UNIVERSITÉ LIBRE DE BRUXELLES

COMPUTER SCIENCES DEPARTMENT, FACULTY OF SCIENCES

COMPUTER SCIENCES MASTER THESIS

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

Quang Linh **BUI QUANG PHUONG**

ULB ID: 000427796

Supervisor: Matthieu Defrance

Academic Year 2020-2021

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 [15] to detect breast cancer by analysing mammogram images with an average accuracy of 80% makes subject of an entire chapter of the document.

In a second phase, some experiments were done by implementing a CNN from scratch on the well-known dataset CIFAR-10 containing non-medical images. This is realized to manipulate parameters in order to perceive variations in the model accuracy and results. The parameters approached are the epochs number, optimizer used, dropout value, weight decay, presence of noise and the use of data augmentation such as flips or enhanced edges. What we have observed from the result is that the use of dropout and data augmentation seems to be the most efficient parameters with the accuracy approaching respectively 78.140% and 78.410%. Then, the CNN was used for the purpose of a medical problem: the identification of leukemia subtypes from microscopic images. The experiments were realised using ALL-IDB and DEMIR-LEUKEMIA datasets. The process is based on the same parameters and the same CNN architecture than the experiments on CIFAR-10 dataset. The results obtained show that data augmentation and images enhancements techniques are overall the most valuable parameters to optimise the CNN model. Using either one of these parameters, the accuracy can go up to 91% for ALL-IDB1, 80.38% for ALL-IDB2 and 71.91% using ALL-IDB and DEMIR-LEUKEMIA datasets. In addition to this, we have also demonstrated by these experiments the importance of having a compliant dataset.

Afterwards, we discuss about the concept of interactive image labelling and active learning which are techniques that can bring forward the use of AI more than a computer-aided detection tool. These techniques are based on the presence of an oracle that helps the model to collect and annotate new informations. In this chapter, the methodology of active learning such as the query strategies are tackled as well as the performance and limitations of such techniques. Finally, the discussion continues by introducing the approach of a neural network extension which consists of mixing data types in addition of images.

Contents

[1] Corresponds to the summary of chapter 1 (Introduction) to chapter 4 (Breast cancer detection) included.
[2] Corresponds to the summary of chapter 5 (Model prototype) and chapter 6 (Leukemia Subtypes identification).
[3] Corresponds to the summary of chapter 7 (Active Learning).

Chapter 1

Introduction

The thesis is structured into 3 distinct sections. The first one is the **state-of-the-art** containing the context of the classification problem (chapter 1), the general concepts of AI (chapter 2), the existing material and methods used to solve this problem (chapter 3) and a review of a well-known application of image classification for biomedical data (chapter 4). The second one is dedicated to my **own experiments** creating and using a classification model, first on classical images dataset (chapter 5) and then on biomedical data (chapter 6). Finally, the last section of the thesis is the **discussion** part where we are approaching further methods that can enhance the existing classification methods (chapter 7 and chapter 8).

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.

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 [6].

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 [19]. 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. Figure extracted from [19]

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.

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

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

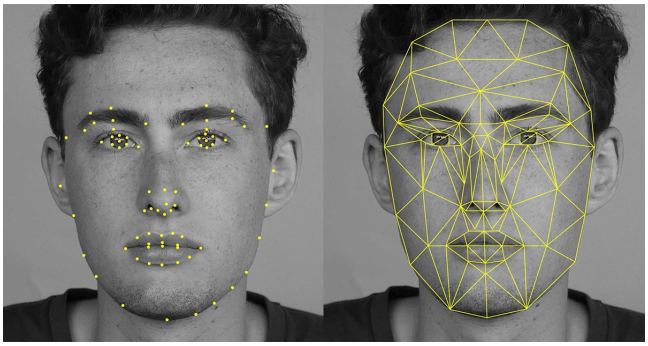


Figure 1.2: Keypoints marking to extract features for face recognition. Figure extracted from [14]

in the large library.

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.

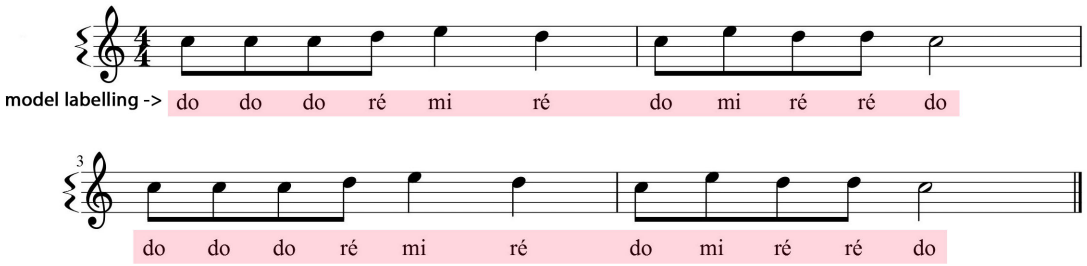


Figure 1.3: (French) Labelling of the different notes of "Au clair de la lune" 's music score

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.

1.3 The thesis' purpose application: 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

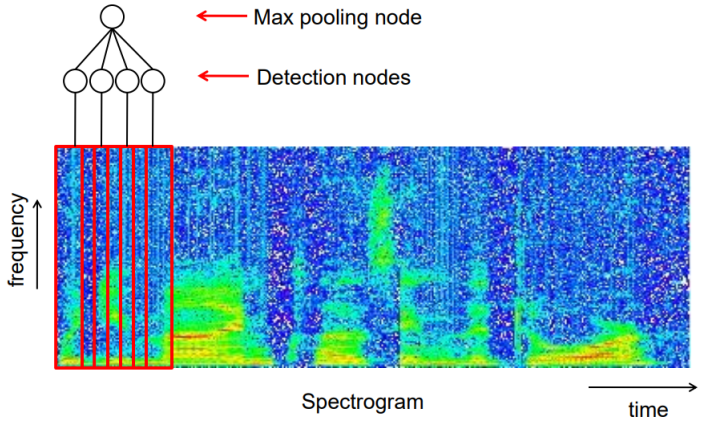


Figure 1.4: A sound spectrogram recognition using convolutional DBN. Figure extracted from [21].

