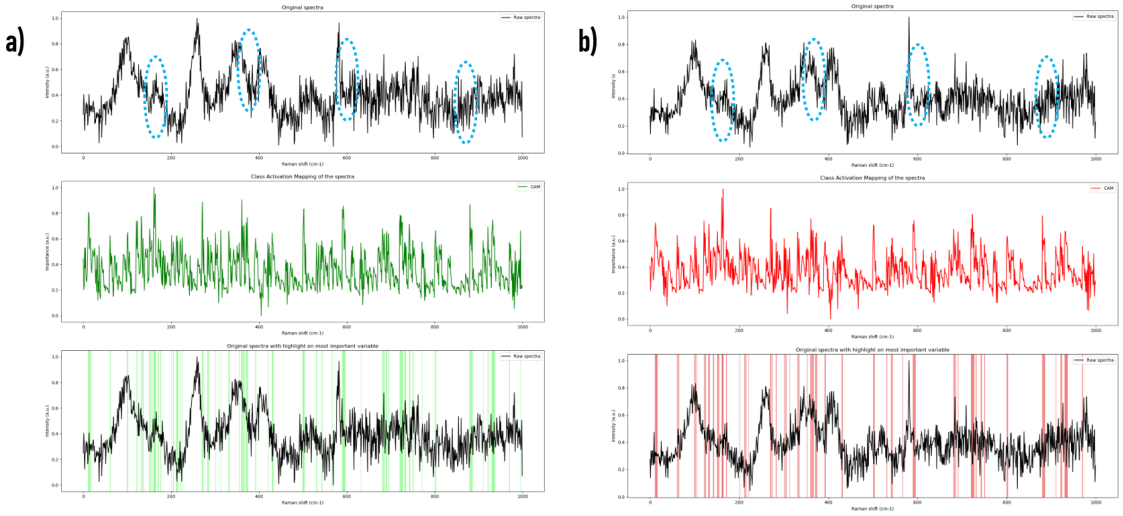

Figure 7.1: Examples of class activation mapping results on a bacterial data set. **a)** Results for a correctly classified *E. Coli*. **b)** Results for a *P. mirabilis* bacterium classified as an *E. Coli* bacterium. From top to bottom, the first spectrum is the raw spectrum, the second is the importance of each variable in the spectrum calculated using the Class Activation Mapping method and the last is the raw spectrum with the most important 10% of the spectrum highlighted.

Furthermore, the most important peaks, those with the highest Class Activation Mapping values, which are circled in blue in Fig. 7.1, are present in both spectra. This ob-

servation confirms the fact that these peaks are really decisive for the model to decide whether a spectrum belongs to the class *E. coli*. Further examples can be found in B.1.

Although the example shown in Fig. 7.1 is informative, it is an example calculated on two spectra. The real aim is to generalise these results. Thus, for all spectra classified as *E. coli*, the 10% of the most important variable are selected, and then the idea is to know which ones are used in the majority of cases to classify a spectrum as *E. coli*. These results, shown in Fig. 7.2, confirm the previously mentioned hypothesis that the variables circled in blue were the most important for the classification.

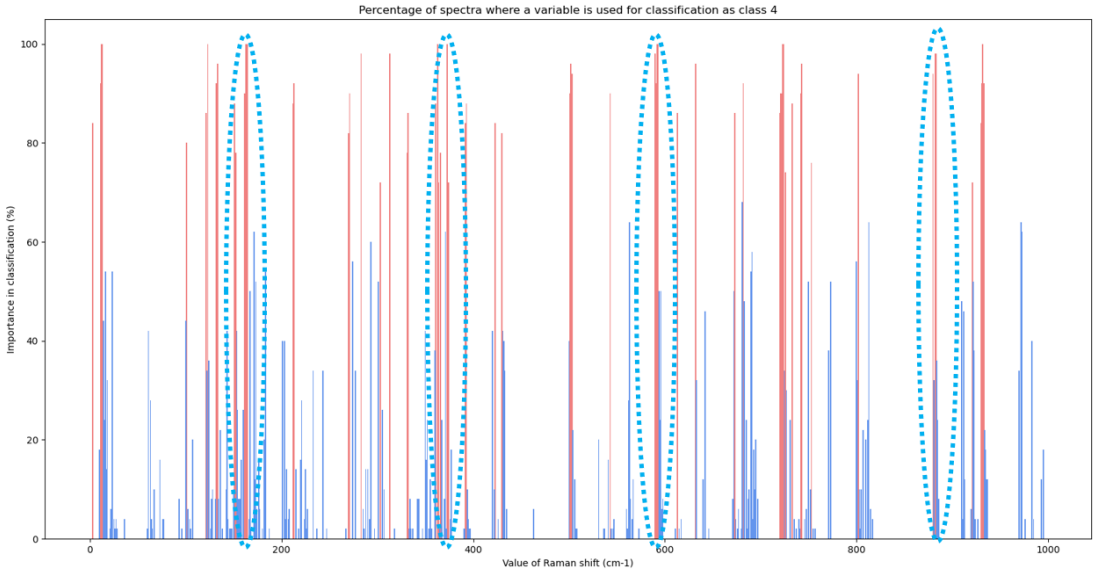

Figure 7.2: Bar chart showing the percentage of spectra for which a variable is in the top 10%. Blue bars represent variables used in less than 70% of classifications, and red bars represent variables used in at least 70% of cases.

It is important to note that the Class Activation Mapping curves obtained here on *E. coli* bacteria are consistent with the results obtained in [99]. This makes it possible to reasonably assume that the implementation proposed here is valid.

7.1.2 Covid case

Before showing and discussing the results of the Class Activation Mapping on the Covid case, it is important to remind the reader of the fact that most misclassifications are made between Covid positive and control patients. Thus, these cases are the ones presented here, but the results for the Covid Negative spectra are available at B.2.

In Fig. 7.3, an example of the results produced by the class activation mapping method is presented. The second graph is a case of misclassification, it is a Control spectrum that has been classified as Covid Positive. But if we look at the shape of the curve representing the importance of each variable (the middle curves), it is quite clear that they are quite similar.

As in the previous case, the most important peaks, those with the highest Class Activation Mapping values, which are circled in blue in Fig. 7.3, are present in both spectra.

In Fig. 7.4, the same pattern can be seen but in the other direction, this time it is a control spectrum that is misclassified with a positive Covid spectrum.

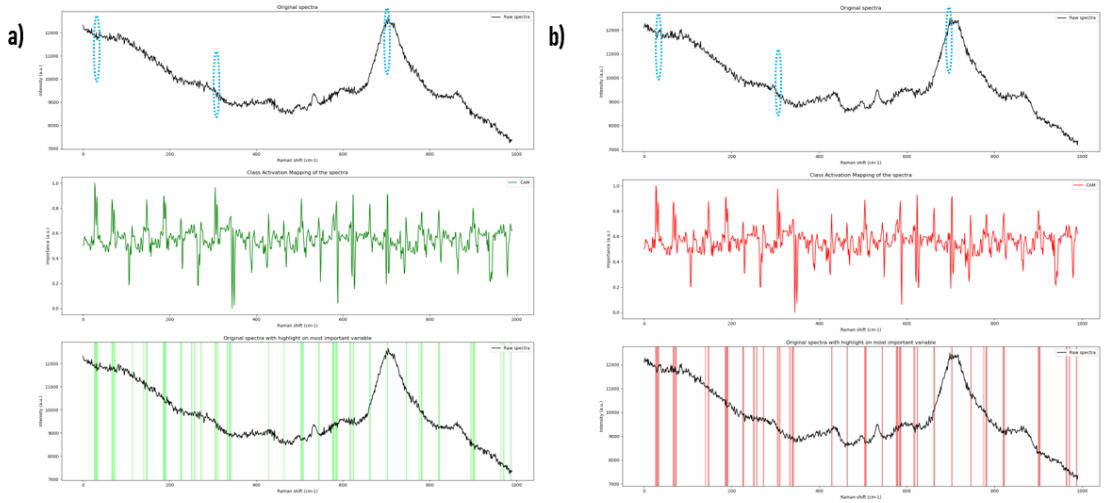

Figure 7.3: Examples of class activation mapping results on covid data set. **a)** Results for a correctly classified Covid Positive. **b)** Results for a Control classified as an Covid Positive. From top to bottom, the first spectrum is the raw spectrum, the second is the importance of each variable in the spectrum calculated using the Class Activation Mapping method and the last is the raw spectrum with the most important 10% of the spectrum highlighted.

