UNIVERSITÉ LIBRE DE BRUXELLES

COMPUTER SCIENCES DEPARTMENT, FACULTY OF SCIENCES

MASTER THESIS

Methods for large-scale image classification and application to biomedical data

Quang Linh **BUI QUANG PHUONG**

ULB ID: 000427796

Promotor : Matthieu Defrance

Academic Year 2019-2020

Abstract

In this document, the important computer vision problem called image classification is tackled. To introduce the problem, different applications of image recognition and classification are given and explained to prove its importance in our actual world, especially in the medical field which will be focused in this paper. Furthermore, machine learning and deep learning methods are used to solve image classification problem. In 2012, convolutional neural networks (CNN) have significantly increase the performance of the previous learning methods. In addition to CNN, this document expounds support vector machines (SVM) which is a well-known method for classifying data into classes. Subsequently, these methods will be used to resolve medical images classification problems. In particular, the methodology following Regab et al.'s paper [10] to detect breast cancer by analysing mammogram images with an average accuracy of 80% makes subject of an entire chapter of the document.

Contents

Chapter 1

Introduction

1.1 Context and birth of the classification problem

1.1.1 Context

In our days, countless numbers of pictures are taken of everything and anything due to the exponential growing of photography and digital photo cameras. Whether it is thanks to the built-in camera of your smartphone or your faithful travel companion, taking a picture has become a disconcerting ease. All these pictures are inevitably shared and searched on the internet. Without any classification, it would be a really laborious task to find a particular picture over the millions (even billions actually!) photos taken by users worldwide. This classification problem can be extended in a large panel of fields: it can go from your personal use to a more impacting field such as medical images analysis. Overall, this problem can be generalized and divided into an image processing and recognition sub-problems. More applications are presented in the next section.

Thereby, fast and efficient methods have to be found to realize a reliable classification of big data. It is in the 50's, when the artificial intelligence (AI) has begun to take shape, that image recognition becomes a more serious problem to manage. As Marvin Minsky, co-founder of the Massachusetts Institute of Technology's (MIT) AI laboratory, defines:

> *"AI is the science of making machines do things that would require intelligence if done by men."* (1968)

Following this definition, image recognition and classification can clearly be resolved by AI methods, such as *Machine Learning* – which is a technique for teaching machines to learn, and indeed becomes one of the main application of AI.

1.1.2 History and first methods

The first methods used are based on *Artificial Neural Networks* (ANN) where the first artificial neuron is created in 1943. Very briefly, the basic idea behind a neural network is to simulate (i.e. copy in a good way) lots of densely interconnected brain cells inside a computer so you can get it to learn things and recognize patterns to finally take a decision as a human would do. This is exactly what we want to apply for an image. Giving on input an image, the neural network will

recognize and learn the different patterns of this input and will assign the image on a specific category where these patterns are retrieved. These patterns are also called **features**.

Nevertheless, it is in the 21st century that image classification and recognition has evolved faster and faster. Previously, the accuracy of the recognition was not very high, implying a somehow large rate of error (at least more than 20%).

In 2010, the appearance of *Deep Learning* – a class of machine learning algorithms that goes deeper into data to recognize patterns, accelerates the evolution of image classification problem and enhances this accuracy, especially due to a new class of neural networks called *Convolutional Neural Networks* (CNN) that will be tackled in details in the next chapters.

For instance, in 2012, Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton created a new recognition algorithm for objects images ensuring an 85% level of accuracy and in 2015, developed tools using CNN reaches a 95% level of accuracy for facial recognition [3].

1.2 Some application of image classification

1.2.1 Google Images

When thinking of image classification, Google Images is probably one of the most famous application of image classification. This extension of Google is the most used image search engine. At the beginning, Google's technology do basic things like extracting text from images, but its real power is in actually recognizing the objects in an image. Google says it has trained the algorithm to recognize "thousands" of different objects. Over time, Google has evolved and offers new features analysis such as sentiment and emotion detection.

Moreover, in 2013, Google released search for personal photos, giving users the ability to retrieve photos in their libraries based on the objects present in the images. Two years later, in 2015, this feature was implemented in Google Photos, the application of Google that stores all of your personal photos to ease your photos classification [13]. Following the AI's definition of Minsky, that proves that computer vision software could classify images to human standards avoiding annoying tasks to humans such as labelling manually all its photos. The models built to label images are even done to see pictures more like humans do. Indeed, the model is analysing every pixel of the photo and does not take in account if the object is in the foreground or in the background, the model will label it anyway as seen in the example given in Figure 1.1.

Figure 1.1: Example of Google Images' photo's labelling [13]

1.2.2 Face recognition

As you can imagine, no face can exactly be the same as another face. Every face has different features specific to each which can be the size of your mouth, the distance between your eyes or even the form of your face. This is how image classification is used in face recognition technology. These features are extracted and analysed to recognize and evaluate which face belongs to which person. As said, every face has different specific features which are characterized by **keypoints** which are essential to mark and extract. An example of keypoints extraction is given in Figure 1.2. The most used technique to extract these features is CNN which is, of course, not only used for facial recognition.

Facial recognition is used recurrently in various fields, here are some examples of face recognition applications:

- **Security and control**: face recognition systems help control access to personal devices, residences, vehicles, offices and other premises alike. It can also be used to secure online payments with face recognition confirmation.
- **Health**: some specific features of the face are an evidence of illnesses. These features could be detected with facial recognition.
- **Criminal identification**: as presented in detective movies, creating a photofit picture is a way to identify criminals to find him in the population database[1] thanks to face recognition.

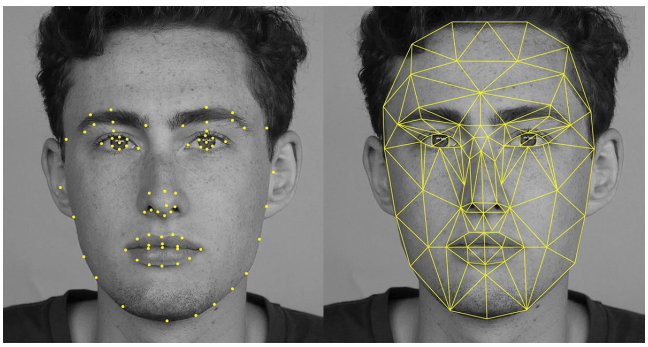


Figure 1.2: Keypoints marking to extract features for face recognition [9]

1.2.3 Musical images analysis

Actually, music is easily represented by images. It can be in the form of music score or spectral density, every song has its proper visual representation. Image recognition technology can be used to recognize the song obviously, but also to classify a large dataset of songs by genre. What was said about the presence of images in our daily life is also valid for music. Indeed, the music's market is growing and new genres are created over the multiple genres already existing. Applications that allows you to listen music (such as Spotify or Deezer) have to classify all their music library by genre. This is especially useful for the songs recommendations that the software is proposing to you following your music preferences or simply to find a specific song in the large library.

[1] Indeed, the FBI currently have a database which includes half of the national population's faces.

Concerning image processing, the overall procedure is the same as for simple images. the genre classification process of music has two main steps: feature extraction and classification. The first step obtains audio signal information, while the second one classifies the music into various genres according to extracted features. For instance, for music score analysis, one of the interesting features are the notes and their rhythm so that we have to create a model that recognizing musical notation as illustrated in Figure 1.3. To do that, it exists different methods that will be tackled further in the document such as CNN or Markov models.

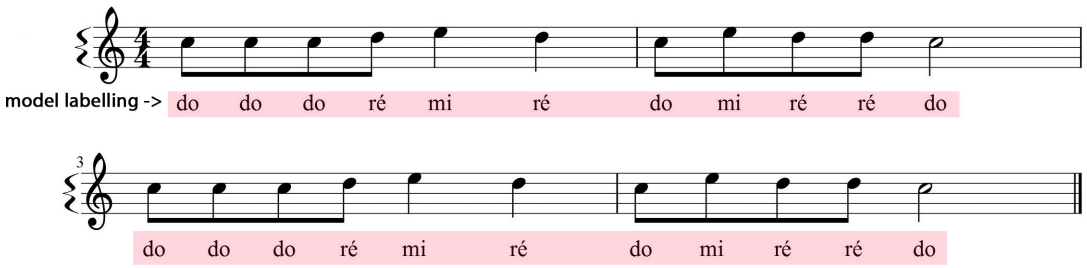


Figure 1.3: (French) Labelling of the different notes of "Au clair de la lune" 's music score

More generally, what is applied to music can also be applied to recognize sounds in general such as dog's bark, riding moto or human voice in particular thanks to their spectrogram representation. The following figure is using Deep belief network (DBL), a class of neural network, to recognize the audio.

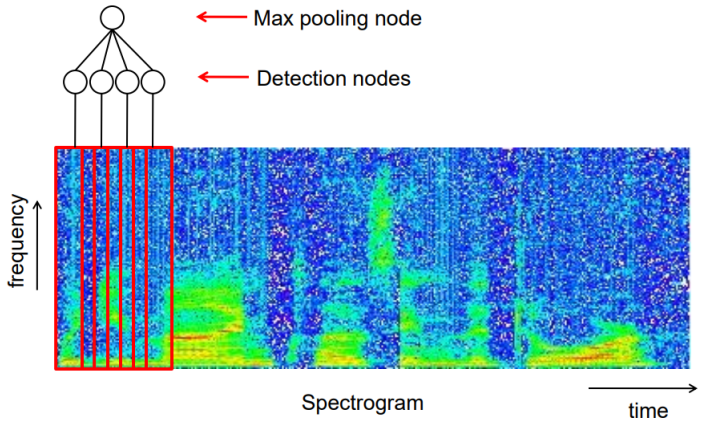


Figure 1.4: A sound spectrogram recognition using convolutional DBN [15].

1.2.4 Medical images analysis

In the medical field, different techniques of medical images exist which make the use of images quite abundant. The most popular one are radiography or ultrasounds scan, but much more imaging techniques exist such as magnetic resonance imaging (MRI) or histological images observed under a microscope. This huge amount of images generated is a precious source of information and constitutes an important training dataset to help to extract important and redundant features for future analysis. Thus, the existing methods for image classification can also used for medical image analysis. The main goal of these methods is to highlight anomalies that may help to detect diseases. Deep learning excels in perceptual tasks such as detection and segmentation. Here are some examples of different types of medical images analysis and their reason for being used.

1. **mammographic mass classification:** this is actually one of the most frequent disease for women over a certain age. The important thing is to detect the disease at starting point to treat it correctly. Thus, what we want to detect here is a center of a lesion, especially to detect malignant tumor which is a sign of breast cancer. To do that, the image includes different types of features that are used to raise the lesion such as contrast features that highlight a feature that is often mentioned by radiologists which is the distance that separates the lesion from the background. An other feature that is relevant to raise is the presence of holes in the candidate lesion which often decrease their suspiciousness, since tumours are solid, with possibly the exception of lobular carcinoma[2] [29]. This is just two types of features that is relevant to analyse mammographic mass but they are many more. In particular, breast cancer detection is subject of an entire part of the document (chapter 4).

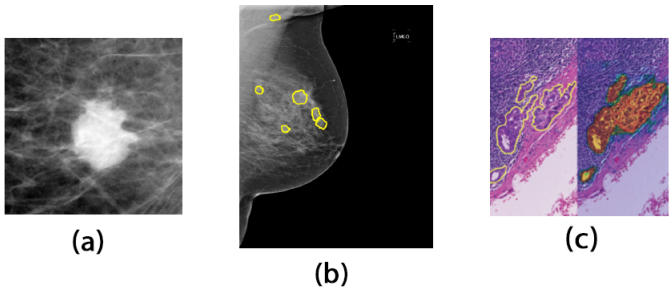


Figure 1.5: Mammographic mass images classification. (a, b) Segmentation of lesion, (a) is the illustration of the lesion on a smaller scale. [29] (c) breast cancer metastases detection in lymph nodes using an histopathology image.

2. **segmentation of lesions in the brain:** brain tumors can easily lead to death if they are not detected early and accurately to avoid disasters. Thus, brain tumors segmentation is an important challenge for medical purposes. Magnetic Resonance Imaging (MRI) is the medical imaging technique which is extensively used for diagnosis and treatment of brain tumors in clinical practice. Of course, these images provided by MRI includes different patterns of tumors which are interesting to collect to detect it automatically from deep learning and AI techniques.
   Usually, healthy brain tissue consists of three parts: gray matter, white matter, and cerebrospinal fluid. The segmentation is used to identify areas surrounded by a tumor by separating the active tumorous tissue from the necrotic tissue. This is done by identifying abnormal areas when compared to normal tissue [36].

[2]From the *National Cancer Institute* (NCI): Cancer that begins in the lobules (milk glands) of the breast. Lobular carcinoma is a noninvasive condition in which abnormal cells are found in the lobules of the breast. Lobular carcinoma rarely becomes invasive cancer, but having lobular carcinoma in one breast increases the risk of developing invasive cancer in either breast.

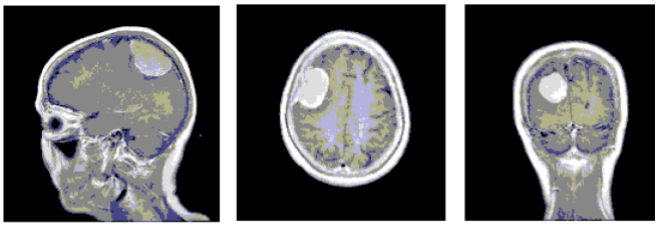


Figure 1.6: Brain MRI slices captured from different directions [36].

3. **diabetic retinopathy (DR) classification :** DR is an eye deficiency created by diabetes and can lead to blindness. The main purposes of this deficiency is due to the high-level of sugar in your blood which can lead to the blockage of the tiny blood vessels that nourish the retina, cutting off its blood supply. Classification methods and pattern recognition are tried to help analysing color fundus photographies (which is represented in the figure) to detect DR automatically and are actually in good progress. These classifying models are capable of quantifying the different features as blood vessels, fluid drip, exudates, hemorrhages and micro aneurysms[3] into different classes. All these features are signs of DR, especially micro aneurysms which is usually the first visible sign of diabetic retinopathy.
   Many filters are used to highlight these features such as edges detection which is the one of the feature extraction methods where the boundaries of the features are highlighted (i.e., the edges of the blood clots, white lesions, veins) are detected based on the sudden change in the pixel values with the neighborhood pixel intensity [27].

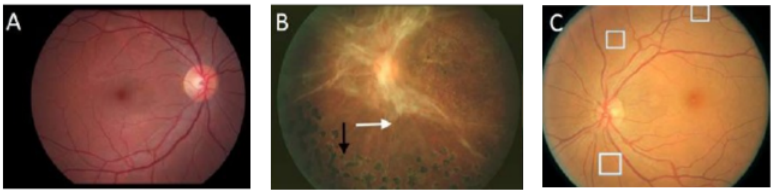


Figure 1.7: Representative retinal images of DR at various stages of the disease, as labeled: A-normal, B- end stage, C- early stage. Arrows in B point to pathological indications. White boxes in C enclose very small lesions that the CNNs have difficulty discerning [6].