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] [47]. 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).

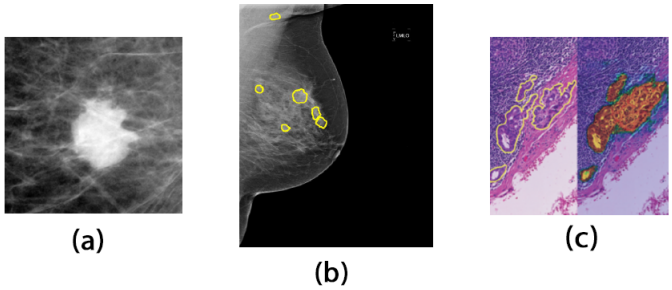
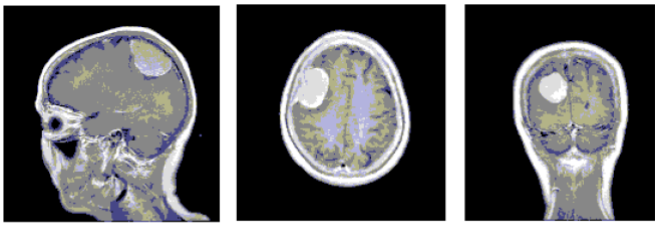


Figure 1.5: Mammographic mass images classification. (a, b) Segmentation of lesion, (a) is the illustration of the lesion on a smaller scale. (c) breast cancer metastases detection in lymph nodes using an histopathology image. Figures extracted from [47]

[2]From the *National Cancer Institute* (NCI) [24]: 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.

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 [54].



Sagittal view Axial view Coronal view

Figure 1.6: Brain MRI slices captured from different directions. Figure extracted from [54].

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 [45].

[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.

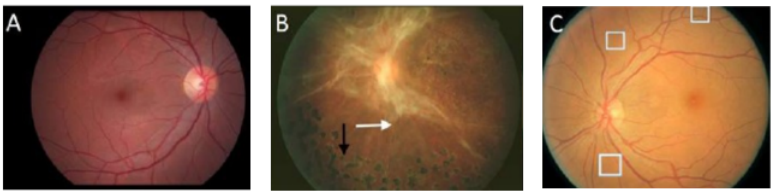


Figure 1.7: Representative retinal images of DR at various stages of the disease, as labeled: A- normal, B- end stage, C- early stage. Arrows in B point to pathological indications. White boxes in C enclose very small lesions that the CNNs have difficulty discerning. Figure extracted from [10].

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.

All these image classification problems are mainly solved by AI methods. But what is exactly AI and what are the main concepts of AI? The next chapter presents the general concepts to understand AI methodology to solve such problems.

Chapter 2

General concepts of AI

This chapter presents the key concepts of AI such as Machine Learning (ML) and Deep Learning (DL) as well as how it is related to the image classification problem.

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 [49].

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 [36].

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, a relevant and useful set of data is needed 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.

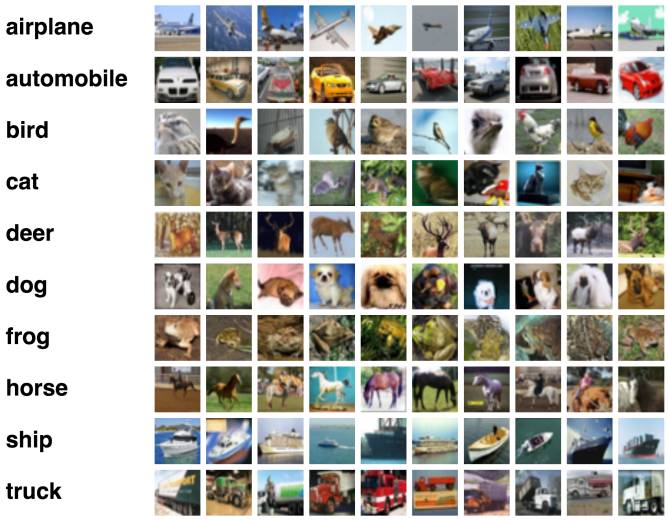


Figure 2.1: Example of training dataset : CIFAR-10. Figure extracted from [3].

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 [37].

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.

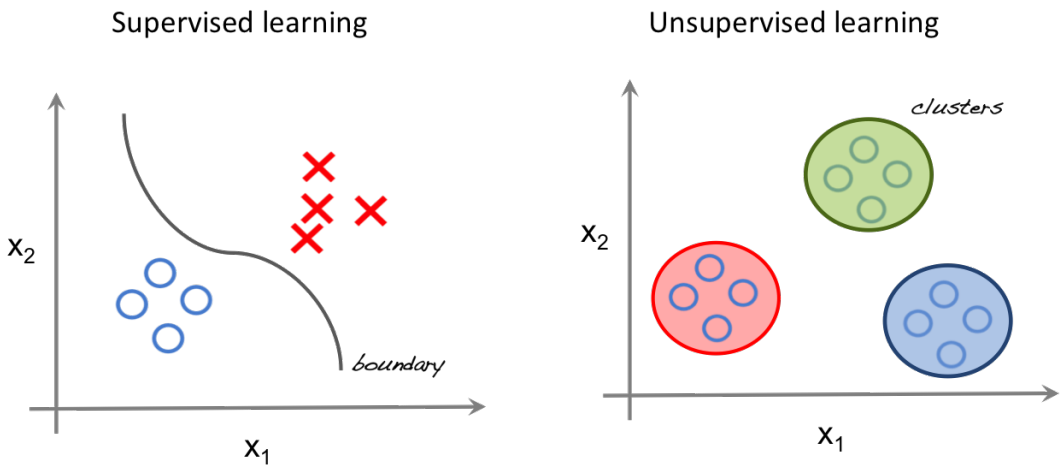


Figure 2.2: Supervised learning VS Unsupervised learning. In supervised learning, the separation and the categories are well defined while in unsupervised learning, clusters have to be found. Figure extracted from [41]

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 [11]. 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 [11]. The objective is 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 [20] .