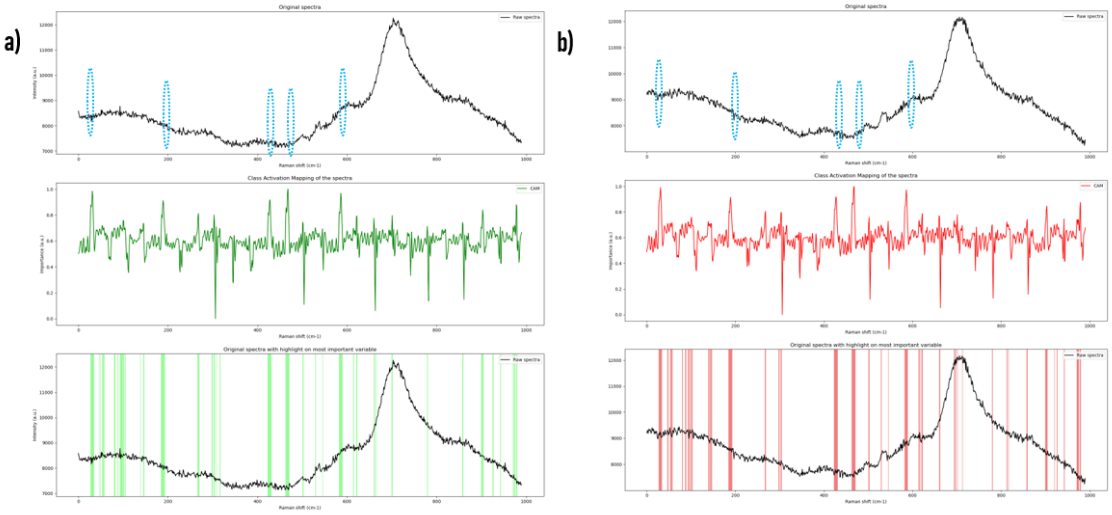

Figure 7.4: Examples of class activation mapping results on covid data set. **a)** Results for a correctly classified Control. **b)** Results for a Covid Positive classified as a Control. From top to bottom, the first spectrum is the raw spectrum, the second is the importance of each variable in the spectrum calculated using the Class Activation Mapping method and the last is the raw spectrum with the most important 10% of the spectrum highlighted.

At first sight, the Class Activation Mapping method seems to work well in the case of Covid. One can notice that in Fig. 7.5 and in Fig. 7.6 something interesting append.

Indeed, in both figures, that is for the Covid Positive classification and for the Control classification, there is no highlighted variable. All the variables in the spectra are used in approximately the same proportion to classify them. These results are not relevant as they do not provide any additional information to the problem presented here, but they

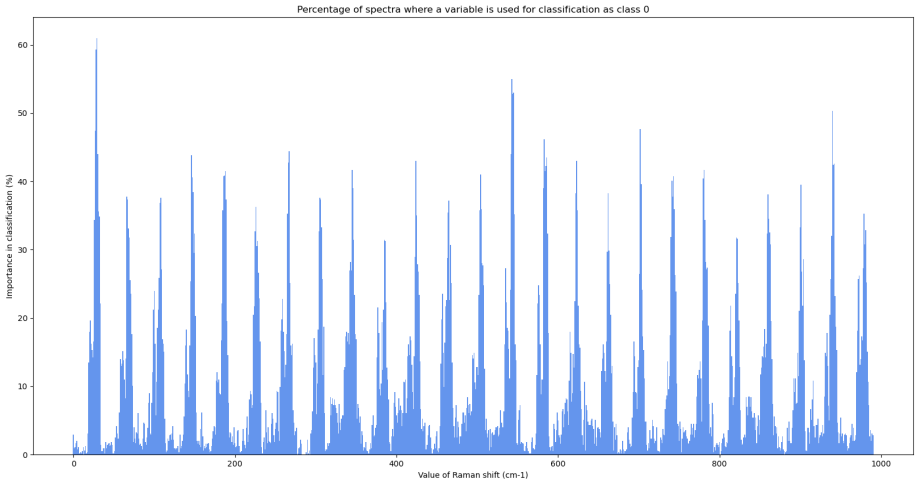

Figure 7.5: Bar chart showing the percentage of spectra for which a variable is in the top 10% for the classification as Covid Positive. Blue bars represent variables used in less than 70% of classifications, and red bars represent variables used in at least 70% of cases.

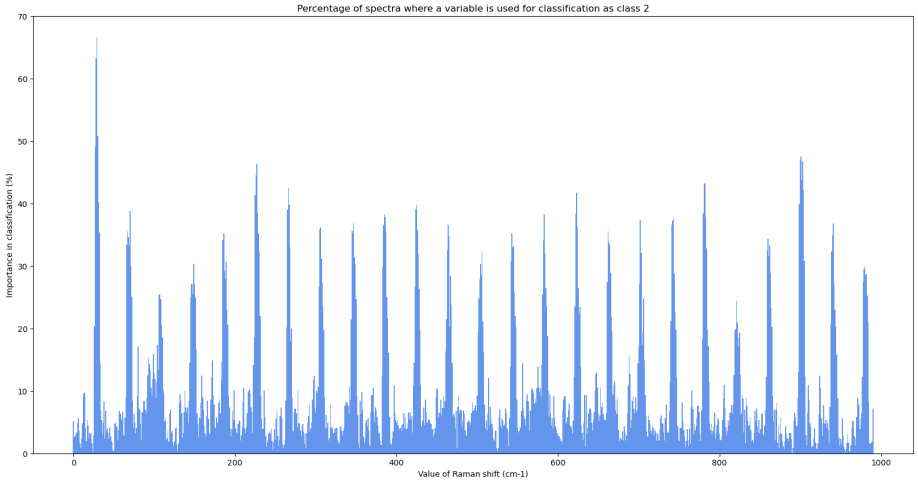

Figure 7.6: Bar chart showing the percentage of spectra for which a variable is in the top 10% for the classification as Control. Blue bars represent variables used in less than 70% of classifications, and red bars represent variables used in at least 70% of cases.

will be discussed later.

7.2 Results of Gradient Class Activation Mapping

Before discussing the results, those obtained with the Gradient Class Activation Mapping are presented in this section. In order to allow comparison of the two methods, the same spectra as in the previous section will be analysed. As for the previous section, more example can be found in B.3 and B.4.

7.2.1 Bacterial case

In Fig. 7.7, the observation made in the previous section is still valid, the peaks with the highest Gradient Class Activation Mapping are present in both spectra. But what is remarkable is that these peaks are not the same as those highlighted by the Class Activation Mapping method.

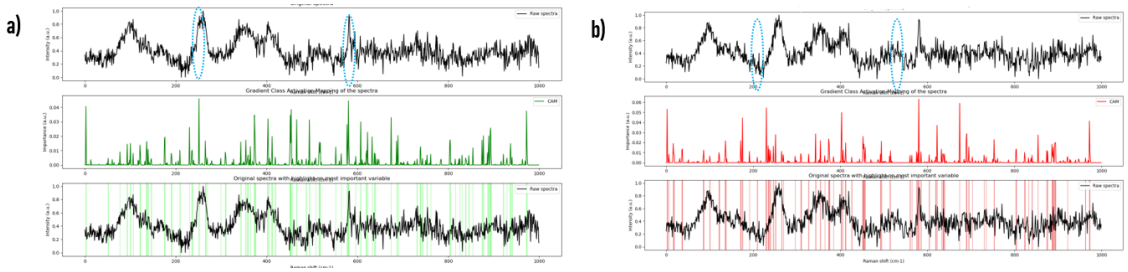

Figure 7.7: Examples of gradient class activation mapping results on a bacterial data set. **a)** Results for a correctly classified *E. Coli*. **b)** Results for a *P. mirabilis* bacterium classified as an *E. Coli* bacterium. From top to bottom, the first spectrum is the raw spectrum, the second is the importance of each variable in the spectrum calculated using the Gradient Class Activation Mapping method and the last is the raw spectrum with the most important 10% of the spectrum highlighted.

The generalisation shown in Fig. 7.8 confirms these results.