However, the outcomes of these methods of image processing is not sufficient to give complete information about the image. The computer has not yet the knowledge of scientists and doctors which means that a human's intervention is necessary to complete the information given by the computer's outcomes. That is why the deep learning methods used are rather called **computer aided detection** (CAD) which is defined as systems that assist doctors in the interpretation of medical images.

[3] A micro aneurysm is a small swelling that forms in the wall of tiny blood vessels. These small swellings may break and allow blood to leak into nearby tissue.

Chapter 2

General concepts of AI

2.1 Machine learning and deep learning

We know that image classification is a problem resolvable by AI methods such as machine learning and especially deep learning. These methods are usually used to resolve data science problems such as image classification. Let's introduce and define them.

2.1.1 Machine learning

Definition 2.1.1. *Machine learning* is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves [30].

The main goal of machine learning is then to allow the computers **learn automatically** without human intervention to find a model that does an approximation of reality by doing predictions. Nevertheless, there exists different types of learning whose supervised and unsupervised learning – which are defined thereafter, where in the first case, data are pretreated by humans while in the second case it does not need it and use instead an iterative approach to review data and arrive at conclusions which is object of deep learning.

2.1.1.1 Supervised and unsupervised learning

Definition 2.1.2. *Supervised* learning is a type of system in which both input and desired output data are provided. Input and output data are labelled for classification to provide a learning basis for future data processing [22].

Example 2.1.1. A simple example for *supervised* learning. Given the dataset presented in Table 2.1, the machine can easily compute the output y of input x. If $x = 10$, the computer will output $y = 100$ by learning the pattern of squaring the input thanks to the provided dataset.

Input x	Output y
1	1
2	4
3	9
4	16

Table 2.1: Example of dataset for supervised learning

Note that in supervised learning, machine learning cannot be used without a good dataset. Before using machine learning to models and evaluates new data, we need to give him a relevant and useful set of data to allow him to learn correctly. If what we are interested for is a specific feature, the data should be adapted to highlight and label this specific feature to ease the learning of the machine. Once data are correctly pretreated, the machine will use this dataset called **training set** to learn from it and acquire knowledge about the data to apply it to new data.

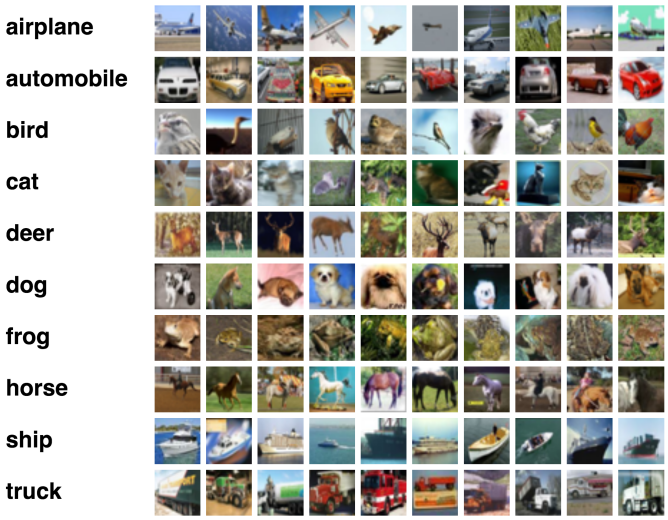


Figure 2.1: Example of training dataset : CIFAR-10

Definition 2.1.3. *Unsupervised* learning is the training of an AI algorithm using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance, i.e. data are unlabeled and the system's algorithms act on the data without prior training [23].

Example 2.1.2. Compared to supervised learning, in unsupervised learning **only inputs** are given as dataset which means that there is no output to match to. As simple example, given a group of people, we are trying separate them in different categories. The machine has different ways to classify them: their ethnicity, their hair colors, their gender, etc. But no one is giving information to the machine about the right way to divide the group. The machine has to learn without any supervisor. That is what unsupervised learning means. To resolve this, different methods of *clustering* algorithms exist and will be used to run through the data and find natural clusters if they exist.

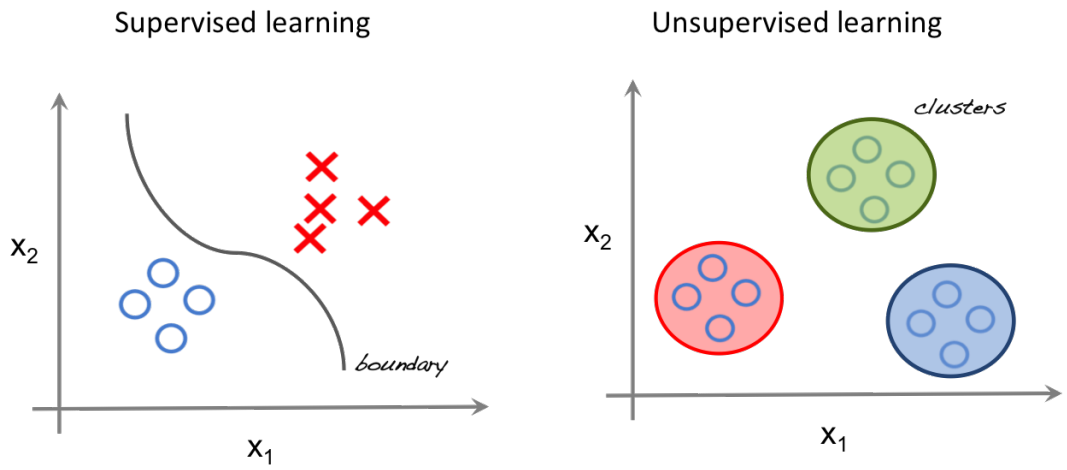


Figure 2.2: Supervised learning VS Unsupervised learning. In supervised learning, the separation and the categories are well defined while in unsupervised learning, clusters have to be found. [25]

2.1.1.2 Categories of machine learning

Machine learning is divided into 4 main categories which are classification, regression, clustering and dimensionality reduction. The two first are generally used for supervised learning while clustering is more often used for unsupervised learning. Regarding dimensionality reduction, its function is to transform data to highly dimensional data which could be used for both types of learning. Let's now introduce and define these terms.

Definition 2.1.4. *Classification* algorithms are used when the desired output is a discrete label. In other words, they're helpful when the answer to your question about your business falls under a finite set of possible outcomes [7]. When the set of possible outcomes is composed by only two elements, such as returning if an image has been forged or not, is called binary classification.

Definition 2.1.5. *Regression* is useful for predicting outputs that are continuous. That means the answer to the question is represented by a quantity that can be flexibly determined based on the inputs of the model rather than being confined to a set of possible labels [7]. We are trying to map input variables to some continuous functions.

Definition 2.1.6. *Clustering* is the assignment of objects to homogeneous groups (called clusters) while making sure that objects in different groups are not similar. As said, clustering is considered an unsupervised task as it aims to describe the hidden structure (features) of the objects [14] .

Definition 2.1.7. As its name said, *dimensionality reduction* is a process where its main objective is to reduce the number of random variables under consideration to treat smaller and specific data. The number of features under consideration decreases, where each feature is a dimension that partly represents the objects.

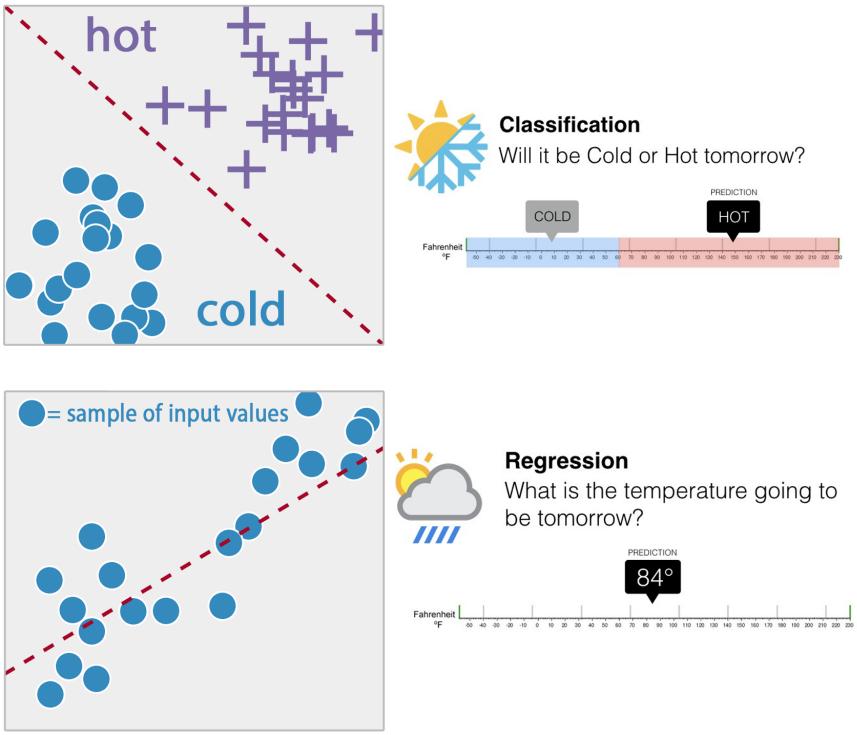


Figure 2.3: Difference between classification and regression on a weather example. For classification, we are doing a clear separation between two classes : hot and cold. For regression, we are trying to mean the degrees values given on input by a certain function.

2.1.2 Deep learning

Definition 2.1.8. *Deep learning* is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network [17].

Contrary to simple machine learning, deep learning is thus used to treat data that do not need to be pretreated and then used for unsupervised learning. To do that, deep learning will use artificial neural networks inspired by human's brain. This network is composed by a certain number of layers (which explains the term "deep" in "deep learning") where every layer receives and interprets informations of previous layer. To see more details, neural networks make object of section 3.1.

2.1.2.1 Differences between machine learning and deep learning

Of course, machine learning and deep learning are both used. Some difference make machine learning more useful than deep learning in a specific case and vice versa. Some of those difference are presented below:

- Machine learning (ML) uses statistical learning algorithms to treat labelled data while deep learning (DL) uses neural networks to treat unlabelled data.

- On one hand (ML), raw data are labelled manually by a data scientist to give a "clean" dataset to the machine. On the other hand (DL) , raw data are directly given to the machine which will uses its neurons to extract features.
- ML uses linear learning techniques while DL uses a non-linear architecture which its complexity is growing proportionally to the number of neurons.
- ... in summary, ML has a **supervisor** (human) to learn while DL does not have.
- But both ML and DL are able to handle massive dataset sizes, however, ML methods make much more sense with small datasets.

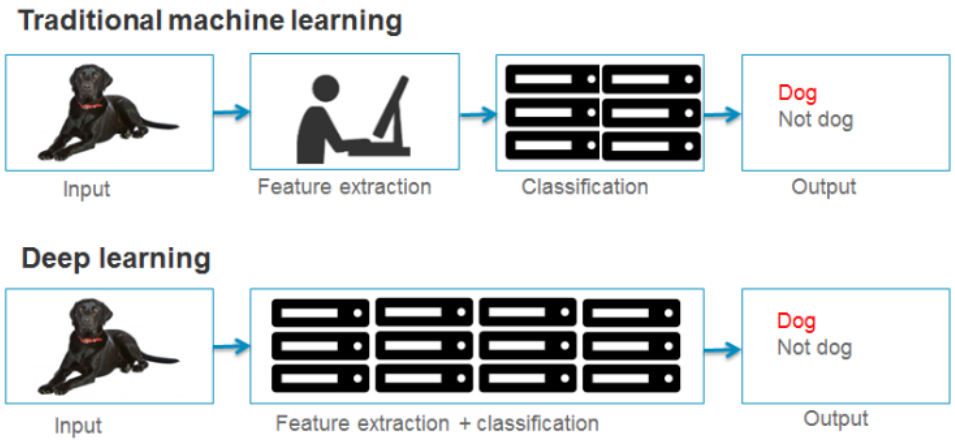


Figure 2.4: Illustration of the main difference between machine learning and deep learning

2.2 Definition of image classification problem

Recognize an animal from an object is a trivial task for humans but how about machines? A machine cannot assert with a 100% accuracy if an image contains such or such thing. That proves that classify images seems to be a complex task for machines. Thus, the main objective of image classification problem is to converge towards a perfect accuracy.

Basically, **image classification is the task of categorizing images into different classes**. This problem can be extended into other computer vision problems, such as localization, detection, and segmentation. However, it can either be a supervised learning problem or an unsupervised learning problem.

Definition 2.2.1. *Supervised image classification* is a problem where we have to define a set of target classes (for instance, objects to identify in images), and train a model to recognize them using labelled example photos [13]. The idea is that a user can select sample pixels in an image that are representative of specific classes and then softwares apply it on new images to retrieve features of these specific classes used as reference.

Definition 2.2.2. *Unsupervised image classification* is a problem where the outcomes (clusters of pixels with same features) are based on the software analysis of an image without the user providing sample classes. The computer uses techniques to determine which pixels are related and groups them into classes.

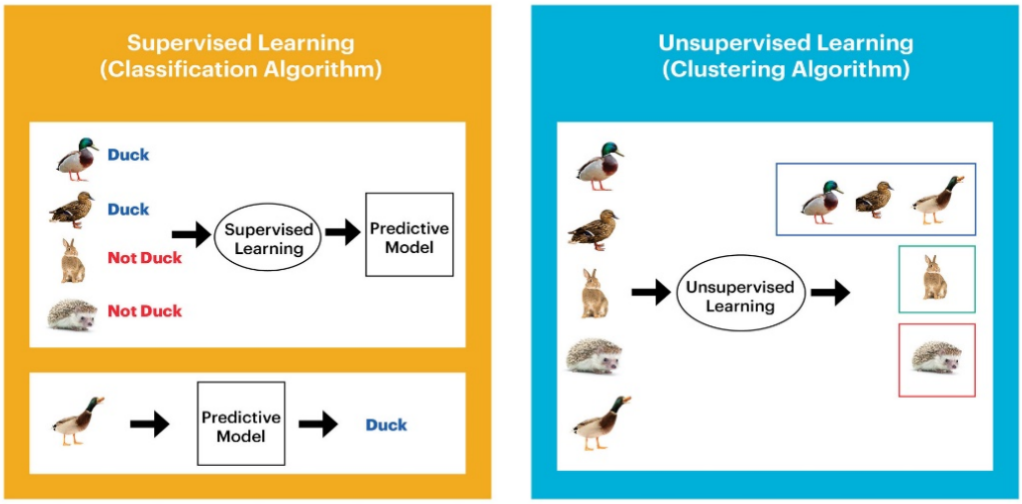


Figure 2.5: Supervised classification VS Unsupervised classification. [37]

2.2.1 Which methods should we use?

It exists different methods whether for supervised or unsupervised learning and all of them are useful. There exists no method better than an other one. Choosing the method depends of different criteria whose information missed, size of data, the insights you want to get from the data, etc. Generally, supervised learning is used to train a model to **make a prediction**. It can be used for instance for weather forecast and more generally for continuous and linear variables. On the other hand, unsupervised learning is employed to explore data and train a model to **find clusters** into the data. Some of the most familiar methods are *Support Vector Machine* (SVM), nearest neighbour, maximum likelihood, decision tree classifier for supervised learning. Concerning unsupervised learning, K-means clustering algorithm or neural networks are the most common methods.