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.

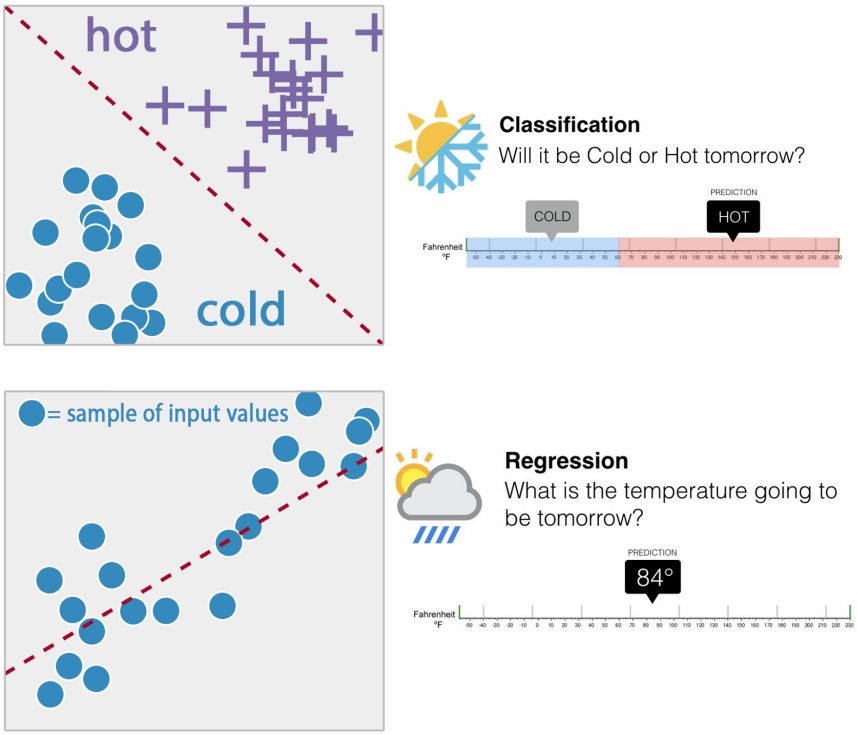


Figure 2.3: Difference between classification and regression on a weather example. For classification, a clear separation between two classes is done: hot and cold. For regression, the objective is to mean the degrees values given on input by a certain function. Figure adapted from [16]

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 [23].

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.

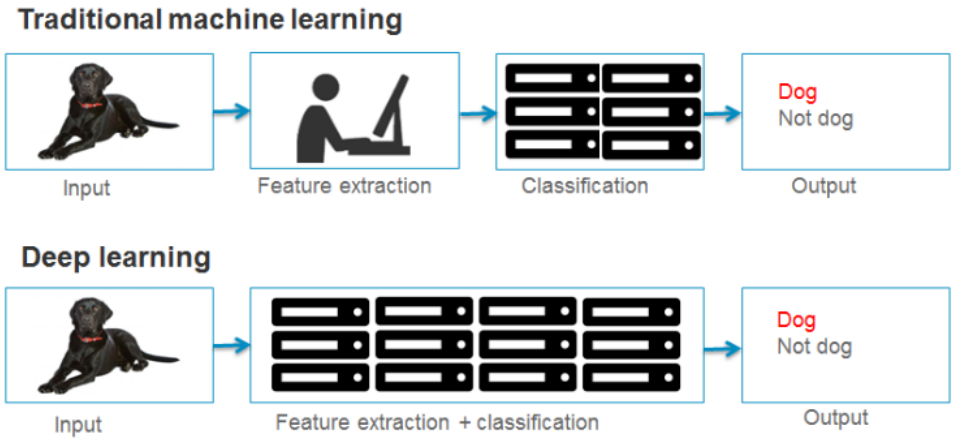


Figure 2.4: Illustration of the main difference between machine learning and deep learning

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 [19]. 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.

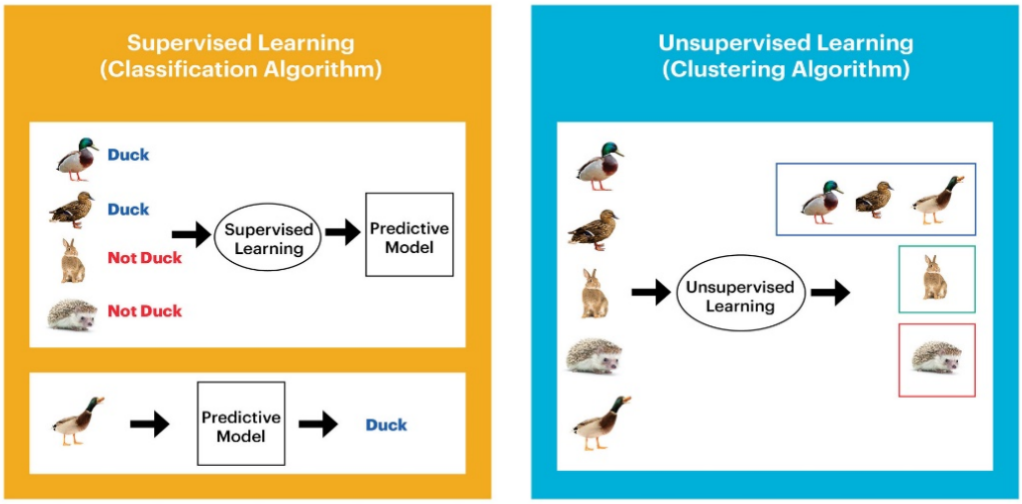


Figure 2.5: Supervised classification VS Unsupervised classification. Figure extracted from [55]

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. To decide which method is more adapted to which situation, it is necessary to define them. This is the purpose of the next chapter.