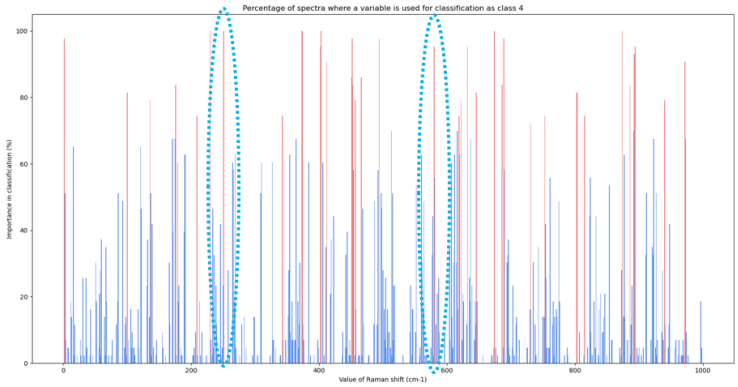

Figure 7.8: Bar chart showing the percentage of spectra for which a variable is in the top 10%. Blue bars represent variables used in less than 70% of classifications, and red bars represent variables used in at least 70% of cases.

It is interesting to note that, although the results are different from those obtained with Class Activation Mapping, they are not inconsistent with those presented in the reference article [99].

7.2.2 Covid case

In Fig. 7.9, an example of the results produced by the gradient class activation mapping method is presented. The second graph is a case of misclassification, it is a Control spectrum that has been classified as Covid Positive.

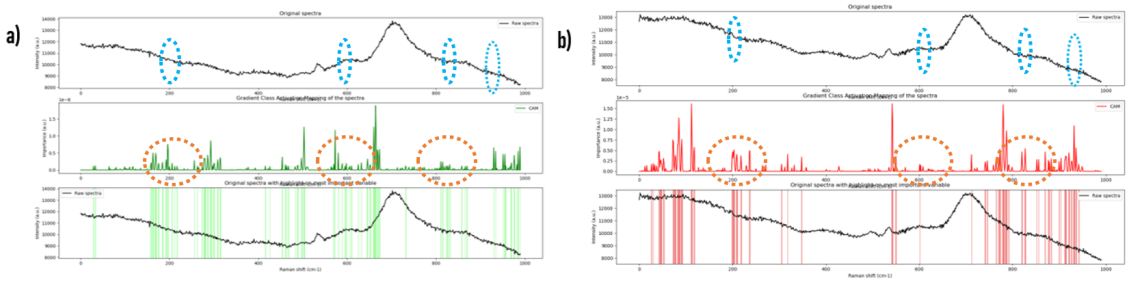

Figure 7.9: Examples of gradient class activation mapping results on covid data set. **a)** Results for a correctly classified Covid Positive. **b)** Results for a Control classified as an Covid Positive. From top to bottom, the first spectrum is the raw spectrum, the second is the importance of each variable in the spectrum calculated using the Gradient Class Activation Mapping method and the last is the raw spectrum with the most important 10% of the spectrum highlighted.

Some peaks are highlighted, but the curves of Gradient Class Activation Mapping seem to be more complex than those of Class Avtivation Mapping. Indeed, this time, the two spectra do not have their highest peak in common.

In Fig. 7.10, the same thing can be seen but in the other direction, this time it is a control spectrum that is misclassified with a positive Covid spectrum.

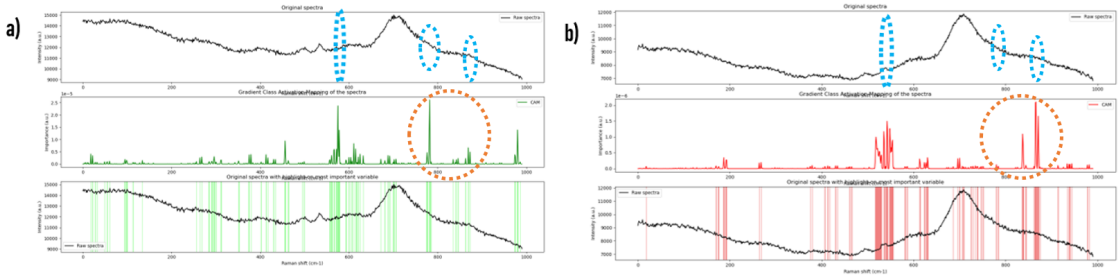

Figure 7.10: Examples of gradient class activation mapping results on covid data set. **a)** Results for a correctly classified Control. **b)** Results for a Covid Positive classified as a Control. From top to bottom, the first spectrum is the raw spectrum, the second is the importance of each variable in the spectrum calculated using the Gradient Class Activation Mapping method and the last is the raw spectrum with the most important 10% of the spectrum highlighted.

In Figs. 7.9 and 7.10, it appears that the peaks that are common to both spectra are not necessarily those that correspond to the highest value of the GRadient CAM. Indeed, circled in orange on the figures, we notice that these values are part of the highest 10% of the CAM Gradient values but are not in the same range for both spectra, a really high Gradient CAM value can be associated with a really smaller value on the other spectrum.

In Fig. 7.11 and in Fig. 7.12 the same lack of generalisation as with the Class Activation Mapping method is noted.

Compared to the Class Activation Mapping, these bar charts seem to be less informative. This can be explained by the more "complex" composition of the importance curve and by the use of ReLU in the calculation. Indeed, there are no negative values with the Gradient Class Activation Mapping method and, in addition, it seems that more variables are valued for the classification task.

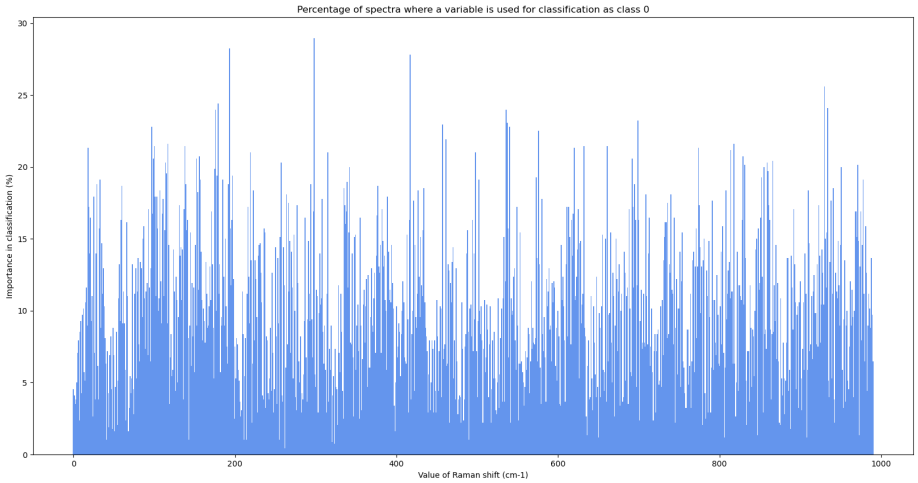

Figure 7.11: Bar chart showing the percentage of spectra for which a variable is in the top 10% for the classification as Covid Positive. Blue bars represent variables used in less than 70% of classifications, and red bars represent variables used in at least 70% of cases.

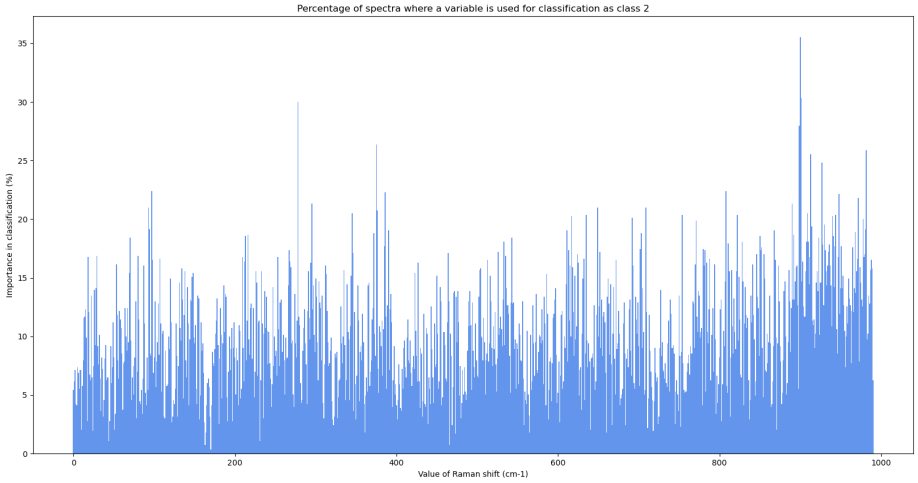

Figure 7.12: Bar chart showing the percentage of spectra for which a variable is in the top 10% for the classification as Control. Blue bars represent variables used in less than 70% of classifications, and red bars represent variables used in at least 70% of cases.