Chapter 3

Materials and Methods

In this chapter, different methods of learning are introduce and presented. In particular, neural networks is the main and most known method for images classification especially convolutional neural networks (CNN). In a second time, support vector machines (SVM) models are presented to associate learning algorithms that analyse data used for classification and regression analysis. Finally, we take interested of the evaluation methods: how do we evaluate the classification? From which percentage is the evaluation correct? Therefore, this evaluation leads to a question of validation to define how do we validate a prediction.

3.1 Neural networks

When speaking about deep learning, the most known technique is probably neural networks. Formally, the simplest definition of a neural network, more properly referred to as an 'artificial' neural network (ANN), is provided by the inventor of one of the first neurocomputers, Dr. Robert Hecht-Nielsen:

> *"A neural network is a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs."*

This definition is somehow explaining how a human's brain work and this is mostly why a ANN is inspired of to recognize patterns. To imitate the behaviour of a human's brain, let's see first how does a brain works. Basically, the biological neural network is composed of four main elements: somas, dendrites, axons and synapses. For a better understanding, we have to define their respective functions to create elements that will act similarly in our ANN.

- **Soma:** central part of a neuron, contains the cell nucleus which mainly contains all of the cell's genome (DNA). In ANN, this should correspond to the neurons.
- **Dendrite:** projection of a neuron that receive signals (information) from other neurons. In ANN, this should correspond to the inputs.
- **Axon:** carries nerve impulses away from the cell body. A neuron typically has one axon that connects it with other neurons or with muscle or gland cells. In ANN, this should correspond to the outputs.

- **Synapse:** transfer electric activity (information) from one cell to another. In ANN, this should correspond to the links between every neuron and their respective weights which correspond to the strength or amplitude of a connection between two nodes. Besides, that's why this weight is also called *synaptic weight*.

To sum up, we can associate these biological elements to artificial ones following Table 3.1. The representation of both neural networks are shown in Figure 3.1.

Biological NN	Artificial NN
Soma	Neuron
Dendrite	Input
Axon	Output
Synapse	Weight

Table 3.1: Association of elements from biological neural network to articial neural network.

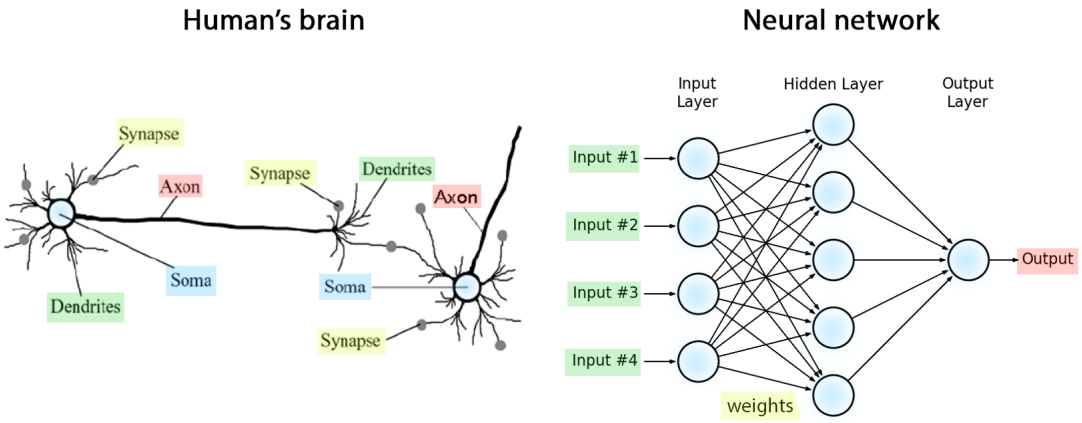


Figure 3.1: Comparison between a biological neural network and an ANN. The same color associates elements with the same function in a biological and artificial point of view.

3.1.1 How does neural network works?

Typically, a neural network is initially trained or fed large amounts of data. Training consists of providing input and telling the network what the output should be. The objective is to obtain a neural network that is trained and generalizes well that should behaves correctly on new instances of the learning task. To do that, we need an architecture of neural network composed by a set of neurons and links connecting neurons such that each link has a synaptic weight. Multiple neural network architectures exist such as single-layer or multi-layer perceptron. Inevitably, we need a learning algorithm to learn. Linear regression, logistic regression, decision tree or SVM are some examples of learning algorithms. The last one cited make object of section 3.2.

Generally, a neural network is represented by the scheme presented below in Figure 3.2.

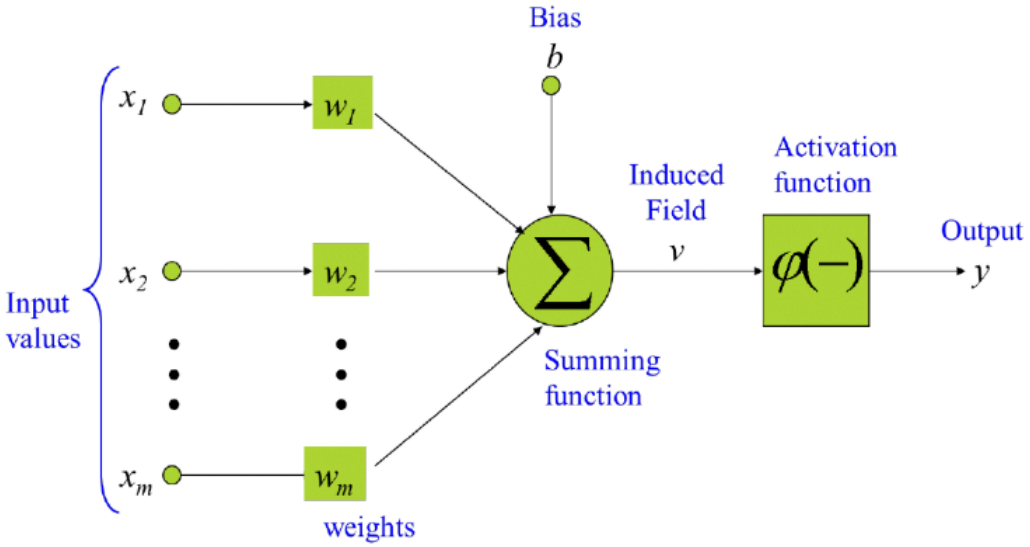


Figure 3.2: General representation of a neural network [16].

Firstly, input values are given to the neural network. In the case of an image processing, the input will be given by each pixel of the image. For instance, if we try to process an image of 200x200 pixels, the input values will be defined by the set $\{x_1, x_2, ..., x_{39999}, x_{40000}\}$ where x_i is a value that describes the level of a certain property (luminosity, contrast, ...) on the pixel. All these informations are given to **neurons**.

Neurons are functions consisting of a set of links described by synaptic weights which are computed by a summing (adder) function:

$$v_j = \sum_{i=1}^{n} x_i w_{ij}$$

where n is the input number, x_i the value of neuron i and w_{ij} the weight between neuron i and neuron j. v_j is called the local induced field of the neuron.

Finally, this induced field added to a bias value b are given to an activation function ϕ (also called decision function) to determine the output value:

$$y = \phi(v + b)$$

Many types of activation function exist but in general, a sigmoid (and its derivative) is used for the statistical interpretation and is the most widely decision function used. The sigmoid is defined:

$$\phi(x) = \frac{1}{1 + e^{-}x}$$

and its derivative:

$$\phi'(x) = \phi(x)[1 - \phi(x)]$$

The reason that sigmoid is one of the most used activation function is that it is a non-linear function which allow then to stack layers and to treat non-binary activations. Moreover, the output of the activation function is always going to be in range (0,1) compared to (-∞, ∞) of linear function. Thus, an output with a value approaching 1 means that the prediction is most likely focused on this solution.

3.1.2 Convolutional neural networks

Concerning image classification and recognition, traditional multilayer perceptron (MLP) models were a successfully method used. Nevertheless, due to the full connectivity between nodes they suffer from the curse of dimensionality, and thus do not scale well to higher resolution images. For example, a 1000×1000-pixel image with RGB color channels has 3 million weights, which is too high to feasibly process efficiently at scale with full connectivity [32]. For this reason, scientists had to find an efficient solution to remedy this situation. That's why in the end of the 80's, convolutional neural network (CNN) were created.

CNN is a type of neural network mostly applied for visual tasks such as image or video recognition. A convolution is the simple application of a filter to an input that results in an activation. Thus, we are applying a filter of a fixed size $N \times N$ to every window of size $N \times N$ of the input image a number of times needed to filter the whole image. This filter is used to highlight the feature that we are interested in. CNN basically uses the same structure as a basic neural network: input, hidden and output layers. However, several difference occur especially for hidden layers. Indeed, the hidden layers of a CNN typically consist of convolutional layers, pooling layers and fully connected layers as shown in Figure 3.3 and detailed in the next sections.

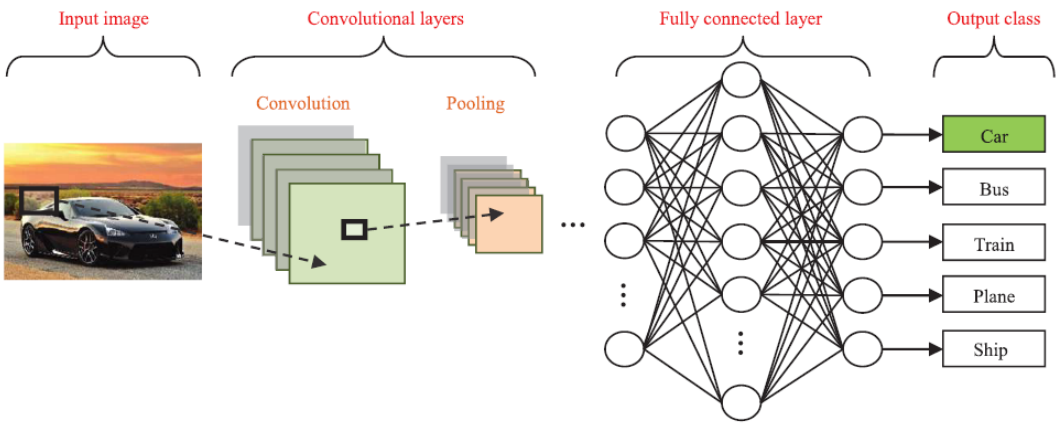


Figure 3.3: CNN image classification pipeline [31].

3.1.2.1 Convolutional layers

The objective of convolutional layers is to extract the high-level features such as edges, by learning features representations of the input image with shape width x height x depth. As said previously, a convolution is simply applying a filter, also called kernel K, to the image. A simple example of convolution using $K = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$ is shown in Figure 3.4 where we are applying the filter to every window of size 3x3 of the 5x5 image. Thus, the filter will be applied and shifted 9 times.

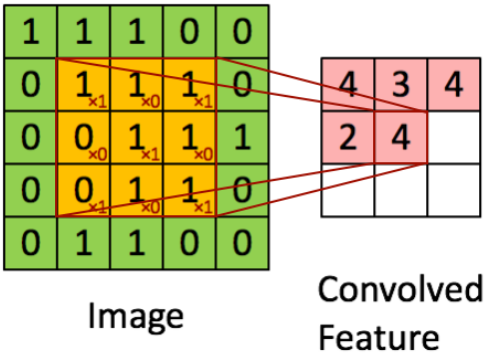


Figure 3.4: Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature.

The neurons in the convolutional layers are then arranged into feature maps. As ANN did, every neuron is connected to a neighborhood of neurons in the previous layer via a set of trainable weights. Inputs are then convolved following these weights in order to compute a new feature map which will be finally sent to an activation function. The kth output feature map Y_k can be computed as [31]:

$$Y_k = f(W_k * x)$$

where $f()$ is the nonlinear activation function, W_k is the convolutional filter related to the kth feature map and x the input image.

3.1.2.2 Pooling layers

Another novelty compared to traditional neural network is the pooling layers. Pooling layers have a dimensionality reduction function to decrease the computational power required to process the data. Same as convolutional layers, reducing dimensionality is a way to detect patterns. The two most known and used pooling types are max pooling and average pooling. Their names are very explicit. Max Pooling returns the maximum value from the portion of the image covered by the filter while average pooling returns the average of all the values from the portion of the image covered by the filter.

But which type of pooling is better? Should I use average or max pooling? Initially, the most common choice was the average pooling. This method smooths out the image and hence the sharp features may not be identified when this pooling method is used. Hence, average pooling simply performs dimensionality reduction as a noise suppressing mechanism by measures the mean value of existence of a pattern in a given region.

On the other hand, max pooling discards the noisy activations altogether and also performs denoising along with dimensionality reduction. This method is then sensitive to existence of some pattern in pooled region and roughly says "this type of feature was detected in this general area".

Therefore, there is no pooling method better than an other. The choice of pooling operation is made based on the data.

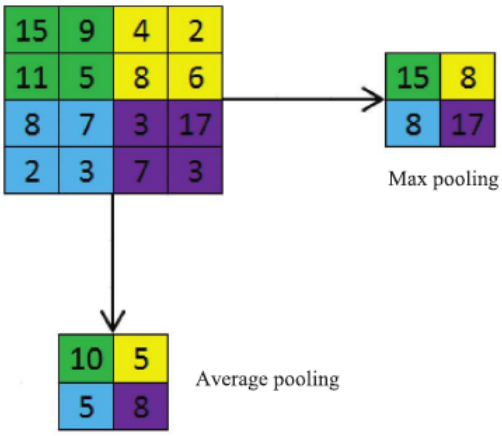


Figure 3.5: Average pooling vs Max pooling [31].

3.1.2.3 Fully connected layers

Now that we can detect these high level features thanks to the convolutional and pooling layers, the fully connected layer follows to interpret these feature representations. The structure of the fully connected layer is the same as a MLP: every neuron in one layer is connected to every neuron in another layer. The way this fully connected layer works is that it looks at the output of the previous layer (which as we remember should represent the activation maps of high level features) and determines which features most correlate to a particular class by mostly using the *Softmax* classification technique. The softmax classifier is defined:

$$L = \frac{1}{N}\sum_{i=1} L_i = \frac{1}{N}\sum_{i=1} -log\left(\frac{e^{f_{yi}}}{\sum_{j=1}^{K} e^{f_j}}\right)$$

where x_i is the ith input feature with corresponding label y_i, K the number of classes, N the amount of training datas and f the softmax function defined by:

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

where z is a vector of arbitrary real-valued scores.