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 a prediction is validated.

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, their respective functions to create elements that will act similarly in our ANN have to be defined.

- **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, an association between these biological elements and artificial ones following Table 3.1 can be done. 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.

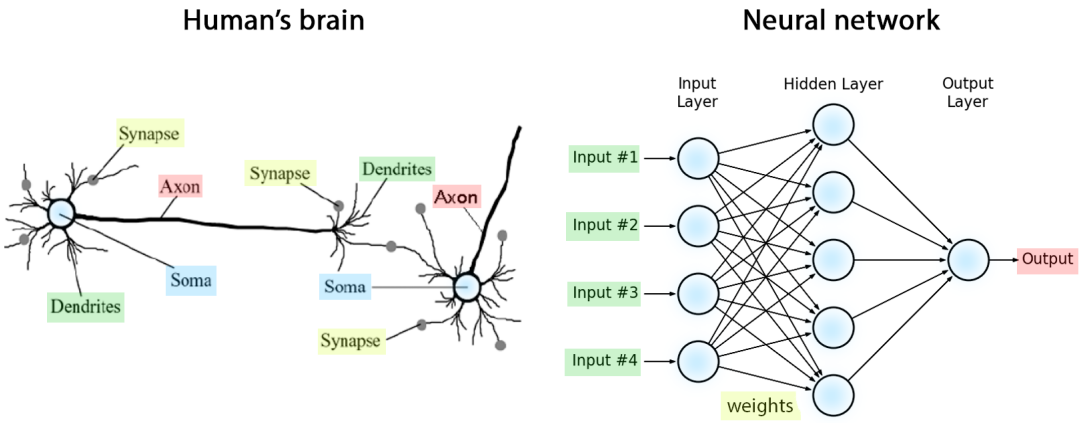


Figure 3.1: Comparison between a biological neural network and an ANN. The same color associates elements with the same function in a biological and artificial point of view.

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, an architecture of neural network composed by a set of neurons and links connecting neurons such that each link has a synaptic weight is needed. Multiple neural network architectures exist such as single-layer or multi-layer perceptron. Inevitably, a learning algorithm to learn is needed. Linear regression, logistic regression, decision tree or SVM are some examples of learning algorithms. The last one cited make object of section 3.3.

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

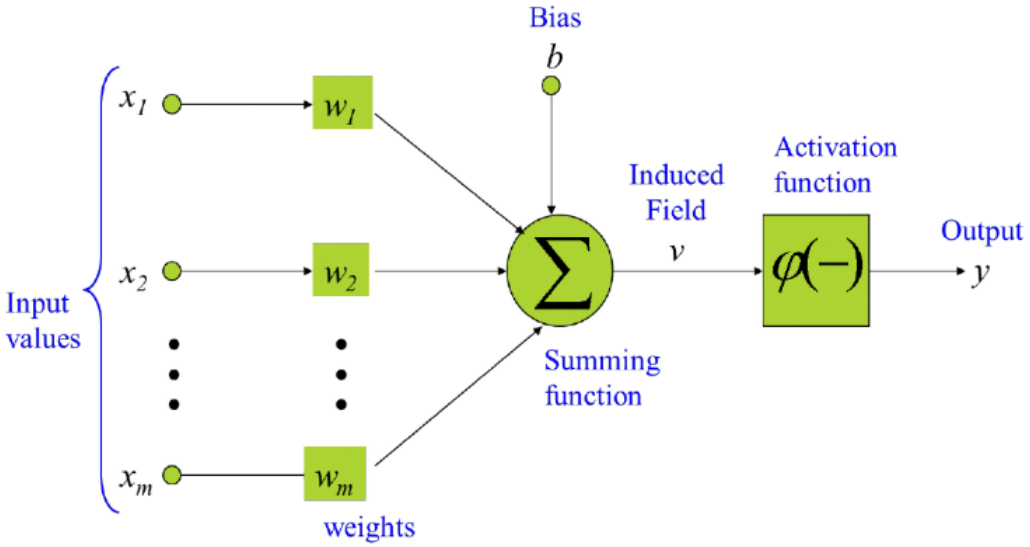


Figure 3.2: General representation of a neural network. Figure extracted from [22].

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, 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.

3.2 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. If we take the example of a 1000x1000-pixel image with RGB color channels, we obtain 3 million weights which is a too large amount to process efficiently at scale with full connectivity. 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, a filter of a fixed size $N \times N$ is applied 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.

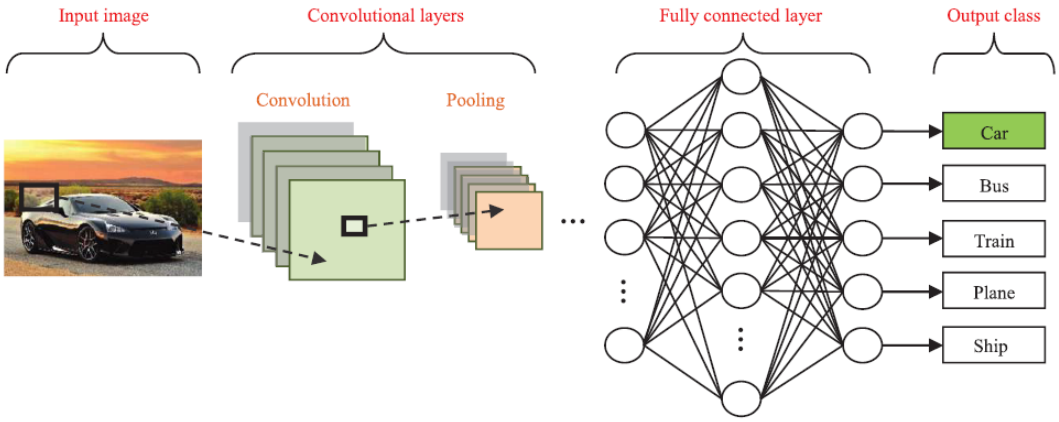


Figure 3.3: CNN image classification pipeline. Figure extracted from [50].