7.3 Discussion

First of all, it is necessary to clarify why, despite the change of the most important variables between the CAM method and the Grad-CAM method, the results remain consistent with the reference article [99]. In fact, it is because the data set and the model are not exactly the same. Moreover, the results presented in the article are not presented in the same format. In reality, they are not really presented, it is their distribution that is presented. So to say that the results are consistent is actually a short way of saying that the distribution of results is consistent. Thus, it is therefore rather the distribution

of the most important variables that is relevant.

The second element that needs to be discussed is the low generalisation of the results obtained on the Covid case, whatever the method used. To explain this fact, two main hypotheses can be put forward:

- The first explanation that can be proposed is certainly the easiest. As the methods have already shown their performance on "simple" spectra, the problem may be the complexity of the input data.
- The second is directly related to the model architecture. Indeed, due to the applied Leave-One-Patient-Out Cross-Validation 4.1.4, evaluation scheme there are 101 models trained and tested on different parts of the original dataset, and each of these models contains different parameters. As these parameters are used to calculate significance, it is likely that each model does not use the same part of the spectrum to rank it. Thus, the generalisation of importance variable is not efficient.

Thus, to confirm the suspicion that the lack of generalisation is due to the multiplicity of the model, it was decided to test the method on each model separately.

In Fig. 7.13, an example of what is obtained model by model with Class Activation Mapping is shown. The first bar graph is from model trained over the 0th fold of the Leave-One-Patient-Out Cross-Validation and the second from model trained over the 2nd fold, the fact is that both models do not agree on the most important variables for classification. This clearly explains why the generalization over all models does not work.

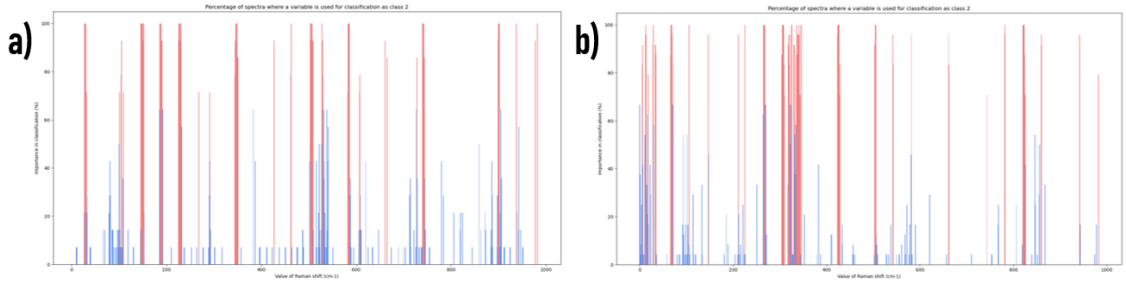

Figure 7.13: Bar chart showing the percentage of spectra for which a variable is in the top 10% for the classification as Control. Blue bars represent variables used in less than 70% of classifications, and red bars represent variables used in at least 70% of cases. **a)** Results obtained with model 0. **b)** Results obtained with model 2.

In Fig. 7.14, the results are from the Gradient Class Activation Mapping method. As in the previous case, the two models do not agree on the most important variables.

These results call into question the whole classification methodology used in this thesis. Indeed, for a brand new spectrum, the classification is performed with all models separated but the fact is that not all models are focused on the same variables. This may also explain the difference in accuracy between the models, as some models have very good accuracy while others have very poor accuracy.

Another problem with these results is that each model has only one patient as a test set, so the generalisation is obtained on a single class. This is not very useful for understanding misclassification.

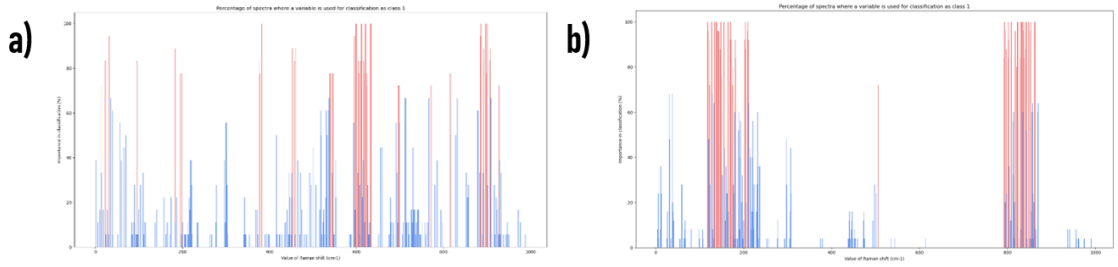

Figure 7.14: Bar chart showing the percentage of spectra for which a variable is in the top 10% for the classification as Covid Negative. Blue bars represent variables used in less than 70% of classifications, and red bars represent variables used in at least 70% of cases. **a)** Results obtained with model 66. **b)** Results obtained with model 70.

7.4 Conclusion

In conclusion, the method proposed in this thesis is promising. Obviously, to really show all its capabilities, future work is needed, and this will be discussed in the next chapter. But, with the current results, it is already possible to confirm several hypotheses.

Firstly, the gradient class activation mapping method is well applicable to spectral data, with minimal adaptation of the original method. Secondly, this method is really consistent with the real-world application because it is really easy to obtain results on a new model.

To another extent, this thesis has already shown the importance of an effective Explicable Artificial Intelligence approach to explain the results of a Deep Neural Network, even more so in the medical field. Indeed, despite the importance of each variable for the model, the explanation of the classification is not easy. The complexity of the importance curves in the Gradient Class Activation Mapping experiments shows that the decision is not made on simple criteria. On the contrary, it seems that many variables are involved in the decision process, that means classification is not based on the presence or absence of a specific peak, but rather on a complex combination of them.

This thesis has also raised a new question as to whether Leave-One-Patient-Out Cross-Validation is really a good training method for this type of problem. Indeed, this method seems to make each model focus on different features.Thus, the important features are not always the same, which can lead to a lack of robustness, especially in terms of reliability of the model. But, on the other hand, at the level of generalisation, this method is interesting. Indeed, the models formed by the Leave-One-Patient-Out Cross-Validation constitute a set (in the mathematical sense). Thus, using the whole set to make a new prediction may be more robust than using just one.

8 Future work

Based on the proposed approach and the current results, many future works can be envisaged :

1. **Carry out further analysis of the results :** First of all, it will be really interesting to study the results already obtained with an expert. Such an analysis will allow a better understanding of how the model has performed the classification and to refine the visualization method.

2. **Reduce the complexity of spectra :** In order to obtain a better generalisation of the importance of the variables for the Covid case, a series of technical experiments on the input spectra could be performed. Indeed, as references in the literature using class activation mapping methods on spectral data do so after applying some preprocessing, including dimension reduction, to the spectra. Thus, with some research, probably starting with the preprocessing pipeline explained in this thesis, it might be possible to reduce the complexity of the spectra to achieve better generalisation.

3. **Overcome the problem of multiple models :** Another line of research to obtain a better generalisation is to overcome the problem caused by the use of multiple models. One option to overcome this problem is to use a single model, obviously, to do this and keep the performance obtained it is necessary to obtain more data.

4. **Investigate the Leave-One-Patient-Out Cross-Validation method :** As described in the conclusion, it might be interesting to study whether this method is a good approach for this type of problem. Indeed, the results highlighted in the discussion section are very interesting and it could be useful to deepen them, not only for the understanding of the model but also for its improvement.

5. **Implement the Guided Gradient Class Activation Mapping method :** As mentioned in this thesis, Gradient Class Activation Mapping has a variant, called Guided Gradient Class Activation Mapping. This method was defined because it seems to give more accurate results on images, so it is likely that it could do the same for spectral data. Thus, a possible future work could be to implement this method.