Therefore, it is at fully connected layers that the model is classifying the image and gives the prediction of what is represented in with a certain probability computed using weights.

3.2 Support Vector Machine

Support vector machines (SVM) are supervised learning models that are efficient to resolve classification problems or regression problems[1]. Basically, in the first case, we talk about *support vector classification* (SVC) while we talk about *support vector regression* (SVR) in the second case. SVM is based on the idea of finding an **hyperplane** that best separates the features into different domains.

3.2.1 Data structure and linearity

The most known type of SVM is the *linear kernel SVM* which the separation between the different domains is a straight line (or hyperplane if working in multidimensional plans). Nevertheless, it is the very basic case which is only usable for data which are linearly separable (as the example shown in Figure 2.3) but not for non-linear data. In this case, in 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the **kernel trick** to **maximum-margin hyperplanes** [34].

3.2.1.1 Margin maximisation

Maximising the margin, which corresponds to the distance between the closest points (also called *support vector points*) and the hyperplane, can be interpreted as maximising the confidence of the model. In other words, more the SV points is far from the hyperplane, more is the probability of correctly classifying the points in their respective region or classes. Therefore, we can denote the importance of the SV points positions which has an direct impact to the hyperplane.

As a reminder for the simplest case, given $a \in \mathbb{R}$ the slope and $b \in \mathbb{R}$ the y-intercept, the equation form of a linear separation y is:

$$y = ax + b \tag{3.1}$$

Concerning the hyperplane π, given x an input vector, w the margin width and b the bias, the form of equation defining the decision surface separating the classes is a hyperplane π of the form:

$$\pi \equiv w^\top x + b = 0 \tag{3.2}$$

Moreover, we can define π_1 as the hyperplane of the first class y^- (green) parallel to the decision border hyperplane π and π_2 as the hyperplane crossing the SV point of the second class y^+ (red) parallel to π. The margin is then defined by $\pi_2 - \pi_1$. Knowing the hyperplane defined by Equation 3.2, we can define π_1 and π_2 as:

$$\pi_1 \equiv w^\top x + b = -1 \tag{3.3}$$

$$\pi_2 \equiv w^\top x + b = 1 \tag{3.4}$$

By solving algebraically $\pi_2 - \pi_1$, we obtain:

$$\begin{aligned} \pi_2 - \pi_1 &= (w^\top x + b - 1) - (w^\top x + b + 1) \\ &= \frac{2}{\mid w \mid} \end{aligned}$$

[1] Defined in subsubsection 2.1.1.2

Thus, to increase the margin, we have to minimize $\mid w \mid$. In other words, with the condition that there are no datapoints between π_1 and π_2:

$$x_i \cdot w + b \geq +1 \Rightarrow y_i = +1 \Rightarrow y^+$$
$$x_i \cdot w + b \leq -1 \Rightarrow y_i = -1 \Rightarrow y^-$$

The following pseudo-code sums up the procedure, so either we save the w and b values and keep going or we adjust the parameters, which corresponds to a convex optimization problem which surely gives us global minimum value, and keep going. Moreover, Figure 3.6 illustrates the problem and all the variables.

Algorithm 1 Margin Maximization - Finding the best hyperplane

for each point x_i we check: **do**
    if $y_i \cdot (w \cdot x_i + b) = 1$ **then**
        $x_i =$ SV point
        Correctly classified $\Rightarrow$ Save parameters b and w
    else if $y_i \cdot (w \cdot x_i + b) > 1$ **then**
        Correctly classified $\Rightarrow$ Save parameters b and w
    else
        Incorrectly classified $\Rightarrow$ Adjust parameters b and w

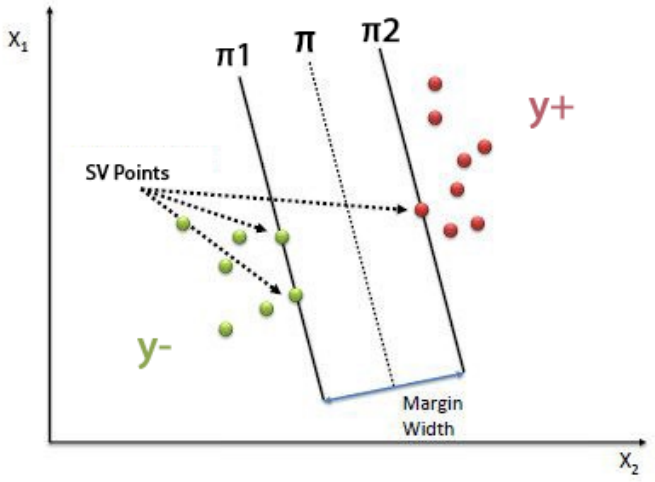


Figure 3.6: SVM hyperplane and margin representation [35].

3.2.1.2 Kernel trick

Let's now define the kernel trick and how does it work. Basically, the kernel trick is used by a *radial basis function* (RBF) SVM where this trick will allow us to use a linear classifier to resolve a non-linear problem by transforming data into another higher dimension (potentially infinite) that has a clear dividing margin between classes of data. The kernel trick is then a way of computing the scalar product of two vectors x and y in some (very high dimensional) feature space.

More formally, given an input vector x labelled by y. We apply a non-linear transformation $\phi : \mathbb{R} \rightarrow H$ to the data where H is an *Hilbert space*[2]. After this transformation, the procedure only depends of the scalar product $\phi(x) \cdot \phi(y)$ but computing this scalar product in a high-dimensional space is somehow complicated. The idea of the kernel trick is then to use a kernel function K of the form: $K(x, y) = \phi(x) \cdot \phi(y)$. Simply, applying kernel trick means just to the replace scalar product of two vectors by the kernel function.

Concerning RBF SVM, the kernel function is defined as:

$$K(x, y) = exp(-\gamma \left\| x - y \right\|^2)$$

where γ defines how far the influence of a single training example reaches, with low values meaning "far" and high values meaning "close". Moreover, RBF SVM implies a cost parameter c which defines a penalty value when a datapoint is not correctly classified. For larger values of c, a smaller margin will be accepted if the decision function is better at classifying all training points correctly. A lower c will encourage a larger margin, therefore a simpler decision function, at the cost of training accuracy.

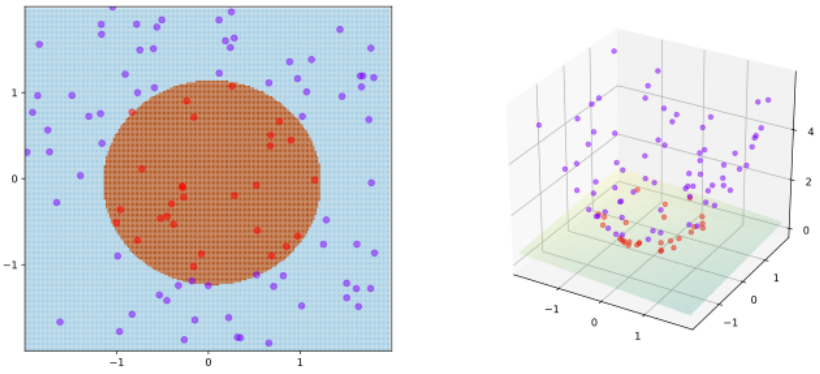


Figure 3.7: A training example of SVM with kernel given by $\phi((a, b)) = (a, b, a^2 + b^2)$ and thus goes from a 2D space to a 3D space defined by $K(x, y) = x \cdot y + x^2 y^2$. The training points are mapped to a 3-dimensional space where a separating hyperplane can be easily found. [34].

[2] An *Hilbert space* is a real vectorial space including a scalar product, which allow us to measure lengths, angles and to define an orthogonality.

3.3 Evaluation and validation

Once classification models are done, it is mandatory to evaluate the prediction to determine potential errors that should falsify the obtained results. To do that, it is useful to know which type of error exist and how are we computing the performance of a classification model to finally validate the prediction or not. More precisely, we call the statistical measures of the performance of a binary classification test "sensitivity" and "specificity".

3.3.1 Sensitivity, specificity and confusion matrix

A confusion matrix is an $N \times N$ matrix that is often used to summarize the performance of a classification model. The confusion matrix gives information about what the model is predicting right and what error it did. These two informations are separated in 4 categories: True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). The two first one are the results where reality is in favor with the model predictions while FP and FN define the two types of errors that the model can encounter.

Given the confusion matrix, we can compute the accuracy of the model to show how often the classifier was correct following this formula:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

A variant of accuracy is the Positive Predicted Value (PPV), also called *precision*, that computes how often the model was correct when it predicted positively:

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

The two other values interesting to compute are the **sensitivity** and **specificity** of the model.

The sensitivity, also called *true positive rate* or *recall*, shows how often the model correctly detect a true response while it is the case. It increases by reducing the number of FN. When the number of FN is null, it means that the positive response of the model is always verified. Likewise, a negative response asserts that the positive case is impossible. In summary, when the model has a 100% sensitivity, it means that the model is predicting perfectly. Mathematically, the sensitivity is computed by:

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

The specificity, also called *true negative rate*, is simply the opposite of sensitivity. It shows how often the model predicted negatively when the response in reality is negative too. Similarly, the specificity is increased by reducing the number of FP.

$$specificity = \frac{TN}{TN + FP}$$

Moreover, it exists a value, called F1 score, that combines recall and precision which indicates the weighted average of both elements. It is usually used to rate classifier performance and hence take into account FP and FN.

$$\text{F1 score} = \frac{2 * recall * precision}{recall + precision}$$

3.3.1.1 Application

To illustrate these terms and the use of the confusion matrix, let's take an example of a medical testing where the model is trying to predict if a patient has a disease or not. Roughly, there exist 4 possible results and evaluations:

TP	the patient has the disease and the test is positive
TN	the patient does not have the disease and the test is negative
FP	the patient does not have the disease but the test is positive
FN	the patient has the disease but the test is negative

Taking a sample of 100 people, the current model predicts 20 people to have the disease but only 15 people have actually disease and among those 15 people, the model predicted 12 correctly. These results put in the confusion matrix gives what is illustrated in Figure 3.8.

Moreover, we can compute the different values tackled previously to evaluate the model.

accuracy	89%
precision	60%
sensitivity	80%
specificity	90%

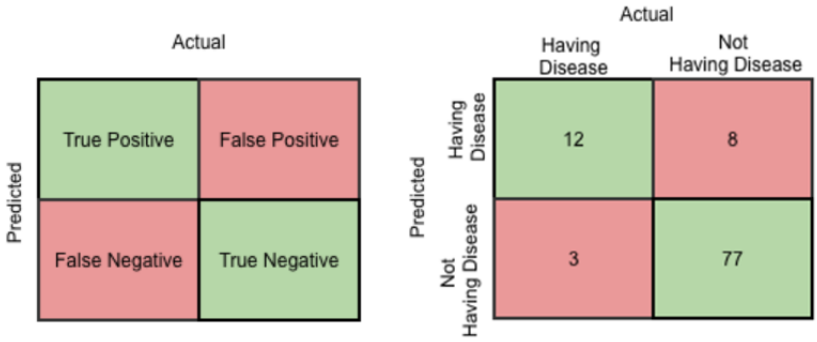


Figure 3.8: A confusion matrix prototype and an application to the above example. $TP = 12$, $FP = 8, TN = 3, TN = 77$.

3.3.2 ROC curves

The *Receiver Operating Characteristic* curve (ROC curve) is a plot of the true positive rate (TPR, sensitivity) against the false positive rate (FPR, corresponding to 1-specificity) for different classification thresholds. In other words, ROC curves can be used to select a threshold for a classifier which maximizes the TP while minimizing the FP. A perfect ROC curve would follow the two axes. In general, real-life classification rules produce ROC curves which lie between these two extremes.

Given a ROC curve, it is interesting to compute what we call the *Area Under The Curve* (AUC) which represents the degree or measure of separability, i.e. how much the model is capable of distinguishing between classes. If $AUC = 1$, it means that the test has perfect accuracy.

For instance, in the example tackled above in subsubsection 3.3.1.1 , higher the AUC is, better the model is at distinguishing between patients with disease and patients with no disease.

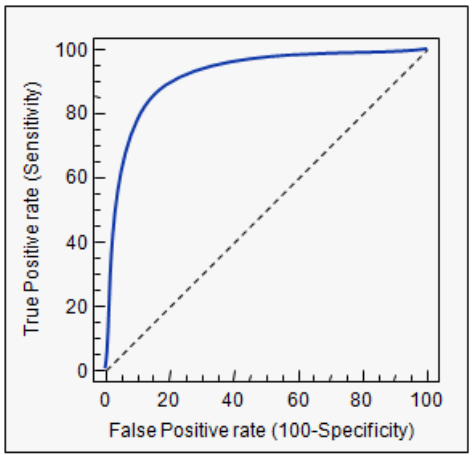


Figure 3.9: A ROC curve. The blue line denotes the change of TPR with different FPR for a model. The AUC is described by the area under this blue line. [20]

3.3.3 Overfit, underfit and cross-validation

A frequent problem in model training is when the fitting is too low or too high which implies a lack of information or in contrary a too large value of information to learn which may lead to noise. The first case is called *underfitting* while the second case is called *overfitting*. Finding a happy medium is then an important case to manage. This value is also called *goodness of fit* which refers to how closely a model's predicted values match the observed (true) values.

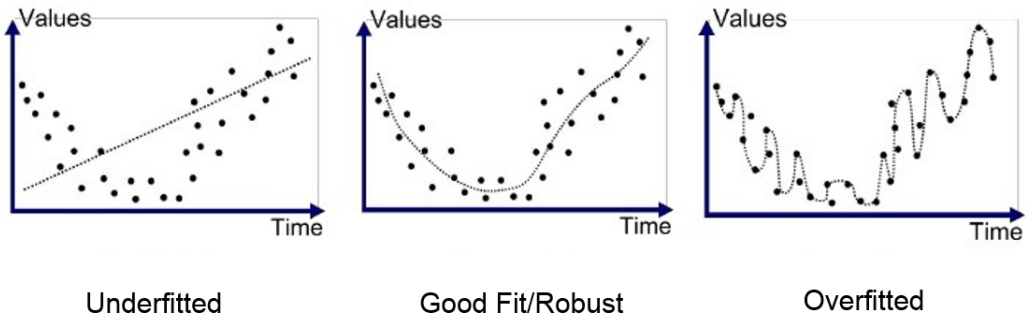


Figure 3.10: Illustration of an underfit, a good fit and an overfit. [4]

3.3.3.1 Overfitting

Generally, more you learn, the better it is. But in case of learning to predict, it can cause troubles. Why would learn a large number of information/features be a bad thing? The objective is to find a model that **generalizes** well certain patterns to finally apply these generalisations to new data. Therefore, while learning too much data, the model may save too much superfluous and too detailed information which are only specific to the images that the model is currently learning. In other words, the predictive model will capture the generalizable correlations and the **noise** produced by the data.