3.2.1 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 the filter to every window of size 3x3 of the 5x5 image is applied. Thus, the filter will be applied and shifted 9 times.

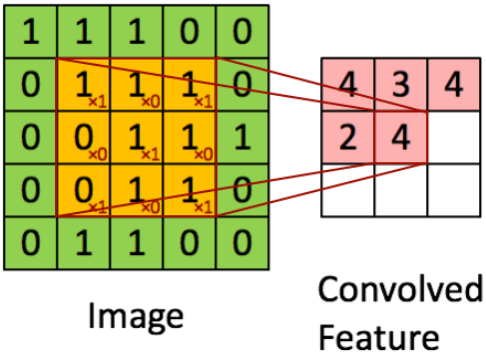


Figure 3.4: Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature. Figure extracted from [39]

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 [50]:

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

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

3.2.2 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 de-noising 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.

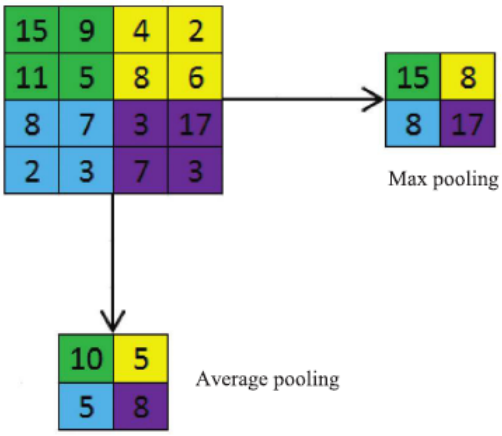


Figure 3.5: Average pooling vs Max pooling. Figure extracted from [50].

3.2.3 Fully connected layers

Now that these high level features are detected 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 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.

3.3 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.

3.3.1 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** [8].

3.3.1.1 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, it denotes 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, π_1 defines 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, π_1 and π_2 are defined 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 $\pi_2 - \pi_1$, it concludes on:

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

[1] Defined in subsubsection 2.1.1.2

Thus, to increase the margin, $\mid w \mid$ has to be minimized. 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 w and b values are saved and keep going or the parameters are adjusted, 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 check: **do**
  if $y_i \cdot (w \cdot x_i + b) = 1$ **then**
    x_i = SV point
    Correctly classified ⇒ Save parameters b and w
  else if $y_i \cdot (w \cdot x_i + b) > 1$ **then**
    Correctly classified ⇒ Save parameters b and w
  else
    Incorrectly classified ⇒ Adjust parameters b and w

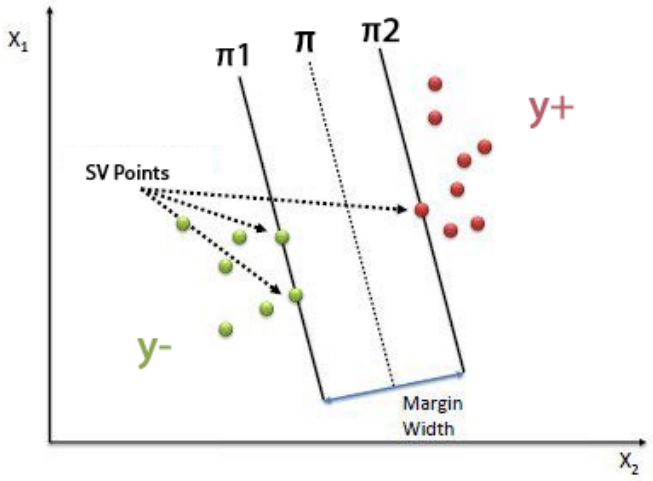


Figure 3.6: SVM hyperplane and margin representation. Figure extracted from [52].

3.3.1.2 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. A non-linear transformation $\phi : \mathbb{R} \rightarrow H$ is applied 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.

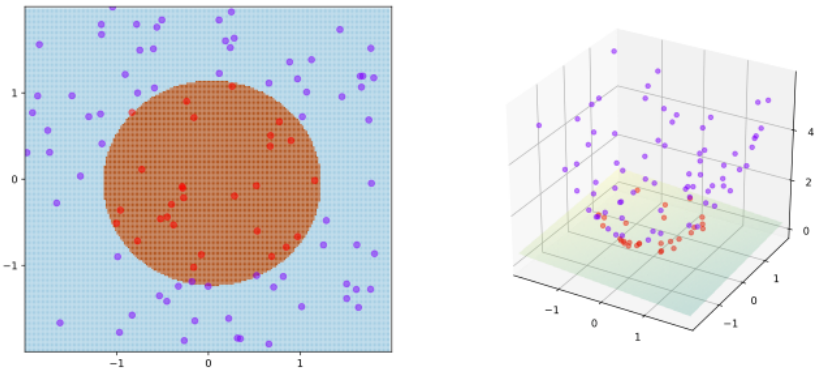


Figure 3.7: A training example of SVM with kernel given by $\phi((a, b)) = (a, b, a^2 + b^2)$ and thus goes from a 2D space to a 3D space defined by $K(x, y) = x \cdot y + x^2 y^2$. The training points are mapped to a 3-dimensional space where a separating hyperplane can be easily found. Figure extracted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

[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] `https://commons.wikimedia.org/wiki/User:Shiyu_Ji#/media/File:Kernel_trick_idea.svg`

3.4 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 the performance of a classification model is computed to finally validate the prediction or not. More precisely, the statistical measures of the performance of a binary classification test are called "sensitivity" and "specificity".

3.4.1 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, the accuracy of the model can be computed 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.