6. **Investigate further Explainable Artificial methods :** This thesis has presented a number of Explicable Artificial Intelligence methods, with a focus on Class Activation Mapping and its variants. However, there are a large number of methods and a significant number of them can be applied to spectral data, so it might be interesting to test different methods to compare results.

7. **Go further with Class Activation Mapping :** It is also possible to go further with the Class Activation Mapping method. Indeed, this thesis just focused on explaining the classification based on the input data, but in the reference paper this method is also used to explain each convolutional layer of the model. Thus, it might be interesting to use it for this purpose in order to get a deeper understanding of the model, and perhaps this can lead to an improvement of the model.

Additional information

The code, detailed thesis and presentation associated with this thesis are available on GitHub.

Bibliography

[1] L.-P. Choo-Smith et al. "Medical applications of Raman spectroscopy: From proof of principle to clinical implementation". In: *Biopolymers* 67 (2002), pp. 1–9. DOI: 10.1002/bip.10064.

[2] Pavel Matousek and Nicholas Stone. "Recent advances in the development of Raman spectroscopy for deep non-invasive medical diagnosis". In: *Journal of Biophotonics* 6 (2013), pp. 7–19. DOI: 10.1002/jbio.201200141.

[3] Isaac Pence and Anita Mahadevan-Jansen. "Clinical instrumentation and applications of Raman spectroscopy". In: *Chemical Society Review* 45 (2016), pp. 1958–1979. DOI: 10.1039/C5CS00581G".

[4] Kenny Kong et al. "Raman spectroscopy for medical diagnostics – From in-vitro biofluid assays to in-vivo cancer detection". In: *Advanced Drug Delivery Reviews* 89 (2015). Pharmaceutical applications of Raman spectroscopy – from diagnosis to therapeutics, pp. 121–134. DOI: 10.1016/j.addr.2015.03.009.

[5] Jinchao Liu et al. "Deep Convolutional Neural Networks for Raman Spectrum Recognition: A Unified Solution". In: *Analyst* 142.21 (2017), pp. 4067–4074. DOI: 10.1039/C7AN01371J.

[6] Félix Lussier et al. "Deep learning and artificial intelligence methods for Raman and surface-enhanced Raman scattering". In: *TrAC Trends in Analytical Chemistry* 124 (2020). DOI: 10.1016/j.trac.2019.115796.

[7] C Carlomagno et al. "OPEN COVID -- 19 salivary Raman fingerprint : innovative approach for the detection of current and past SARS -- CoV-2 infections". In: *Scientific Reports* (2021), pp. 1–13. DOI: 10.1038/s41598-021-84565-3.

[8] Tao Qin. "Machine Learning Basics". In: *Dual Learning*. Springer Singapore, 2020, pp. 11–23. DOI: 10.1007/978-981-15-8884-6_2.

[9] Gopinath Rebala, Ajay Ravi, and Sanjay Churiwala. "Machine Learning Definition and Basics". In: *An Introduction to Machine Learning*. Springer International Publishing, 2019, pp. 1–17. DOI: 10.1007/978-3-030-15729-6_1.

[10] Sandro Skansi. "Machine Learning Basics". In: *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. Springer International Publishing, 2018, pp. 51–77. DOI: 10.1007/978-3-319-73004-2_3.

[11] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., 1997.

[12] F.Y. Osisanwo et al. "Supervised Machine Learning Algorithms: Classification and Comparison". In: *International Journal of Computer Trends and Technology* 48 (2017), pp. 128–138. DOI: 10.14445/22312803/IJCTT-V48P126.

[13] Kai Ming Ting. "Confusion Matrix". In: *Encyclopedia of Machine Learning and Data Mining*. Springer US, 2017. DOI: 10.1007/978-1-4899-7687-1_50.

[14] Reyhan Selin Uysal et al. "Determination of butter adulteration with margarine using Raman spectroscopy". In: *Food Chemistry* 141.4 (2013), pp. 4397–4403. DOI: 10.1016/J.FOODCHEM.2013.06.061.

[15] Arslan Amjad et al. "Raman spectroscopy based analysis of milk using random forest classification". In: *Vibrational Spectroscopy* 99 (2018), pp. 124–129. DOI: 10.1016/J.VIBSPEC.2018.09.003.

[16] H. Mohamadi Monavar et al. "Determining quality of caviar from Caspian Sea based on Raman spectroscopy and using artificial neural networks". In: *Talanta* 111 (2013), pp. 98–104. DOI: 10.1016/J.TALANTA.2013.02.046.

[17] Hannah Dies et al. "Rapid identification and quantification of illicit drugs on nanodendritic surface-enhanced Raman scattering substrates". In: *Sensors and Actuators B: Chemical* 257 (2018), pp. 382–388. DOI: 10.1016/J.SNB.2017.10.181.

[18] Yves Roggo, Klara Degardin, and Pierre Margot. "Identification of pharmaceutical tablets by Raman spectroscopy and chemometrics". In: *Talanta* 81.3 (2010), pp. 988–995. DOI: 10.1016/J.TALANTA.2010.01.046.

[19] Kyle C. Doty and Igor K. Lednev. "Differentiation of human blood from animal blood using Raman spectroscopy: A survey of forensically relevant species". In: *Forensic Science International* 282 (2018), pp. 204–210. DOI: 10.1016/J.FORSCIINT.2017.11.033.

[20] "Determining Gender by Raman Spectroscopy of a Bloodstain". In: *Analytical Chemistry* 89.3 (2017), pp. 1486–1492. DOI: 10.1021/ACS.ANALCHEM.6B02986.

[21] Kyle C. Doty and Igor K. Lednev. "Differentiating Donor Age Groups Based on Raman Spectroscopy of Bloodstains for Forensic Purposes". In: *ACS Central Science* 4.7 (2018), pp. 862–867. DOI: 10.1021/ACSCENTSCI.8B00198.

[22] Christoph Gasser et al. "Stand-off Hyperspectral Raman Imaging and Random Decision Forest Classification: A Potent Duo for the Fast, Remote Identification of Explosives". In: *Analytical Chemistry* 91.12 (2019), pp. 7712–7718. DOI: 10.1021/ACS.ANALCHEM.9B00890.

[23] M. Yogesha et al. "A micro-Raman and chemometric study of urinary tract infection-causing bacterial pathogens in mixed cultures". In: *Analytical and Bioanalytical Chemistry* 411.14 (2019), pp. 3165–3177. DOI: 10.1007/S00216-019-01784-4.

[24] Sandra Kloß et al. "Culture Independent Raman Spectroscopic Identification of Urinary Tract Infection Pathogens: A Proof of Principle Study". In: *Analytical Chemistry* 85.20 (2013), pp. 9610–9616. DOI: 10.1021/AC401806F.

[25] Satya Kiran Koya et al. "Rapid Detection of Clostridium difficile Toxins in Stool by Raman Spectroscopy". In: *Journal of Surgical Research* 244 (2019), pp. 111–116. DOI: 10.1016/J.JSS.2019.06.039.

[26] Saranjam Khan et al. "Random Forest-Based Evaluation of Raman Spectroscopy for Dengue Fever Analysis:" in: 71.9 (2017), pp. 2111–2117. DOI: 10.1177/0003702817695571.

[27] Saranjam Khan et al. "Analysis of dengue infection based on Raman spectroscopy and support vector machine (SVM)". In: *Biomedical Optics Express* 7.6 (2016). DOI: 10.1364/BOE.7.002249.

[28] Katarína Rebrošová et al. "Rapid identification of staphylococci by Raman spectroscopy". In: *Scientific Reports 2017 7:1* 7.1 (2017), pp. 1–8. DOI: 10.1038/s41598-017-13940-w.

[29] M. Harz et al. "Micro-Raman spectroscopic identification of bacterial cells of the genus Staphylococcus and dependence on their cultivation conditions". In: *Analyst* 130.11 (2005), pp. 1543–1550. DOI: 10.1039/B507715J.

[30] Edgar Guevara et al. "Use of Raman spectroscopy to screen diabetes mellitus with machine learning tools". In: *Biomed. Opt. Express* 9.10 (2018), pp. 4998–5010. DOI: 10.1364/BOE.9.004998.

[31] Xiaoyu Cui et al. "Analysis and classification of kidney stones based on Raman spectroscopy". In: *Biomed. Opt. Express* 9.9 (2018), pp. 4175–4183. DOI: 10.1364/BOE.9.004175.