In summary, we can define overfitting as a model that is too specialized on training set data and that will not generalize well. That implies that the model will predict very well on training

set data but bad on new datasets. Therefore, to detect overfitting, we see if the model predict much better on the training set (used to learn) than on the test set (that is never touched before prediction), if it is the case, then we are likely overfitting.

3.3.3.2 Underfitting

Underfitting problem is more meaningful to everyone. This is exactly the contrary to overfitting which means that not enough information are learnt to predict new datasets correctly, even for the training set. In this case, the model suffers from a large bias and then will not be viable for any dataset.

3.3.3.3 Cross-validation

To prevent those problems, cross-validation (fully called *k-fold cross-validation*) is one of the most known method which is estimating the skill of machine learning models. In standard k-fold cross-validation, we partition the data into k subsets, called folds. Then, we iteratively train the algorithm on $k-1$ folds while using the remaining fold as the test set. In other words, one round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set) [33]. This way, we generate multiple mini train-test splits (then, the training tests number) from the initial training data to multiply the chance to detect and prevent from over or underfitting.

The choice of k is usually 5 or 10. But as k gets larger, the difference in size between the training set and the resampling subsets decreases which implies the reduction of the bias of the technique.

Of course, cross-validation is not the only way to prevent from over and underfitting. There exists much solutions such as simply train with more data or remove some features.

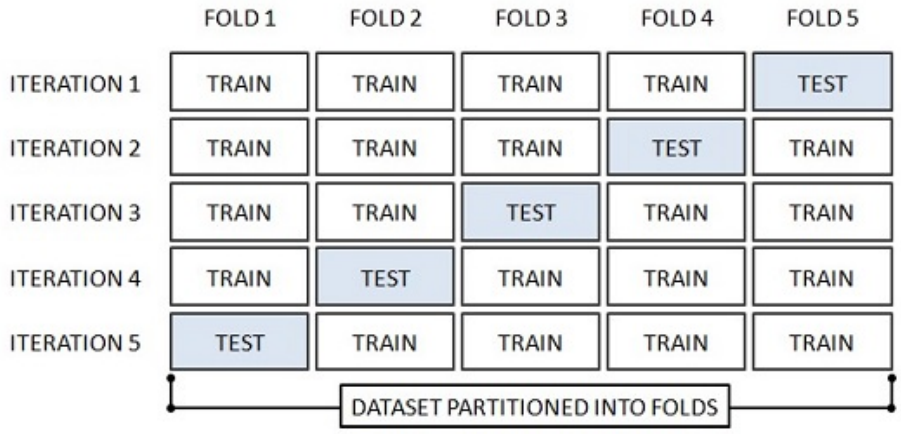


Figure 3.11: Illustration of 5-fold cross-validation [19].

Chapter 4

A well-known application: breast cancer detection

One of the most recurrent serious disease which is one of the leading for women globally is breast cancer. To overcome this disease, mammography is one of the important methods which use medical imaging to detect breast cancer. This chapter is devoted to the state-of-the-art of the article "Breast cancer detection using deep convolutional neural networks and support vector machines" [10] written by Dina A. Ragab, Maha Sharkas, Stephen Marshall and Jinchang Ren published recently in January 2019. We will study the methodology adopted by the authors to detect breast cancer through magnetic resonance imaging and present the results that they have obtained. These results are obtained using the digital database for screening mammography (**DDSM**) and the Curated Breast Imaging Subset of DDSM (**CBIS-DDSM**) that are publicly available datasets.

4.1 Biological features

To detect breast cancer, it is essential to detect the indicators of it, i.e. tumors. Globally, the two most important and most significant signs of the disease are *masses* and *microcalcifcations* (MCs).

Definition 4.1.1. A mass is a lump in the body and may be caused by the abnormal growth of cells, a cyst, hormonal changes, or an immune reaction. It can be either benign or malignant [18].

Definition 4.1.2. A MC is a tiny deposit of calcium in the breast that cannot be felt but can be detected on a mammogram. A cluster of these very small specks of calcium may indicate that cancer is present [18]. Same as masses, MCs can either be benign or malignant.

4.1.1 Benign and malignant tumors

In two words, the difference between benign and malignant tumors is that the meaning of malignant is cancerous and the meaning of benign is non-cancerous but they have each one specific features. The benign tumors won't invade nearby tissues or spread to other areas of the body and are characterized for breast cancer detection by round or oval shapes. Nevertheless, even if at the start they are non-cancerous, it may lead to malignant tumors which denotes the importance of removing it.

On the other hand, malignant tumors can invade nearby tissues. In the case of breast cancer, the tumor begins in breast tissue but could spread to lymph nodes which may lead the cancer

cells to other areas of the body, like the liver or bones. In this case, the malignant tumors are characterized by partially rounded shape with an irregular outline and will appear whiter than any tissue surrounding it.

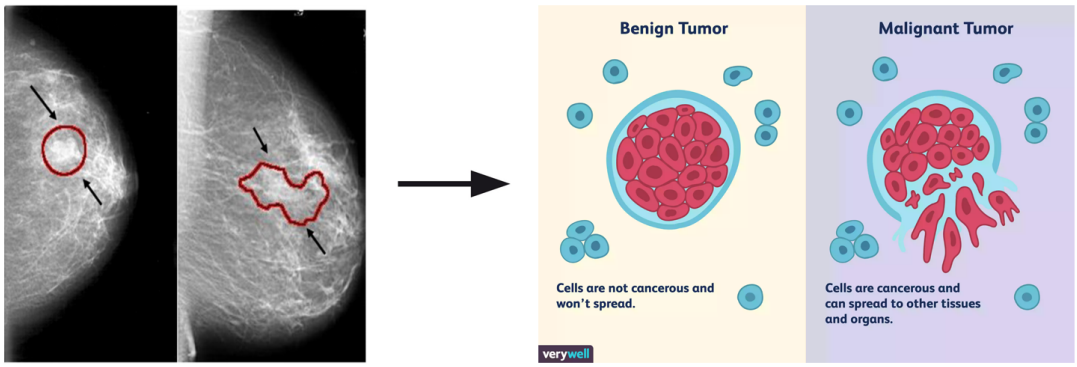


Figure 4.1: Illustration of the different types of tumor. (a) Mass, (b) MCs, (c) Benign, (d) Malignant. First, the tumor can be masses or MCs and no matter what type, they are either benign or malignant. (a) and (b) images are from [10] and (c) and (d) from Joshua Seong ©Verywell, 2017.

Therefore, the purpose of the work of Ragab et al. [10] is to detect the masses and to classify benign and malignant tissues in mammograms. The methodology to do that is presented in the next section.

4.2 Methodology for breast cancer detection

The methodology tackled in the paper [10] is divided into 5 main steps: (1) image enhancement, (2) image segmentation, (3) feature extraction, (4) feature classification, and finally, (5) an evaluation for the classifier.

The functions and the methods applied for these steps make subject of the next subsections. Basically, the objective is to detect the region of interest (ROI) of the image to extract and classify the features that would indicate if the disease is present or not.

4.2.1 Image enhancement

Taking a mammogram image in input returned by magnetic resonance imaging, the first step consists to suppress the noise to make the image more clear to highlight potential anomalies. To do that, the authors of the paper are using *contrast-limited adaptive histogram equalization* (CLAHE) which is a type of *adaptive contrast enhancement* (AHE) will be used to improve the contrast in images.

The CLAHE technique is characterized by these following steps [10]:

1. Divide the original image into contextual regions of equal size.
   → *Contextual regions* are relevant regions depending on neighbourhood pixels. For example, if we take a region of size 2x2, it is quite hard to extract information from that. The difficulty is then to find regions' sizes which may help.

2. Apply the histogram[1] equalization on each region
   → *Histogram equalization* is a method in image processing of contrast adjustment using the image's histogram characterized by the following equation [12]:

$$g_{i,j} = floor((L-1)\sum_{n=0}^{f_{i,j}} p_n)$$

   where g is the histogram equalized image, L is the number of possible intensity values, f the image and p the normalized histogram computed by $p_n = \frac{\text{pixel numbers with intensity n}}{\text{total number of pixels}}$.

3. Limit this histogram by the clip level (which corresponds to the threshold level to determine how the clipping occurs)

4. Redistribute the clipped amount among the histogram to be as close as possible to the desired histogram

5. Obtain the enhanced pixel value by the histogram integration.

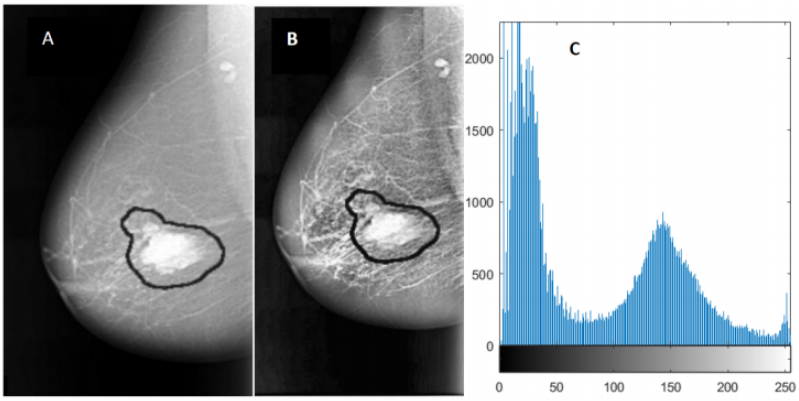


Figure 4.2: Mammogram enhancement using CLAHE [10]. (A) Original malignant mass case extracted from DDSM,(B) Enhanced image using CLAHE, and (C) Histogram representation of the image.

4.2.2 Image segmentation

After the enhancement of the image done, the next step is to segment the image into different regions which have similar features and properties. This segmentation can for example be done by using ANN, which is presented in section 3.1. The methods used in the paper are thresholding and region-based methods in addition to CNN tackled in subsection 3.1.2 to extract the region of interest (ROI). The tumors in the DDSM dataset are labelled with a red contour and accordingly, these contours are determined manually by examining the pixel values of the tumor and using them to extract the ROI [10].

4.2.2.1 Thresholding method

Thresholding is the simplest method of image segmentation and usually used to transform an image into binary images[2] (black and white images). As its name said, the thresholding method

[1] An *histogram* is an accurate representation of the distribution of numerical data. (Wikipedia)

[2] Warning: binary images are not grey-scale images. In binary images, only 2 values are possible for each pixel: black or white.

is based on a threshold value T which will determine if a pixel is 0 (white) or 1 (black) in the output image $p(x, y)$ for every pixel of the original image $q(x, y)$. Formally:

$$q(x,y) = \begin{cases} 1 & \text{if } p(x,y) > T \\ 0 & \text{else if } p(x,y) < T \end{cases}$$

In our case, thresholding method is used to determine the tumor region by a threshold value which is a value determined with respect to the red color pixel. After some trials, the threshold was set to 76 for all the images regardless of the size of the tumor [10].

4.2.2.2 Region-based method

Region-based method divides an image into different regions based on pre-defined criteria (color, intensity, etc). The main advantage of this method is that it is noise resilient. The method is defined by two types of algorithms: (1) region growing and (2) region splitting and merging. Thus, after the thresholding method which returns the biggest area within this threshold along the image, region-based method is used to crop automatically the tumor.

- **Region growing:** also classified as a pixel-based image segmentation, region growing algorithm is based on a single pixel called *seed pixel* and checks the neighboring pixels and add them to the region if they are similar to the seed. This last is repeated for each of the newly added pixels and is stopped if no more pixels can be added. Thus, these adding steps are making the region growing.
- **Region splitting and merging:** the image is successively split into quadrants based on a homogeneity criterion and regions that possess similarities regarding this criterion are merged to create the segmented result. All regions are then compared with its neighbours to compute their homogeneity and merged with them if the region pass the homogeneity test. The process is repeated until all regions pass this homogeneity test. The region splitting and merging is somehow the opposite of the region growing algorithm.

4.2.2.3 Image segmentation summary

The methodology used by Ragab et al. is then summarized by these following steps combining CNN, threshold and region-based techniques [10]:

1. [Threshold][3] Convert the original mammogram grayscale image into a binary image.
2. [CNN] Binary image objects are labelled and the number of pixels are counted.
3. [Threshold & region-based] All binary objects are removed except for the largest one, which is the tumor with respect to the threshold. The largest area is the area enclosed within the red contour labelled around the tumor.
4. [Threshold] After the algorithm checks all pixels in the binary image, the largest area pixels within the threshold are set to 1, otherwise all other pixels are set to 0
5. The resulting binary image is multiplied with the original mammogram image to get the final image without taking in account any useless information in the image.

[3]The [] indicates the technique used for this step.

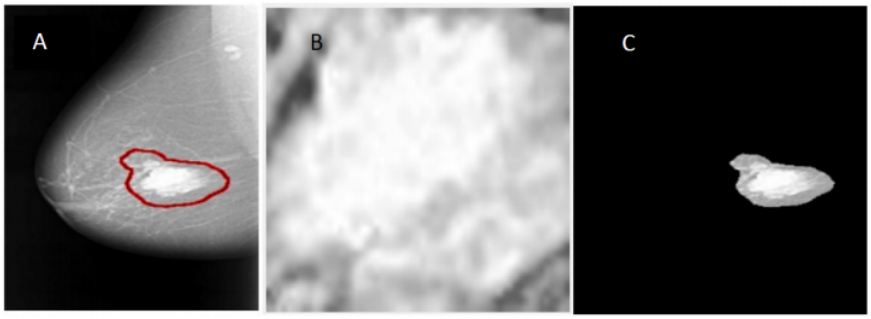


Figure 4.3: Image segmentation using above methodology. (A) Original malignant mass case extracted from DDSM, (B) enhanced ROI extracted manually using circular contours, and (C) enhanced ROI extracted automatically by the region based method [10].

4.2.3 Feature extraction

The next step after image enhancement and segmentation is to extract features from these pre-processed images. To do that, CNN is the most used common and used method of the recent years. Different types of CNN architectures exist such as CiFarNet, the Inception v1 (GoogleNet) or AlexNet. This last is the one chosen by the authors from its performance that has been proved better over the other deep learning methods for image recognition by reducing the top-5 error from 26% to 15.3%.

4.2.3.1 AlexNet architecture

AlexNet architecture is defined by 5 convolution layers, 3 pooling layers, and 2 fully connected layers with approximately 60 million free parameters. As reminder, the function of these different types of layers is explained in subsection 3.1.2. Concerning the activation function, AlexNet uses Rectified Linear Units (ReLUs) defined by $f(x) = max(0, x)$ which turns every negative values to zero.