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

3.4.1.1 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, here are the different values tackled previously computed to evaluate the model.

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

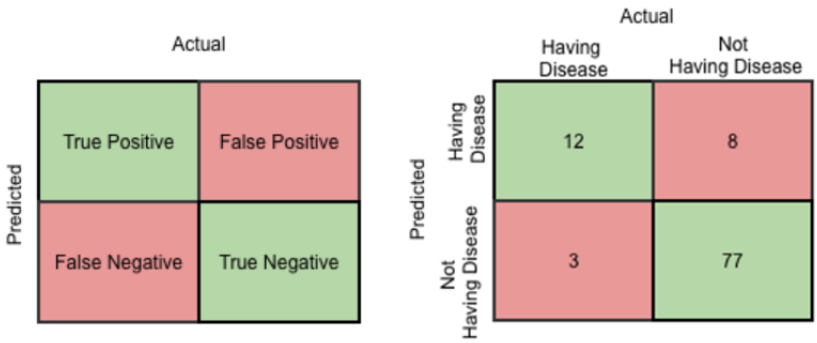


Figure 3.8: A confusion matrix prototype and an application to the above example. $TP = 12$, $FP = 8$, $TN = 3$, $TN = 77$.

3.4.2 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 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 3.4.1.1 , higher the AUC is, better the model is at distinguishing between patients with disease and patients with no disease.

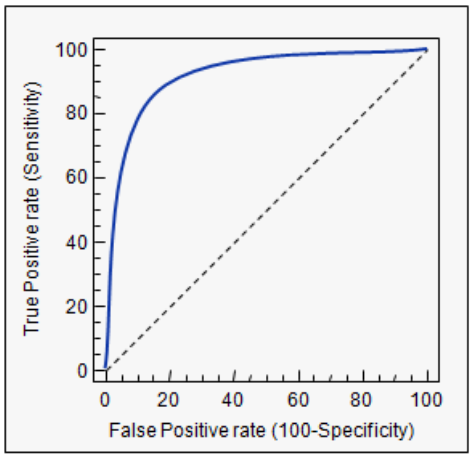


Figure 3.9: A ROC curve. The blue line denotes the change of TPR with different FPR for a model. The AUC is described by the area under this blue line. Figure extracted from [27]

3.4.3 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.

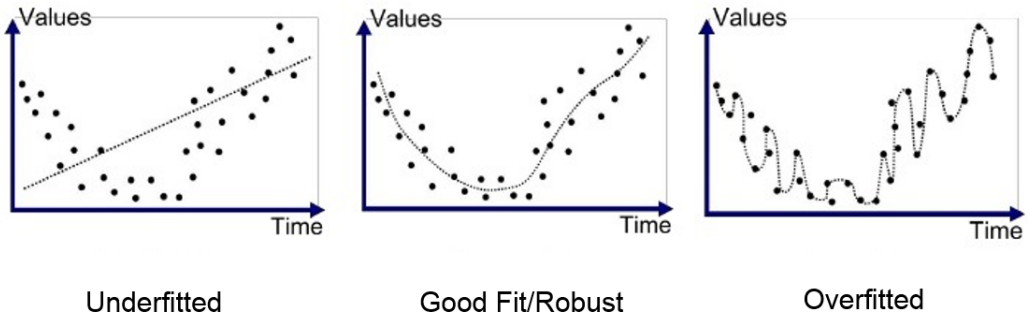


Figure 3.10: Illustration of an underfit, a good fit and an overfit. Figure extracted from [7]

3.4.3.1 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, overfitting is defined 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 observe 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.

3.4.3.2 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.

3.4.3.3 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, the data is partitioned into k subsets, called folds. Then, the algorithm is iteratively trained 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) [51]. This way, multiple mini train-test splits are generated (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.

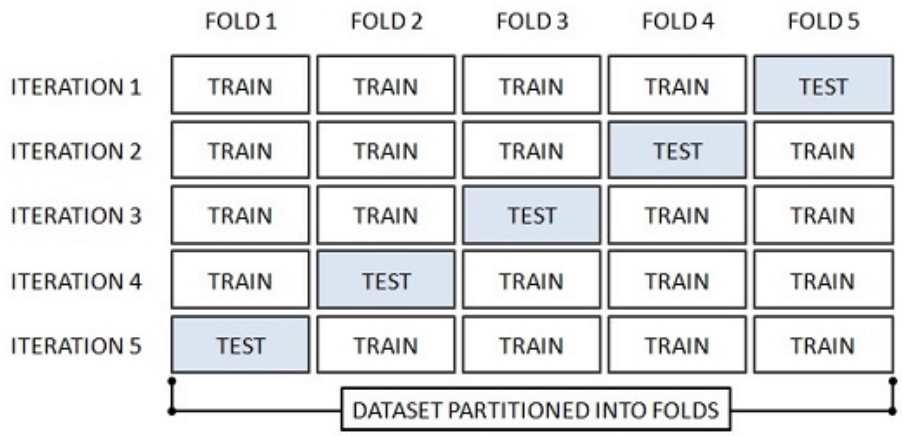


Figure 3.11: Illustration of 5-fold cross-validation. Figure extracted from [25].

Chapter 4

A well-known application: breast cancer detection

Now that multiple methods of ML and DL are defined, let's see how are they applied in a concrete and significant situation such as a medical problem. 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" [15] 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 [24].

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 [24]. 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.