[32] Martina Sattlecker, Nicholas Stone, and Conrad Bessant. "Current trends in machine-learning methods applied to spectroscopic cancer diagnosis". In: *TrAC Trends in Analytical Chemistry* 59 (2014), pp. 17–25. DOI: 10.1016/J.TRAC.2014.02.016.

[33] J. Nicholas Taylor et al. "High-Resolution Raman Microscopic Detection of Follicular Thyroid Cancer Cells with Unsupervised Machine Learning". In: *Journal of Physical Chemistry B* 123.20 (2019), pp. 4358–4372. DOI: 10.1021/ACS.JPCB.9B01159.

[34] Shangyuan Feng et al. "A noninvasive cancer detection strategy based on gold nanoparticle surface-enhanced raman spectroscopy of urinary modified nucleosides isolated by affinity chromatography". In: *Biosensors and Bioelectronics* 91 (2017), pp. 616–622. DOI: 10.1016/J.BIOS.2017.01.006.

[35] Shaveta Dargan et al. "A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning". In: *Archives of Computational Methods in Engineering 2019 27:4* 27.4 (2019), pp. 1071–1092. DOI: 10.1007/S11831-019-09344-W.

[36] C. M. BERNERS-LEE. "Cybernetics and Forecasting". In: 219.5150 (1968), pp. 202–203. DOI: 10.1038/219202B0.

[37] A. G. Ivakhnenko. "Polynomial Theory of Complex Systems". In: *IEEE Transactions on Systems, Man and Cybernetics* 1.4 (1971), pp. 364–378. DOI: 10.1109/TSMC.1971.4308320.

[38] Rina Dechter. "Learning While Searching In Constraint-Satisfaction-Problems". In: *Annals of Mathematics* (1986), pp. 178–183.

[39] Anurag Bhardwaj, Wei Di, and Jianing Wei. 2018.

[40] Chigozie Nwankpa et al. "Activation Functions: Comparison of trends in Practice and Research for Deep Learning". In: (2018).

[41] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks*. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (visited on 08/10/2021).

[42] Wenfeng Gong et al. "A Novel Deep Learning Method for Intelligent Fault Diagnosis of Rotating Machinery Based on Improved CNN-SVM and Multichannel Data Fusion". In: *Sensors* 19.7 (2019). DOI: 10.3390/s19071693.

[43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[44] Claude Lemaréchal. "Cauchy and the Gradient Method". In: *Documenta Mathematica* ISMP (2012), pp. 251–254.

[45] Herbert Robbins and Sutton Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407. DOI: 10.1214/aoms/1177729586.

[46] Boris T Polyak. "Some methods of speeding up the convergence of iteration methods". In: *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964), pp. 1–17. DOI: 10.1016/0041-5553(64)90137-5.

[47] Y. NESTEROV. "A method for solving the convex programming problem with convergence rate $O(1/k^2)$". In: *Dokl. Akad. Nauk SSSR* 269 (1983), pp. 543–547.

[48] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536. DOI: 10.1038/323533a0.

[49] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization". In: vol. 12. 2010, pp. 257–269.

[50] Diederik P. Kingma and Jimmy Lei Ba. "Adam: A method for stochastic optimization". In: 2015.

[51] Courville Aaron Goodfellow lan, Bengio Yoshua. *Deep Learning*. 2016.

[52] Jacopo Cavazza et al. "Dropout as a low-rank regularizer for matrix factorization". In: PMLR, 2018, pp. 435–444.

[53] Poorya Mianjy, Raman Arora, and Rene Vidal. "On the Implicit Bias of Dropout". In: *35th International Conference on Machine Learning, ICML 2018* 8 (2018), pp. 5701–5717.

[54] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958.

[55] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *32nd International Conference on Machine Learning, ICML 2015* 1 (2015), pp. 448–456.

[56] Chris M. Bishop. "Training with Noise is Equivalent to Tikhonov Regularization". In: *Neural Computation* 7.1 (1995), pp. 108–116. DOI: 10.1162/NECO.1995.7.1.108.

[57] Kam Jim, C L Giles, and Bill G Horne. "Effects of Noise on Convergence and Generalization in Recurrent Networks". In: *Neural Information Processing Systems (NIPS)* (1995), pp. 649–656.

[58] Jinchao Liu et al. "Dynamic Spectrum Matching with One-shot Learning". In: *Chemometrics and Intelligent Laboratory Systems* 184 (2018), pp. 175–181.

[59] Xiaqiong Fan et al. "Deep learning-based component identification for the Raman spectra of mixtures". In: *Analyst* 144.5 (2019), pp. 1789–1798. DOI: 10.1039/C8AN02212G.

[60] Chi-Sing Ho et al. "Rapid identification of pathogenic bacteria using Raman spectroscopy and deep learning". In: *Nature Communications* 10.1 (2019). DOI: 10.1038/s41467-019-12898-9.

[61] Feiyu Xu et al. "Explainable AI: A Brief Survey on History, Research Areas, Approaches and Challenges". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11839 LNAI (2019), pp. 563–574. DOI: 10.1007/978-3-030-32236-6_51.

[62] A. Carlisle Scott et al. "Explanation Capabilities of Production-Based Consultation Systems". In: *American Journal of Computational Linguistics* (1977), pp. 1–50.

[63] W. R Swartout. "Explaining and Justifying in Expert Consulting Programs". In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence* (1981).

[64] Matt Turek. *Explainable Artificial Intelligence*. URL: https://www.darpa.mil/program/explainable-artificial-intelligence.

[65] Zachary C. Lipton. "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery." In: *Queue* 16.3 (2018). DOI: 10.1145/3236386.3241340.

[66] David Baehrens et al. "How to explain individual classification decisions". In: *Journal of Machine Learning Research* 11 (2010), pp. 1803–1831.

[67] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. "Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models". In: (2017).

[68] Sebastian Bach et al. "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation". In: 10.7 (2015). DOI: 10.1371/JOURNAL.PONE.0130140.

[69] Quanshi Zhang et al. "Interpreting CNN Knowledge via an Explanatory Graph". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (2018).

[70] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. "Interpretable Convolutional Neural Networks". In: (2017), pp. 8827–8836.

[71] Lisa Anne Hendricks et al. "Generating Visual Explanations". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2016), pp. 3–19.

[72] Quanshi Zhang and Song-Chun Zhu. "Visual Interpretability for Deep Learning: a Survey". In: *Frontiers of Information Technology and Electronic Engineering* 19.1 (2018), pp. 27–39.

[73] Matthew D Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: (2013).

[74] Jost Tobias Springenberg et al. "Striving for Simplicity: The All Convolutional Net". In: *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings* (2014).

[75] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: *2nd International Conference on Learning Representations, ICLR 2014 - Workshop Track Proceedings* (2013).

[76] Aravindh Mahendran and Andrea Vedaldi. "Understanding Deep Image Representations by Inverting Them". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2014), pp. 5188–5196.

[77] Alexey Dosovitskiy and Thomas Brox. "Inverting Visual Representations with Convolutional Networks". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2015), pp. 4829–4837.

[78] Anh Nguyen et al. "Plug and Play Generative Networks: Conditional Iterative Generation of Images in Latent Space". In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* (2016), pp. 3510–3520.

[79] Bolei Zhou et al. "Object Detectors Emerge in Deep Scene CNNs". In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* (2014).

[80] Christian Szegedy et al. "Intriguing properties of neural networks". In: *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings* (2013).

[81] Jason Yosinski et al. "How transferable are features in deep neural networks?" In: *Advances in Neural Information Processing Systems* 4 (2014), pp. 3320–3328.

[82] Yao Lu. "Unsupervised Learning on Neural Network Outputs: with Application in Zero-shot Learning". In: *IJCAI International Joint Conference on Artificial Intelligence* (2015), pp. 3432–3438.

[83] Mathieu Aubry and Bryan C. Russell. "Understanding deep features with computer-generated imagery". In: 2015, pp. 2875–2883. DOI: 10.1109/ICCV.2015.329.

[84] Ruth Fong and Andrea Vedaldi. "Interpretable Explanations of Black Boxes by Meaningful Perturbation". In: 2017-October (2017), pp. 3449–3457. DOI: 10.1109/iccv.2017.371.