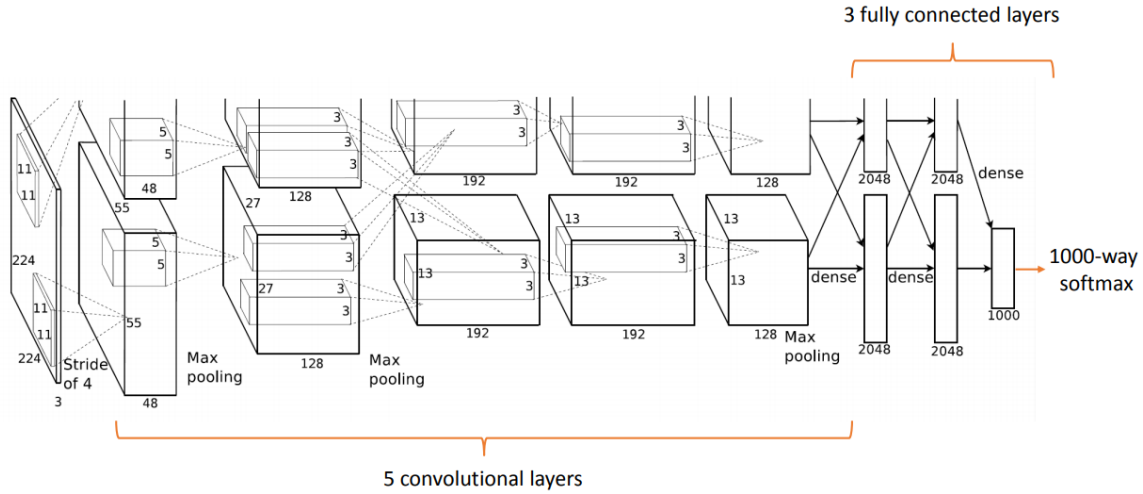


Figure 4.4: Illustration of AlexNet architecture [1].

4.2.3.2 Types of features

There exists different types of features to extract in a mammogram image. Here are a non-exhaustive list of these different features types [29]:

- **Contrast features:** this feature's type captures how well a lesion is separated from the background. An important nuisance in this setting is the tissue surrounding the lesion in addition to its border sharpness computed by the acutance feature and margin contrast.
- **Texture features:** a sign that may indicate the presence of a tumor is the non-presence of holes in the candidate lesion. Indeed, linear structures within a lesion can indicate an unfortunate projection rather then cancer [29].
- **Geometrical features:** regularity of the border of a lesion is often used to classify lesions. The authors of [29] have incorporated 5 topology descriptor that may help to detect the disease. These are eccentricity, convexity, compactness, circular variance and elliptic variance [29].

Moreover, note that there exists other feature's type that are not related to the mammogram image. For instance, location features indicate the locations that are more sensible to the occurrence of lesions such as lymph nodes or patient features such as his age.

4.2.4 Classification

The breast cancer detection has obviously two different possible outputs regarding the lesion. The ROI is classified as either **benign** or **malignant** according to the features. To classify this type of data, there exists many methods such as ANN, binary decision tree or SVM. This last has proven its efficiency for breast cancer detection by achieving high classification rates. That is the reason why the authors of the paper are using it. SVM method is fully explained in section 3.2.

To do that, after the training step, the last fully connected layer is replaced by a new layer for the classification of these two classes: benign and malignant masses.

4.2.5 Evaluation

Finally, the last step of the classification is the evaluation. We have seen in section 3.3 different tools and methods to evaluate a classifier. All these tools are appropriate to use in order to get an accurate classifier to make potential modification to enhance the model if necessary. The evaluation results of Ragab et al.'s model make subject of the next section.

A global summary of the methodology is presented in the Figure 4.5.

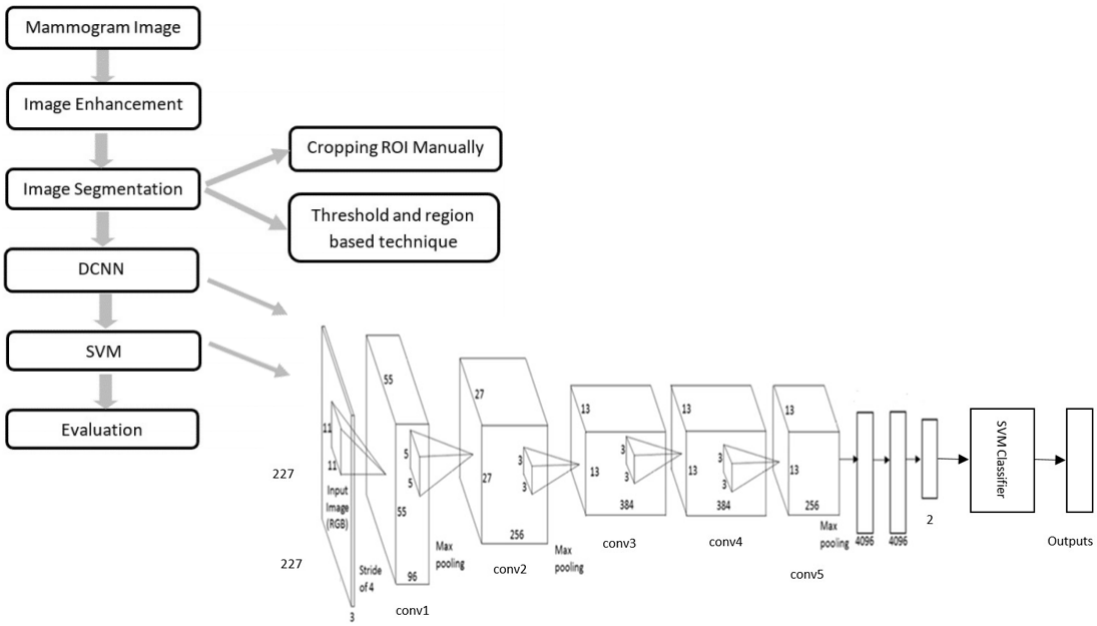


Figure 4.5: Summary of the breast cancer detection methodology of Regab et al.'s paper [10].

4.3 Current results

The following results are obtained by taking samples from the DDSM dataset which contains images that have been pre-processed and converted to 299x299 images by extracting the ROIs. The entire dataset contains 55,890 training examples, of which 14% are positive and the remaining 86% negative. 70% of the DDSM samples taken have been used for training and 30% for testing. This proportion is usually used when doing a classification problem.

The first steps are images enhancement and segmentation. For those steps, the following numbers of training and testing samples were used:

	Training	Testing	Total
DDSM (ROI cropped manually)	1580	676	2256
DDSM (ROI using threshold and region based)	1288	552	1840
CBIS-DDSM	3691	1581	5272

Table 4.1: The number of training and testing samples for all the datasets used [10].

Initially, when using manual cropping method for segmentation in addition to CNN to classify, the accuracy of the new-trained AlexNet was only 71.01%. However, when combining CNN with SVM for the classification, this accuracy increased to 79% with AUC equals to 0.88 (88%). As reminder, higher the AUC is, better the model is at predicting malignant as malignant and benign as benign.

Whereas, concerning the thresholding and region-based method for segmentation, the accuracy while using only CNN reached only 69.2%. Nevertheless, this accuracy was strongly increased and even more than the first method since it achieved 80.9% with same AUC value.

The complete evaluation of both methods using SVM with different kernel functions is presented in Figure 4.6 by computing the accuracy, AUC, sensitivity, specificity, precision and F1 score.

In conclusion, the choice concerning the kernel function is unequivocal, the best result was always obtained by using linear kernel functions. Concerning the choice of the segmentation technique, both are viable. Nevertheless, thresholding added to region-based method has slightly a better average accuracy and precision.

SVM kernel functions	Cropping ROI manually					
	Accuracy	AUC	Sensitivity	Specificity	Precision	F1 score
Linear	79%	0.88	0.763	0.822	0.85	0.8
Quadratic	77.9%	0.87	0.764	0.797	0.81	0.786
Cubic	77.2%	0.86	0.759	0.781	0.79	0.774
Fine Gaussian	64.2%	0.74	0.741	0.598	0.43	0.544
Medium Gaussian	77%	0.87	0.754	0.787	0.8	0.776
Coarse Gaussian	73.7%	0.83	0.696	0.807	0.83	0.765

SVM Kernel functions	Threshold + region based segmentation technique					
	Accuracy	AUC	Sensitivity	Specificity	Precision	F1 score
Linear	80.5%	0.88	0.774	0.842	0.86	0.815
Quadratic	80.1%	0.87	0.772	0.833	0.85	0.809
Cubic	78.3%	0.85	0.764	0.797	0.81	0.786
Fine Gaussian	54%	0.7	0.51	0.833	0.99	0.673
Medium Gaussian	79.1%	0.86	0.756	0.820	0.84	0.796
Coarse Gaussian	77.2%	0.85	0.736	0.813	0.84	0.785

	Segmentation techniques	
	Cropping ROI manually	Threshold + Region based
Trained DCNN accuracy	71.01%	69.2%
Error in testing	30.17%	30.43%
SVM accuracy	79%	80.5%
Sensitivity	0.763	0.774
Specificity	0.822	0.842
AUC	0.88	0.88
Precision	0.84	0.86
F1 score	0.8	0.815

Figure 4.6: Complete evaluation of the different methods used. The first results table is obtained by using manual cropping for segmentation while the second one is using thresholding and region-based methods. Moreover, these two first tables are comparing results with different kernel functions for SVM. The last table is summarizing the different evaluation values between these two segmentation methods. A red value indicates the greatest value between the compared methods [10].

Chapter 5

Model prototype

After discussing theoretically about the different methods and materials of image classification and recognition, this chapter is tackling a simple image classification model implementation for the non-medical dataset CIFAR-10 [2] based on the CNN implementation of Jason Brownlee [5] and Dan Nelson [21]. Henceforth we can determine the parameters which optimize the evaluation criteria of a predictive model and how it reacts to noise.

5.1 The CIFAR10 dataset

The CIFAR (Canadian Institute For Advanced Research) created two versions of datasets: CIFAR-10 and CIFAR-100 which differs from the number of image classes that contains the respective datasets. In this work, we will use the CIFAR10 dataset whose involves a collection of images belonging to the following 10 classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks illustrated in Figure 2.1.

The CIFAR-10 dataset consists of 60000 32×32 colour images in these 10 classes, with 6000 images per class mutually exclusive. 50000 of these images are dedicated to the training phase and constitutes the training set while the 10000 remaining are dedicated to the evaluation and then shapes the test set (test batch). These 50000 training images are divided into batches of 10000 elements. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

The fact that the images are in low resolution allows mainly fast training phase execution which makes the testing easier. Moreover, CIFAR-10 is a well-known dataset and has proven several times that it was a friendly and efficient dataset to use in order to test different models on it as well as MNIST dataset which approaches a trivial problem (recognizing 10 digits).

5.2 Model implementation

In this section, the development of a convolutional neural network using Python3 and its machine learning packages such as TensorFlow and Keras is done. Moreover, parameters used are reviewed in order to optimize the model's accuracy.

5.2.1 Why CNN over other models?

As seen in subsection 3.1.2, the best model type for images are convolutional neural networks. Some reasons of this efficiency may be that CNN is a kind of deep learning neural network where **transfer** learning happens which means that learned knowledge about the data are stored and used for the future. An example of transfert learning related to CIFAR-10 would be that the knowledge acquired to recognize cars could be applied when trying to recognize trucks. A simple analogy of transfert learning could be the learning of a Romance (e.g. Spanish) language while your native language is another Romance language (e.g. Italian) facilitates and accelerates the learning. This works the same for machine learning and image recognition.

Moreover, thanks to its pooling layers (see subsubsection 3.1.2.2), CNN manages dimensionality reduction which enhances the feature extraction. Thus, CNN can somehow be thought of automatic feature extractors from the image. In comparison with other methods such as KNN, SVM or logistic regression where we need to spend much time on feature selection, this time is then less significant for CNN.

5.2.2 Structure and implementation

5.2.2.1 Dataset preprocessing

Sampling

Before starting to create the model, the preprocessing step on dataset is essential in order to get more significant prediction results. Initially, the training set of CIFAR-10 is composed of 50000 images. However, it is necessary to **sample** this starting training set for two reasons:

- To avoid multiple large execution times: multiple tests are done, it would be really long and constraint to train on the whole training set at each execution
- To produce highly imbalanced set in order to test the methods on it and obtain significant results

Hence, the sampling phase is splitting up the 50000 images into a drastically smaller number. Knowing that the CIFAR-10 is divided into 5 batches of 10000 images, the sample method of our model is simply to use only one batch of the dataset.

One hot encoding

The second preprocessing to apply on the dataset is a process called **One hot encoding**. What is this doing is converting the categorical data label encoding (i.e. airplanes=0, cars=1, birds=2, ...) into another type of encoding that enhances the reading of these data by the machine learning algorithms. But why this label encoding would not be enough? The problem with label encoding is that the categories now have natural ordered relationships. The model will then assumes that the higher numbers has higher weights which may result in poor performance or unexpected results. Therefore, rather than labelling things as a number starting from 0 and then increasing for each category, we will go for more of a binary style of categorizing.

Thus, one hot encoding transforms our categorical labels into vectors of zeros and ones. The length of this vectors is equal to the number of classes that our model is expected to classify. The one hot encoding of CIFAR-10 is illustrated in Figure 5.1. For example, the dog class is then represented by the vector $[0, 0, 0, 0, 0, 1, 0, 0, 0, 0]$.

index	label
0	**airplane** (0)
1	**automobile** (1)
2	**bird** (2)
3	**cat** (3)
4	**deer** (4)
5	**dog** (5)
6	**frog** (6)
7	**horse** (7)
8	**ship** (8)
9	**truck** (9)
...	...
...	...

original label data

label	index											
	0	1	2	3	4	5	6	7	8	9	...	...
airplane	**1**	0	0	0	0	0	0	0	0	0	...	...
automobile	0	**1**	0	0	0	0	0	0	0	0	...	...
bird	0	0	**1**	0	0	0	0	0	0	0	...	...
cat	0	0	0	**1**	0	0	0	0	0	0	...	...
deer	0	0	0	0	**1**	0	0	0	0	0	...	...
dog	0	0	0	0	0	**1**	0	0	0	0	...	...
frog	0	0	0	0	0	0	**1**	0	0	0	...	...
horse	0	0	0	0	0	0	0	**1**	0	0	...	...
ship	0	0	0	0	0	0	0	0	**1**	0	...	...
truck	0	0	0	0	0	0	0	0	0	**1**	...	...

one-hot-encoded label data

Figure 5.1: One hot encoding of CIFAR-10 dataset. To the left, the categorical data label encoding. To the right, the one hot encoding.

5.2.2.2 Creating the CNN

The CNN is created using keras library from Python. Its architecture is based on the classic CNN architecture illustrated on Figure 3.3 taking on input an image passing through convolutional layers and a fully connected layer giving the output class. Below is described the basic structure of our implemented CNN (baseline).