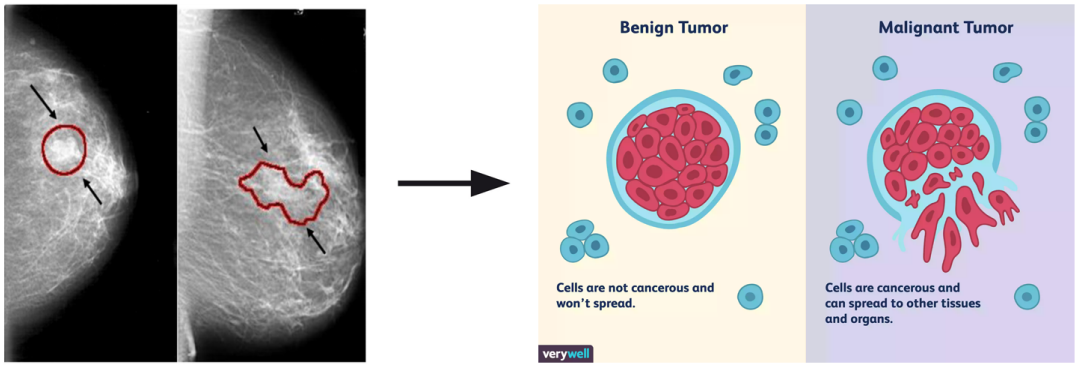


Figure 4.1: Illustration of the different types of tumor. (a) Mass, (b) MCs, (c) Benign, (d) Malignant. First, the tumor can be masses or MCs and no matter what type, they are either benign or malignant. (a) and (b) images are from [15] and (c) and (d) from Joshua Seong ©Verywell, 2017.

Therefore, the purpose of the work of *Ragab et al.* [15] 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 [15] 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 [15]:

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 [18]:

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

   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.

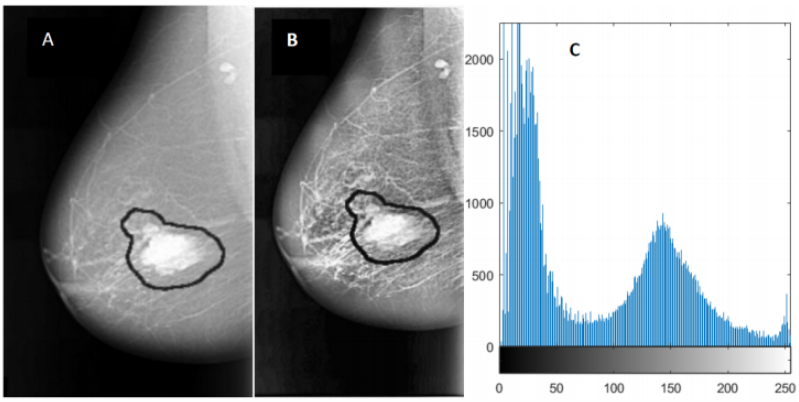


Figure 4.2: Mammogram enhancement using CLAHE. (A) Original malignant mass case extracted from DDSM,(B) Enhanced image using CLAHE, and (C) Histogram representation of the image. Figures extracted from [15]

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 section 3.2 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 [15].

[1] An *histogram* is an accurate representation of the distribution of numerical data. (via *Wikipedia* [51] definition)

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 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 [15].

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 [15]:

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.

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

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

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.

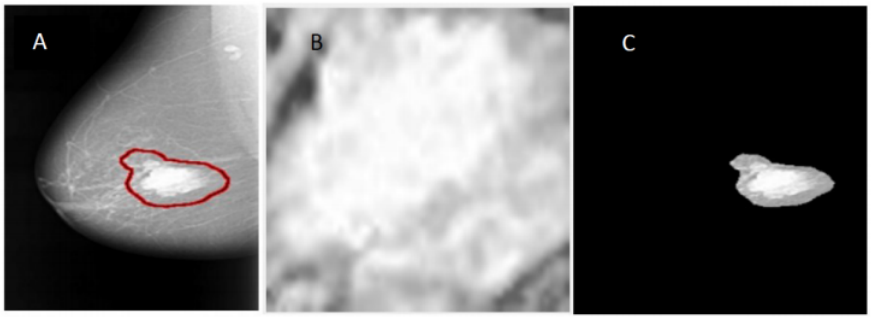


Figure 4.3: Image segmentation using above methodology. (A) Original malignant mass case extracted from DDSM, (B) enhanced ROI extracted manually using circular contours, and (C) enhanced ROI extracted automatically by the region based method. Figures extracted from [15].

4.2.3 Feature extraction

The next step after image enhancement and segmentation is to extract features from these preprocessed 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 section 3.2. Concerning the activation function, AlexNet uses Rectified Linear Units (ReLUs) defined by $f(x) = max(0, x)$ which turns every negative values to zero.

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 [47]:

- **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.

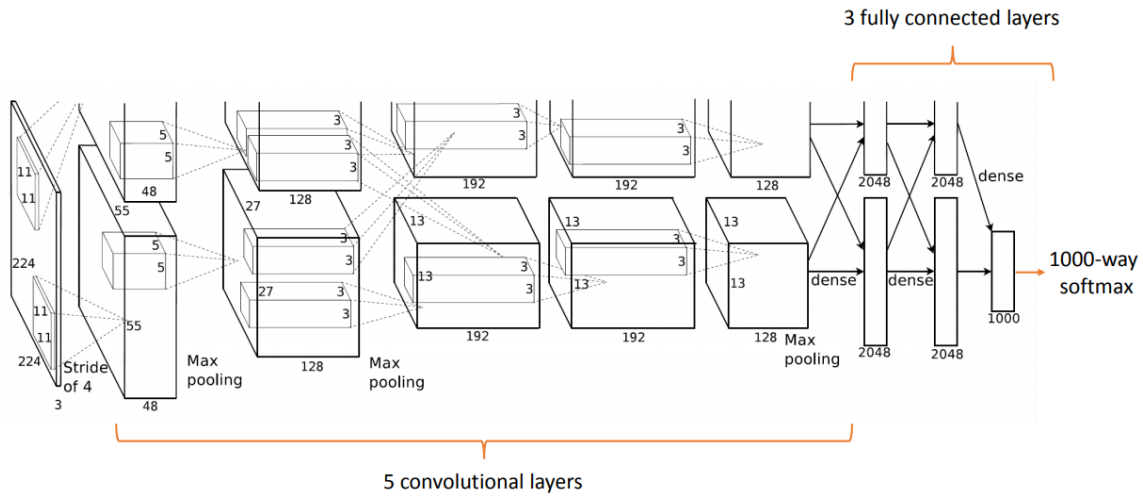


Figure 4.4: Illustration of AlexNet architecture. Figure extracted from [2].