[85] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: (2017), pp. 618–626. DOI: 10.1109/ICCV.2017.74.

[86] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 13-17-August-2016 (2016), pp. 1135–1144.

[87] Luisa M Zintgraf et al. "Visualizing Deep Neural Network Decisions: Prediction Difference Analysis". In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings* (2017).

[88] Pieter-Jan Kindermans et al. "Learning how to explain neural networks: PatternNet and PatternAttribution". In: *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings* (2017).

[89] Devinder Kumar, Alexander Wong, and Graham W. Taylor. "Explaining the Unexplained: A CLass-Enhanced Attentive Response (CLEAR) Approach to Understanding Deep Neural Networks". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (2017), pp. 1686–1694.

[90] Jiawei Su, Danilo Vasconcellos Vargas, and Sakurai Kouichi. "One pixel attack for fooling deep neural networks". In: *IEEE Transactions on Evolutionary Computation* 23.5 (2017), pp. 828–841. DOI: 10.1109/tevc.2019.2890858.

[91] Pang Wei Koh and Percy Liang. "Understanding Black-box Predictions via Influence Functions". In: *34th International Conference on Machine Learning, ICML 2017* 4 (2017), pp. 2976–2987.

[92] Himabindu Lakkaraju et al. "Identifying Unknown Unknowns in the Open World: Representations and Policies for Guided Exploration". In: *31st AAAI Conference on Artificial Intelligence, AAAI 2017* (2016), pp. 2124–2132.

[93] Zhiting Hu et al. "Harnessing Deep Neural Networks with Logic Rules". In: *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers* 4 (2016), pp. 2410–2420.

[94] Quanshi Zhang et al. "Interpreting CNNs via Decision Trees". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2018), pp. 6254–6263.

[95] Quanshi Zhang et al. "Mining Object Parts from CNNs via Active Question-Answering". In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* (2017), pp. 3890–3899.

[96] Royston Goodacre. "Explanatory analysis of spectroscopic data using machine learning of simple, interpretable rules". In: (). DOI: 10.1016/S0924-2031(03)00045-6.

[97] Masashi Fukuhara et al. "Feature visualization of Raman spectrum analysis with deep convolutional neural network". In: *Analytica Chimica Acta* 1087 (2019), pp. 11–19. DOI: 10.1016/j.aca.2019.08.064.

[98] Bolei Zhou et al. *Learning Deep Features for Discriminative Localization*. Tech. rep.

[99] Xiaolei Zhang et al. "Understanding the learning mechanism of convolutional neural networks in spectral analysis". In: *Analytica Chimica Acta* 1119 (2020), pp. 41–51. DOI: 10.1016/j.aca.2020.03.055.

[100] Dario Bertazioli. "Decoding Raman Spectroscopy Towards the Diagnosis of SARS-COV-2 Infection and Other Diseases ." MA thesis. Università degli Studi di Milano-Bicocca – ITA, 2020.

[101] Michele Andrico. "Analisi e sviluppo di modelli di Deep Learning per l ' analisi e la caratterizzazione di spettri Raman". MA thesis. Università degli Studi di Milano-Bicocca – ITA, 2020.

[102] Rekha Gautam et al. "Review of multidimensional data processing approaches for Raman and infrared spectroscopy". In: *EPJ Techniques and Instrumentation* 2.1 (2015), pp. 1–38. DOI: 10.1140/epjti/s40485-015-0018-6.

[103] Darren Whitaker and Kevin Hayes. "A Simple Algorithm for Despiking Raman Spectra". In: *Chemometrics and Intelligent Laboratory Systems* 179 (2018), pp. 82–84. DOI: 10.26434/chemrxiv.5993011.v2.

[104] Esben Jannik Bjerrum, Mads Glahder, and Thomas Skov. "Data Augmentation of Spectral Data for Convolutional Neural Network (CNN) Based Deep Chemometrics". In: (2017).

[105] O. Al-Jowder, E. K. Kemsley, and R. H. Wilson. "Mid-infrared spectroscopy and authenticity problems in selected meats: A feasibility study". In: *Food Chemistry* 59 (1997), pp. 195–201. DOI: 10.1016/S0308-8146(96)00289-0.

A Details for the chapter 4

A.1 Detailed description of the dataset

Table A.1: Detailed description of the dataset used in this work.

Begin of description		
Categories	Patient	Number of spectra
Covid Negative	CovNeg 8020	25
	CovNeg 6120	25
	CovNeg 44420	25
	CovNeg 26	25
	CovNeg 44620	25
	CovNeg 22	25
	CovNeg 45620	25
	CovNeg 42920	25
	CovNeg 47420	25
	CovNeg 34	18
	CovNeg 46820	25
	CovNeg 48220	25
	CovNeg 19	25
	CovNeg 11	25
	CovNeg 45120	25
	CovNeg 16020	25
	CovNeg 16	25
	CovNeg 39820	25
	CovNeg 13	25
	CovNeg 18	25
	CovNeg 42420	20
	CovNeg 21	23
	CovNeg 41820	25
	CovNeg 41520	24
	CovNeg 12	25
	CovNeg 29	25
	CovNeg 25	17
	CovNeg 46720	25
	CovNeg 20	13
	CovNeg 17	25
	CovNeg 27	25
	CovNeg 23	20
	CovNeg 45020	20
	CovNeg 32	25
	CovNeg 43920	24

Continuation of Table A.1		
Categories	Patient	Number of spectra
	CovNeg 42220	24
	CovNeg 44120	24
Covid Positive	Cov 15	25
	Cov 03	24
	Cov 46920	25
	Cov 02	25
	Cov 05	25
	Cov 08	25
	Cov 24	25
	Cov 10	25
	Cov 45420	25
	Cov 01	25
	Cov 45720	25
	Cov 45920	25
	Cov 45520	25
	Cov 31	25
	Cov 09	25
	Cov 48520	25
	Cov 07	25
	Cov 47820	24
	Cov 43520	25
	Cov 47520	25
	Cov 30	25
	Cov 47020	25
	Cov 47720	25
	Cov 28	25
	Cov 42220	25
	Cov 36520	25
	Cov 47620	25
	Cov 44320	25
	Cov 04	24
	Cov 06	25
Control	C 06	20
	C 03	24
	C 08	20
	C 04	23
	C 01	14
	C 02	20
	C 05	24
	C 07	18
	C 18	25
	C 15	25
	C 27	25
	C 33	23
	C 14	21
	C 23	25

Continuation of Table A.1		
Categories	Patient	Number of spectra
	C 11	20
	C 17	25
	C 30	25
	C 09	22
	C 22	25
	C 19	25
	C 21	25
	C 10	25
	C 13	24
	C 34	14
	C 29	25
	C 32	23
	C 26	25
	C 16	24
	C 20	25
	C 35	25
	C 12	25
	C 36	25
	C 28	25
	C 31	25
End of Table		

A.2 Detailed results of the Convolutional Neural Network

	Accuracy	Recall		F1-score		Specificity	Sensitivity
Patient-Level	0.89	COV+ COV- CTRL	0.83 0.97 0.85	COV+ COV- CTRL	0.86 0.91 0.89	0.94	0.87
Spectra-Level	0.82	COV+ COV- CTRL	0.79 0.83 0.85	COV+ COV- CTRL	0.77 0.87 0.82	0.91	0.82

Table A.2: Detailed results of the original network, showing some of the most common measurements.

B Details for the chapter 7

B.1 Further examples of Class Activation Mapping proceeded on bacterial case

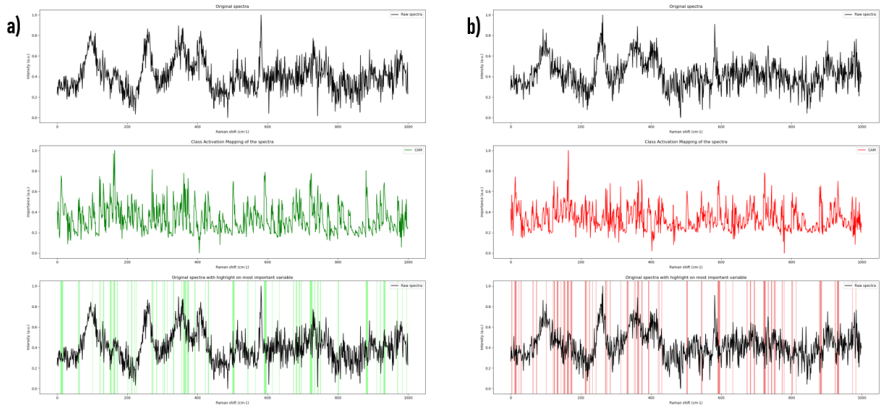

Figure B.1: Examples of class activation mapping results on a bacterial data set. **a)** Results for a correctly classified *E. Coli*. **b)** Results for a *K. pneumoniae* bacterium classified as an *E. Coli* bacterium. From top to bottom, the first spectrum is the raw spectrum, the second is the importance of each variable in the spectrum calculated using the Class Activation Mapping method and the last is the raw spectrum with the most important 10% of the spectrum highlighted.