Model basis

When using keras, it allows two ways to generate a model: sequential or functional. The major difference between them is that the functional model allows you to connect layers to more than just the previous and next layers which allows you to create much more complex models and networks. In our case, it is not necessary. A sequential model is widely sufficient. Indeed, its limit is defined by its incapacity to share layers or have multiple inputs or outputs which is not necessary to resolve a simple image classification problem. The model is then created using the following method:

```
Sequential()
```

Convolutional layers

Once the basis of the model has been created, we can generate the first convolutional layers. As we manage images, we have to use 2D convolutional layer which mainly take the number of filters for the convolution, the filter's size and the input shape as parameters. The keras method to create a 2D convolutional layer is then basically defined by:

```
Conv2D(filters, filter_size, input_shape)
```

In our case, we use 32 as starting filters number which doubles each time we add a convolutional layer. The size of each filter is a 3x3 window. We also add `padding="same"` (which defines that we do not change the input image size) as well as its activation function as parameters. We

generally use `ReLU` as activation function which is the most widely used activation function in neural networks today such as the AlexNet architecture. The exhaustive list of `Conv2D` method parameters can be found on the official keras library documentation [8].

Between each layer, we use batch normalization which normalizes the inputs heading into the next layer using, ensuring that the network always creates activations with the same distribution that we desire [21]:

```
BatchNormalization()
```

After creating two convolutional layers, we generate a pooling layer of size 2x2 which compresses the information of the image to make it smaller (see subsubsection 3.1.2.2 for full explanation). The pooling layer is created using:

```
MaxPooling2D(size)
```

It's important to not have too many pooling layers, as each pooling discards some data. Pooling too often will lead to a too small number of information that can be used for the next layers (densely connected layers) to learn about the data. Generally, we add a pooling layer between 2 convolutional layers to reduce the spatial volume of input image after convolution.

Should we add another convolutional layer? There is no exact number of convolutional layers but each one added increases the number of filters so the model can learn more complex representations but obviously also adds more computation expenses.

As we manage small sized images (32x32), the model that we are creating will be quite small. We then only add a third convolutional layer. Thus, we only pool once as our model contains only 3 convolutional layers.

Fully-connected layer

Before creating the fully-connected layer, we need to flatten the data to obtain the data into a 1-dimensional array (containing all the pixels data) for inputting it to the next layer so that he can process pixel by pixel to finally have the final output in the good format (a single vector of 10 elements representing the likelihood of each 10 outputs). We then add a flatten layer by using:

```
Flatten()
```

We can now create the fully-connected layer (dense layers) where the neurons have a complete connection to all the activations from the previous layers to generate a prediction from the results of these previous layers. A dense layer basically takes the number of neurons as parameter and then can be used in keras using (all parameters available in keras documentation [8]:

```
Dense(neuronsNumber)
```

The number of neurons chosen are starting from 256 which decreases in succeeding layers to finally approach the same number of neurons as there are classes in the dataset. Thus, we pass to the final layer the number of classes for the number of neurons (=10). Each neuron then represents a class with a certain probability that the image is belonging to this class. Moreover, as we want to prevent from overfitting, we also add a kernel constraint as parameter which regularize the data as it learns.

Finally, as we have seen in subsubsection 3.1.2.3, we use *Softmax* as activation function which selects the neuron with the highest probability as its output, voting that the image belongs to that class.

The Figure 5.2 is summarizing the baseline of the CNN implemented.

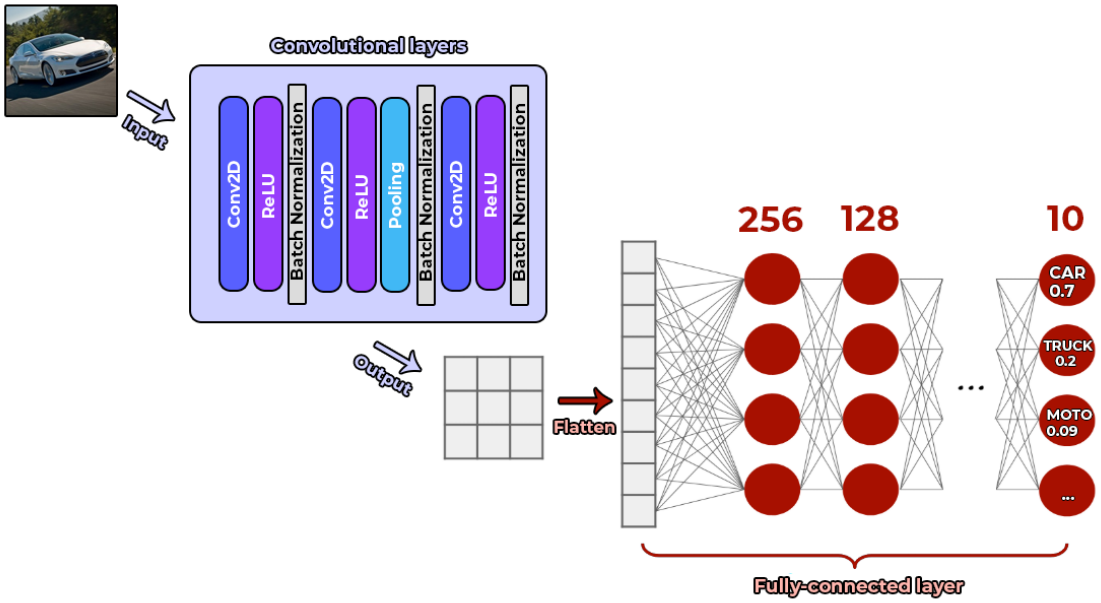


Figure 5.2: The baseline of the CNN implemented. In this example, the input image is a car and the CNN model outputs "car" as highest probability (0.7) and is then assigning the input image as "car" class.

5.2.3 Summary pseudo-code

Algorithm 2 Baseline CNN implementation

```
Init the model: Sequential()
for a number of convolutional layer to add do
    Add convolutional layer
    ReLU activation function
    if between 2 convolutional layer then
        Add pooling layer
    Batch Normalization
Flatten the data
for a number of Dense layer to form the fully-connected layer do
    Add dense layer
    if not last layer then
        ReLU activation function
        Batch Normalization
    else
        Softmax activation function
Compile the model
Train the model: model.fit(X_train, y_train, X_test, y_test)
Predict: model.predict()
```

5.3 Optimizing parameters (Regularization)

The CNN model created above is the simplest way to create a CNN and would probably not have the best prediction results. In this section, the different parameters that may enhance the accuracy and the results of our predictive CNN model are theoretically described in order to understand how to get better results. Generally, these parameters add information in order to prevent overfitting of which the process is called **regularization**.

5.3.1 Epochs

When compiling your model, you give a number of epochs and a batch size to your model. Batch size may not influence the accuracy result but only the memory space needed. Higher the batch size, the more memory space you'll need. We usually use 64. In contrast, epochs certainly influences the performance of the model since an epoch refers to one cycle through the full training dataset. Thus, more epochs the model is learning, more features it will learn which can be damaging the model's performance due to overfitting, i.e. your model does not learn the data, it memorizes the data. Finding the good epochs number is then meaningful for the model's performance.

5.3.2 Optimizer

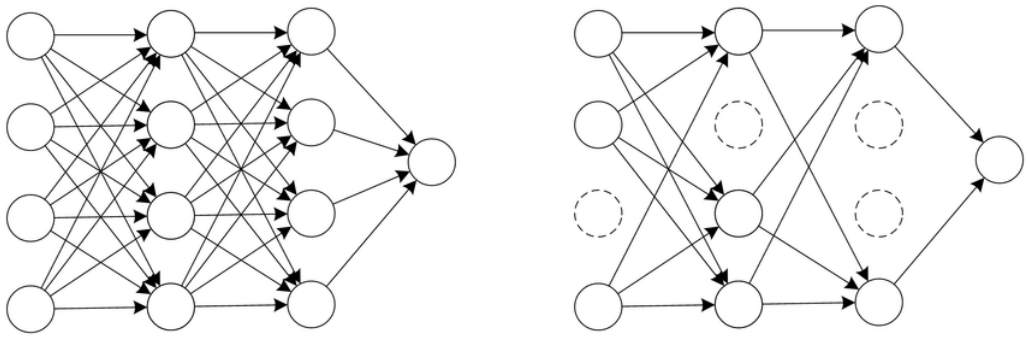
Also at compilation step, you need to specify an optimizer. Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the losses.

Optimizers algorithms are generally based on *gradient descent algorithm* which is optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient [24].

The choice of optimizers are based on the type of network, type of layers and type of data. In the subsection 5.4.1, we will experiment 5 different optimizers: RMSProp, ADAM, SGD, AdaGrad and AdaDelta.

5.3.3 Dropout

Dropout refers to ignoring units (i.e. neurons) during the training phase of certain set of neurons which is chosen at random. For instance, a dropout value of 0.2 means that you randomly select 20% of the neurons and set their weights to zero for the forward and backward passes i.e. for one iteration. This dropout process is used to prevent from overfitting that can be led by neurons developing co-dependency amongst each other during training phase.



(a) Standard Neural Network

(b) Network after Dropout

Figure 5.3: Dropout Neural Net Model. Left: A standard neural net with 2 hidden layers. Right: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped. [26]

5.3.4 Weight decay

Another way to prevent from overfitting is to use weight decay which involves updating the loss function to penalize the model in proportion to the size of the model weights. By updating weights this way, we penalize the learning complexity and then reduces the chance of overfitting. Weight decay is a regularization method existing by 3 types of regularizers, with x a small number:

- $L1(x)$: we multiply the sum of the absolute weights with x.
- $L2(x)$: we multiply the sum of the squared weights with x.
- $L1L2(x)$: we multiply the sum of the absolute and squared weights with x.

Generally, we use L2 weight regularization which is the most common type used for neural networks since L2 regularizer does not have sparse estimations and is computational efficient due to having analytical solutions.

5.3.5 Data augmentation

Same as previous parameters, our objective is to reduce overfitting to generalizes well the data. Another way to allow the model to learn the same general features is making copies of the examples in the training dataset and add small random modifications to the images. It expands the dataset so that model learns to not take in account these modifications but only the general features. This process is called data augmentation. Many types of data application can be applied to images such as cropping, rotating, shifting, zooming, random erasing, etc.

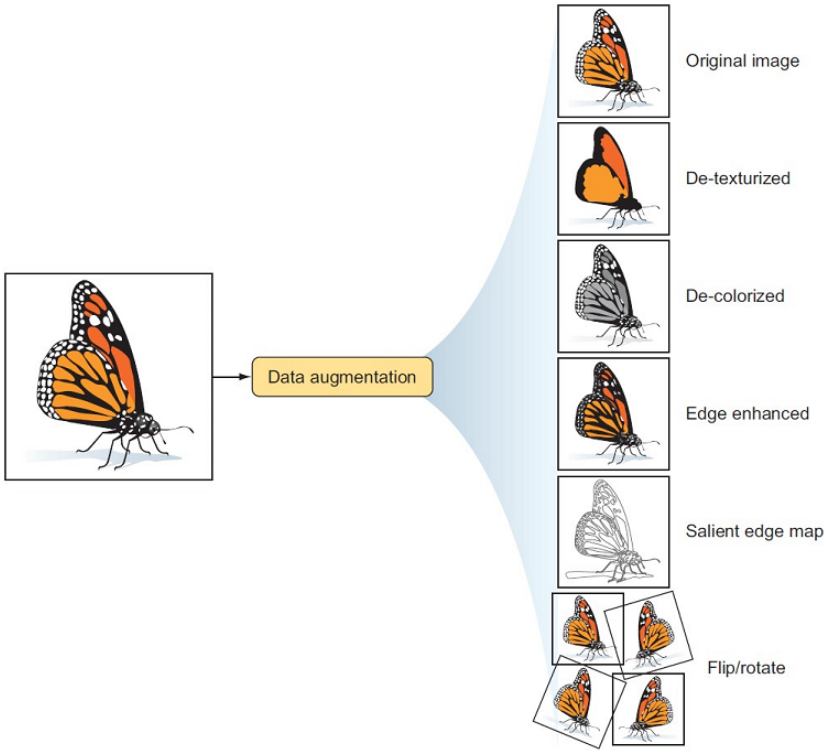


Figure 5.4: Some image data augmentation examples.

5.3.6 Adding noise

What if the images are getting noisy over time? Is the model reacting well to that kind of perturbations? Theoretically, if the model suffers from overfitting, adding noise may be benefit to the accuracy of the model for the reason that learning noisy images is more complex and the model may only distinguish general features and big lines of the images. Thus, adding noise during the training phase should have a regularization effect and can make the training process more robust and reduce generalization error.

The most common type of noise is Gaussian noise illustrated in Figure 5.5. Of course, adding noise is done relating to a configurable hyperparameter. The choice of this value is then decisive since too little noise has no effect, whereas too much noise makes the mapping function too challenging to learn.

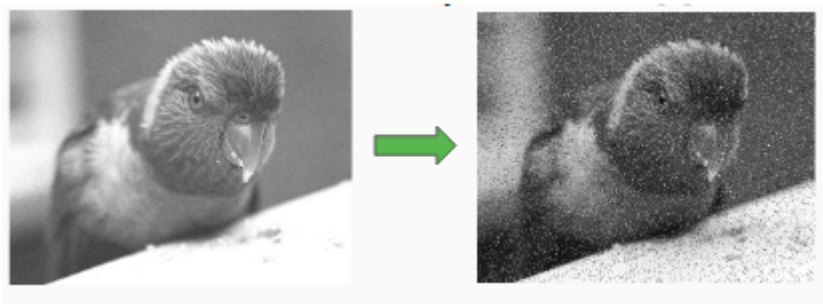


Figure 5.5: Illustration of adding Gaussian noise to an image. The image becomes blurry.

5.4 Experiments and results

Several experiments testing the efficiency of the different parameters are done and the corresponding results are presented to find optimal parameters.

5.4.1 Experiments

Here under are the experiments that have been done in order to encompass all the parameters tests to show how the CNN model react to them and how they impact its performance. So that, each parameter are approached **separately** (= without any other regularizer than the parameter in question). These experiments are done with sampling on 1 batch of the training set (10000 images) on 5 epochs. Note that the sampling is **not stratified** which means that the batch selected does not respect the proportions of each class from the complete dataset. Moreover, we also have done the experiments concerning regularizers (experiment 3 to 6) on the complete training set also on 5 epochs. Indeed, regularizers are processes used to counter overfitting. Therefore, using the whole training set may be more reliable to observe their efficiency. For both cases, the testing phase has been done on the test set (10000 images). They all have been done using AdaDelta as optimizer (except experiment 2 obviously).

- *Experiment 1 - Epochs number:* run on 40 epochs, on the baseline CNN without any regularizer (= no dropout, no weight decay, no data augmentation and no added noise)
- *Experiment 2 - Optimizer:* 5 runs testing the following optimizers: RMSProp, ADAM, SGD, AdaGrad and AdaDelta.
- *Experiment 3 - Dropout value:* 6 runs with the following dropout values: $[0, 0.1, 0.2, 0.4]$, the last value means that the dropout value is increasing from 0.2 to 0.5 over layers.
- *Experiment 4 - Weight decay values:* 2 runs with the following weight decay values: $[0.001, 0.01]$.
- *Experiment 5 - Data augmentation:* unique run with data augmentation on images : horizontal flip, minor shifts of the image and small zooming.
- *Experiment 6 - Adding Gaussian noise:* 3 runs with the following Gaussian noise rate: $[0.1, 0.2, 0.4]$.