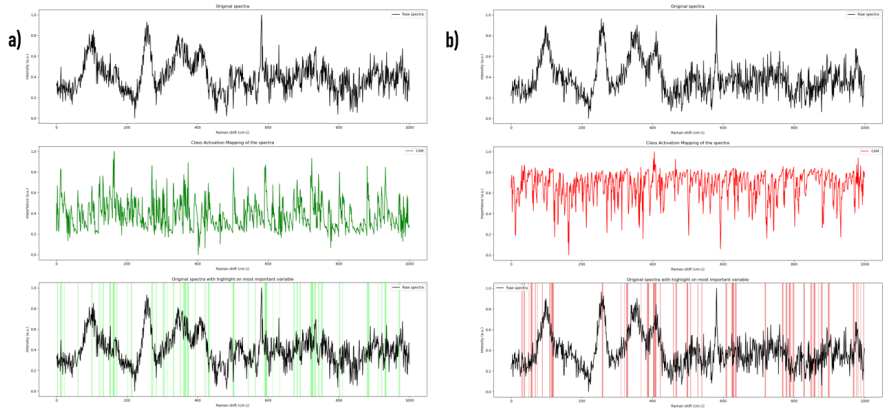

Figure B.2: Examples of class activation mapping results on a bacterial data set. **a)** Results for a correctly classified *E. Coli*. **b)** Results for a *E. cloacae* bacterium classified as an *E. Coli* bacterium. From top to bottom, the first spectrum is the raw spectrum, the second is the importance of each variable in the spectrum calculated using the Class Activation Mapping method and the last is the raw spectrum with the most important 10% of the spectrum highlighted. **This example is particularly interesting because it does not allow us to understand this misclassification.**

B.2 Further examples of Class Activation Mapping proceeded on covid case

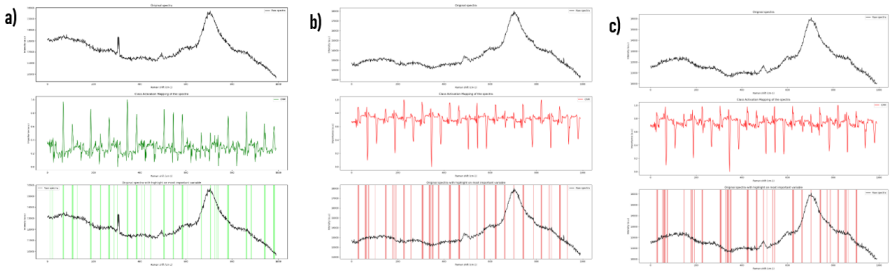

Figure B.3: Examples of class activation mapping results on covid data set. **a)** Results for a correctly classified Covid negative. **b)** Results for a Control classified as an Covid negative. **c)** Results for a Covid positive classified as an Covid negative. From top to bottom, the first spectrum is the raw spectrum, the second is the importance of each variable in the spectrum calculated using the Class Activation Mapping method and the last is the raw spectrum with the most important 10% of the spectrum highlighted.

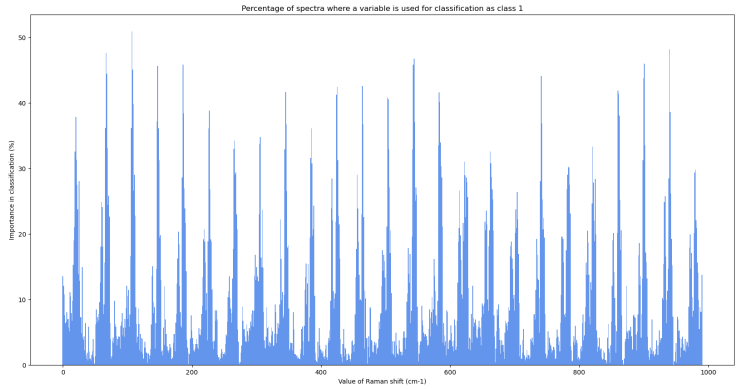

Figure B.4: Bar chart showing the percentage of spectra for which a variable is in the top 10% for the classification as Covid Negative. Blue bars represent variables used in less than 70% of classifications, and red bars represent variables used in at least 70% of cases.

B.3 Further examples of Gradient Class Activation Mapping proceeded on bacterial case

It is interesting to note that the case of misclassification presented in Fig. B.2 no longer exists with the use of the original full model.

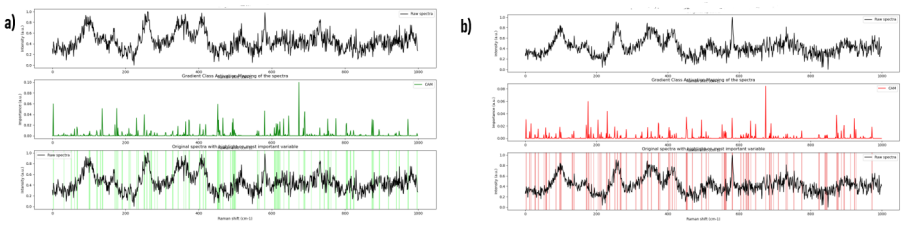

Figure B.5: Examples of gradient class activation mapping results on a bacterial data set. **a)** Results for a correctly classified *E. Coli*. **b)** Results for a *K. pneumoniae* bacterium classified as an *E. Coli* bacterium. From top to bottom, the first spectrum is the raw spectrum, the second is the importance of each variable in the spectrum calculated using the Gradient Class Activation Mapping method and the last is the raw spectrum with the most important 10% of the spectrum highlighted.

B.4 Further examples of Gradient Class Activation Mapping proceeded on covid case

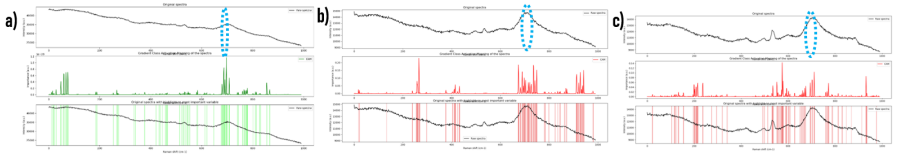

Figure B.6: Examples of gradient class activation mapping results on covid data set. **a)** Results for a correctly classified Covid negative. **b)** Results for a Control classified as an Covid negative. **c)** Results for a Covid positive classified as an Covid negative. From top to bottom, the first spectrum is the raw spectrum, the second is the importance of each variable in the spectrum calculated using the Gradient Class Activation Mapping method and the last is the raw spectrum with the most important 10% of the spectrum highlighted.

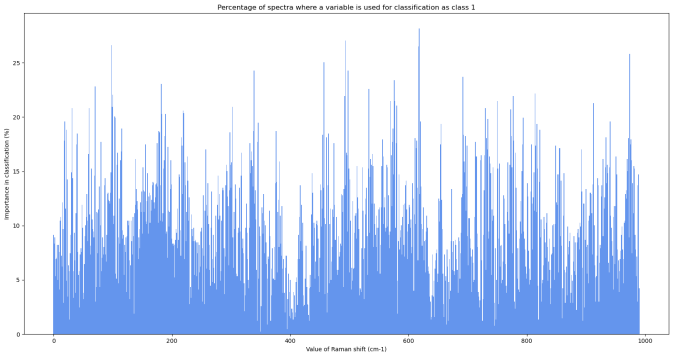

Figure B.7: Bar chart showing the percentage of spectra for which a variable is in the top 10% for the classification as Covid Negative. Blue bars represent variables used in less than 70% of classifications, and red bars represent variables used in at least 70% of cases.