5.4.2 Results

N.B.: As we use sampling to avoid multiple large execution times, the results may be quite lower compared to if we have used the whole dataset. Nevertheless, these results can be more significant since the training set is highly imbalanced.

5.4.2.1 Experiment 1 - Epochs number

What we can see from the results presented in Figure 5.6 is that the accuracy reaches $\approx 100\%$ for the training set after only 5-6 epochs which shows that using more epochs should not be relevant and even counterproductive. Indeed, we can see that the global accuracy is not evolving and stalling around 65% between 10 and 40 epochs which shows the uselessness of using too much epochs.

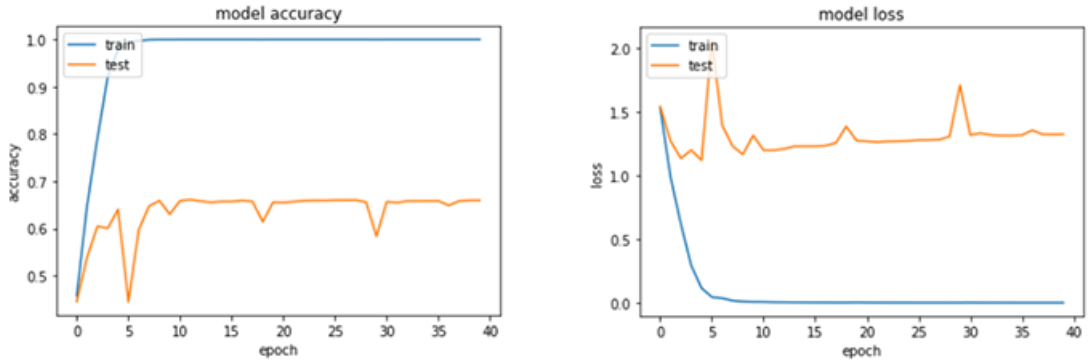


Figure 5.6: Experiment 1 result: model accuracy and lost over epochs.

5.4.2.2 Experiment 2 - Optimizer

This experiment represents the results without any regularizer depending only on optimizer choice.

The general behaviour of the results observed about the optimizers is that SGD (Stochastic gradient descent) seems to not be a good optimizer in this case. Indeed, SGD is very basic and rarely used now: although SGD minimizes loss faster, it is noisier and it oscillates around the the minimum giving some variation in accuracy and loss run to run.

The other optimizers as for them have quite similar results even if ADAM and AdaGrad seems to be the most adapted in our case. In fact, their algorithm is quite similar too: RMSProp and AdaDelta have both been developed independently but AdaDelta is an extension of AdaGrad and ADAM is somehow a combination of AdaDelta and RMSProp. However, we generally use AdaGrad to deal with sparse data and ADAM for more general cases since it requires relatively low memory and combines the good properties of AdaDelta and RMSProp. That may explain why ADAM and AdaGrad stand out from the crowd.

Optimizer	Accuracy (with sampling)
RMSProp	54.640%
ADAM	61.220%
SGD	51.720%
AdaGrad	**63.490%**
AdaDelta	55.770%

Table 5.1: Experiment 2 results.

5.4.2.3 Experiment 3 - Dropout value

For the sampling case, we see that the best result is obtained while not using dropout. It is somehow logical since sampling removes already lot of information and is hence less subject to overfitting but more to underfitting.

However, when using the entire training set, it is more to be prone to overfitting. That is why we obtain better results when using dropout. Watch out, we have to take care of the value of dropout added. Indeed, the results show that see that a too high dropout value is decreasing drastically

the accuracy which means that too much dropout was done impacting the capacity of the model to learn well. On the following results, we see that a dropout value above 0.2 is decreasing the accuracy which can be considered as the threshold value which impact negatively the model's learning. Using keras, dropout can be added using `model.add(Dropout(dropoutValue))`.

Dropout value	Accuracy (no sampling)	Accuracy (with sampling)
0	76.13%	**63.490%**
0.1	76.51%	55.530%
0.2	**78.140%**	57.840%
0.4	59.890%	45.140%

Table 5.2: Experiment 3 results table.

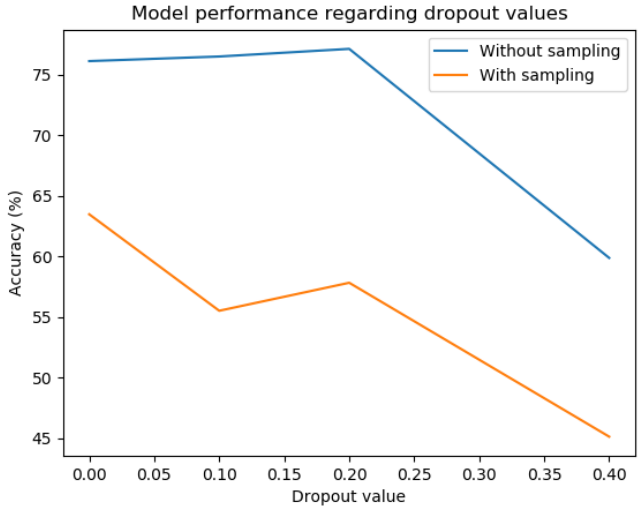


Figure 5.7: Experiment 3 result graph: the blue line is the experiment without sampling and the orange line is the one with sampling. $0.2 \rightarrow 0.5$ refers to the experiment using increasing dropout values.

5.4.2.4 Experiment 4 - Weight decay values

The impact of weight decay looks a bit less effective as dropout in most and general cases. But in the sampling case, we now observe a small accuracy enhancement which was not observable when using dropout. However, using larger weighting (0.01) looks to not be a good idea neither since weight decay is usually used with a very small weighting number. Using keras, weight decay is done by adding `kernel_regularizer=l2(0.01)` to `Conv2D` parameters.

Weight decay value	Accuracy (no sampling)	Accuracy (with sampling)
0	76.13%	63.490%
0.001	**77.470%**	**64.190%**
0.01	76.090%	60.140%

Table 5.3: Experiment 4 results table.

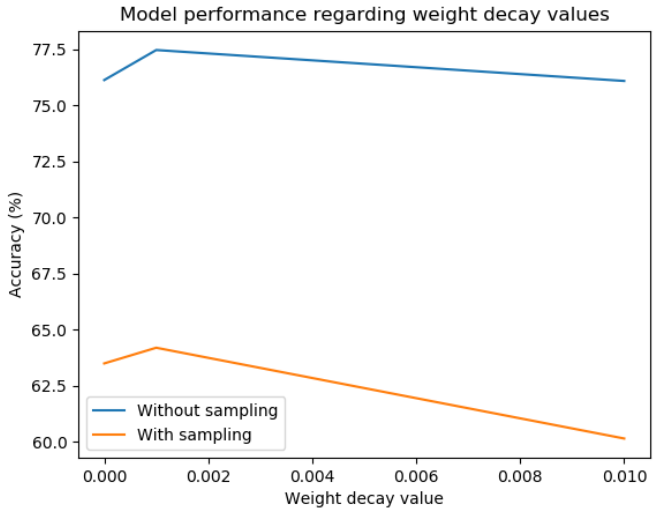


Figure 5.8: Experiment 4 result: the blue line is the experiment without sampling and the orange line is the one with sampling.

5.4.2.5 Experiment 5 - Data augmentation

In both cases, we see an improvement compared to the baseline results. Data augmentation seems then to be a good process to enhance the model's accuracy. Making copies of the images with random modifications expands the training set passed to the model and then can only be benefit the model's learning. To implement data augmentation, we have used `ImageDataGenerator` method also from keras library.

Data augmentation	Accuracy (no sampling)	Accuracy (with sampling)
No	76.13%	63.490%
Yes	**78.410%**	**64.810%**

Table 5.4: Experiment 5 results table.

5.4.2.6 Experiment 6 - Adding Gaussian noise

Same as previous regularizers, choosing the right added noise rate is really important. As the training set is highly balanced, this can be well observed in the non-sampling case where we observe that the accuracy is decreasing a lot when adding too much noise. This can easily be explained same as experiment 3 with dropout, i.e. in the sampling case, we learn less then we already have missing information (underfitting), adding noise is then not benefit to increase the model accuracy.

However, for the non-sampling case that the model is reacting well to noise since that the accuracy is not fluctuating at all and keep constant. This Gaussian noise has been added after the pooling layer by using `GaussianNoise(value)` from keras.

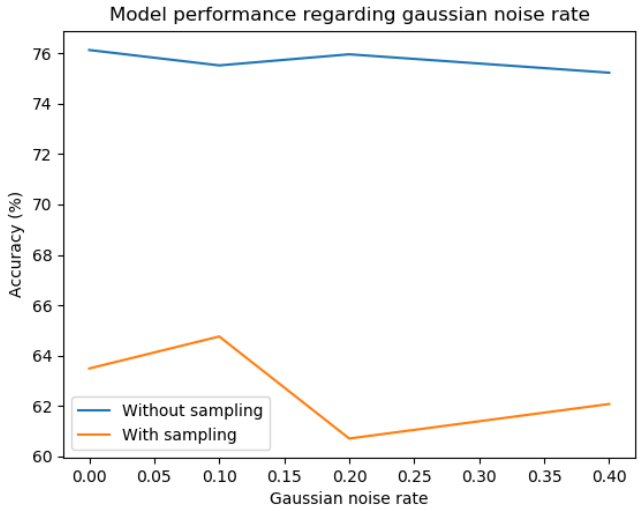


Figure 5.9: Experiment 6 result: the blue line is the experiment without sampling and the orange line is the one with sampling.

5.4.3 Conclusion

When talking about image classification, the most popular model is a convolutional neural network. After creating a baseline CNN, several regularizers have been introduced and have been experimented to observe their impact on our model's accuracy. Following the results obtained, dropout regularizer and data augmentation seems to be the most efficient ones. After discussing about a simple non-medical case, we can henceforth tackle medical images using our knowledge about CNNs.

Chapter 6

Identification of Leukemia Subtypes from microscopic images

Bibliography

[1] Alex Krizhevsky, Ilya Sutskever, G. H. (2012). Imagenet classification with deep convolutional neural networks. *University of Toronto*.

[2] Alex Krizhevsky, V. N. et Hinton, G. (2009). The cifar-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html.

[3] Arm, P. (2017). Image recognition: A short history and all you need to know about it. https://www.projectarm.com/image-recognition-what-it-is/.

[4] Bhande, A. (2018). What is underfitting and overfitting in machine learning and how to deal with it. https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning.

[5] Brownlee, J. (2019). How to develop a cnn from scratch for cifar-10 photo classification. https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/.

[6] Carson Lam, Darvin Yi, M. G. T. L. (2018). Automated detection of diabetic retinopathy using deep learning. *AMIA Jt Summits Transl Sci Proc*.

[7] Castle, N. (2018). Regression vs. classification algorithms. https://www.datascience.com/blog/regression-and-classification-machine-learning-algorithms.

[8] Chollet, F. (2015). Keras documentation. https://keras.io/.

[9] Cole, M. (2017). Building a facial recognition pipeline with deep learning in tensorflow. https://hackernoon.com/.

[10] Dina A. Ragab, Maha Sharkas, S. M. et Ren, J. (2019). Breast cancer detection using deep convolutional neural networks and support vector machines. *PeerJ*.

[11] Geert Litjens, Thijs Kooi, B. E. B. A. A. A. S. F. C. M. G. J. A. v. d. L. B. v. G. C. I. S. (2017). A survey on deep learning in medical image analysis. *Radboud University Medical Center*.

[12] Gonzalez, R. C. et Woods, R. E. (2008). *Digital Image Processing, 3rd edition*. Prentice Hall.

[13] Google (2019). Ml practicum: Image classification. https://developers.google.com/machine-learning/practica/image-classification/.

[14] Guy Shtar, S. M. (2017). Clustering and dimensionality reduction: Understanding the "magic" behind machine learning. https://www.imperva.com/blog/.

[15] H. Lee, Y. Largman, P. P. et Ng, A. Y. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. *NIPL*.

[16] Hanieh, B. (2014). Artificial neural network part 1.

[17] Hargrave, M. (2019). Deep learning. https://www.investopedia.com/terms/d/deep-learning.asp.

[18] Institute, N. C. (2019). Nci dictionary of cancer terms. https://www.cancer.gov/publications/dictionaries/cancer-terms/.

[19] John Paul Mueller, L. M. (2016). *Machine Learning for Dummies*. Learning Made Easy.

[20] MedCalc (2019). Roc curve analysis. https://www.medcalc.org/manual/roc-curves.php.

[21] Nelson, D. (2019). Image recognition in python with tensorflow and keras. https://stackabuse.com/image-recognition-in-python-with-tensorflow-and-keras/.

[22] Rouse, M. (2016a). Supervised learning. https://searchenterpriseai.techtarget.com/definition/supervised-learning.

[23] Rouse, M. (2016b). Unsupervised learning. https://whatis.techtarget.com/definition/unsupervised-learning.

[24] Ruder, S. (2016). An overview of gradient descent optimization algorithms. https://ruder.io/optimizing-gradient-descent/.

[25] Seif, G. (2018). Deep learning for image recognition: why it's challenging, where we've been, and what's next. https://towardsdatascience.com/.

[26] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I. et Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

[27] Suvajit Dutta, Bonthala CS Manideep, S. M. B. R. D. C. et Iyengar, N. C. S. N. (2018). Classification of diabetic retinopathy images by using deep learning models. *International Journal of Grid and Distributed Computing*.

[28] S.V.S. Prasad, Satya Savithri, I. V. M. K. (2015). Techniques in image classification; a survey. *Global Journals Inc.*

[29] Thijs Kooi, Geert Litjens, B. v. G. A. G.-M. C. I. S. R. M. A. d. H. N. K. (2016). Large scale deep learning for computer aided detection of mammographic lesions. *Elsevier B.V.*

[30] Varone, M. (2019). What is machine learning? a definition. https://www.expertsystem.com/machine-learning-definition/.

[31] Waseem Rawat, Z. W. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *University of South Africa*.

[32] Wikipedia (2019a). Convolutional neural networks. https://en.wikipedia.org/wiki/Convolutional_neural_network.

[33] Wikipedia (2019b). Cross-validation (statistics). https://en.wikipedia.org/wiki/Cross-validation_(statistics).

[34] Wikipedia (2019c). Support vector machines. https://en.wikipedia.org/wiki/Support-vector_machine.

[35] Yadav, A. (2018). Support vector machines (svm). https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589.

[36] Zahra Sobhaninia, Safiyeh Rezaei, A. N. M. A. H. Z. N. K. A. E. S. S. (2018). Brain tumor segmentation using deep learning by type specific sorting of images. *Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan*.

[37] Zhou, L. (2018). Simplify machine learning pipeline analysis with object storage. https://blog.westerndigital.com/machine-learning-pipeline-object-storage/.