- **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 [47].
- **Geometrical features:** regularity of the border of a lesion is often used to classify lesions. The authors of [47] have incorporated 5 topology descriptor that may help to detect the disease. These are eccentricity, convexity, compactness, circular variance and elliptic variance [47].

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 3.3.

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 3.4 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.

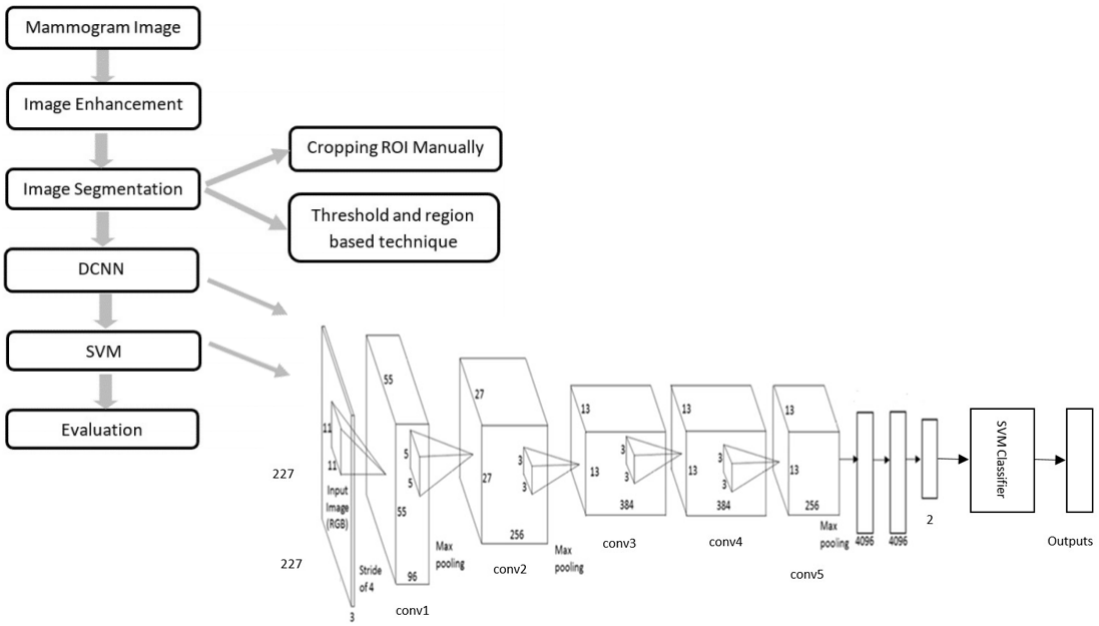


Figure 4.5: Summary of the breast cancer detection methodology of *Regab et al.*'s paper [15].

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 [15].

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 [15].

Chapter 5

My experiments: a model prototype

After discussing theoretically about the different existing 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 [3] based on the CNN implementation of Jason Brownlee [9] and Dan Nelson [28]. 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 section 3.2, 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 subsection 3.2.2), 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 [13].

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 [28]:

```
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 subsection 3.2.2 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 [13]:

```
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 subsection 3.2.3, 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.

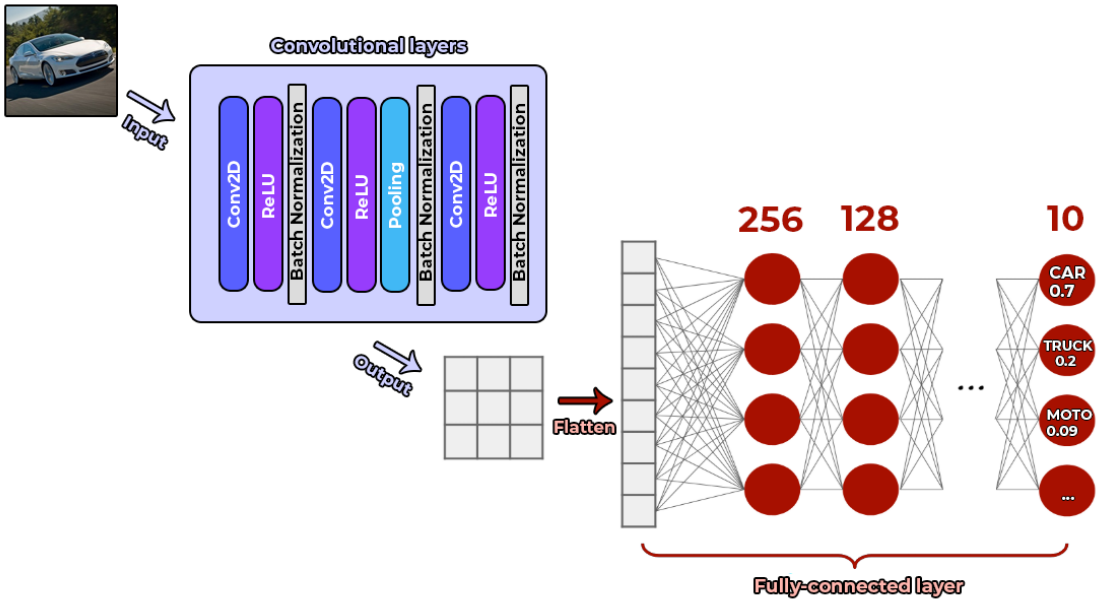


Figure 5.2: The baseline of the CNN implemented. In this example, the input image is a car and the CNN model outputs "car" as highest probability (0.7) and is then assigning the input image as "car" class.

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

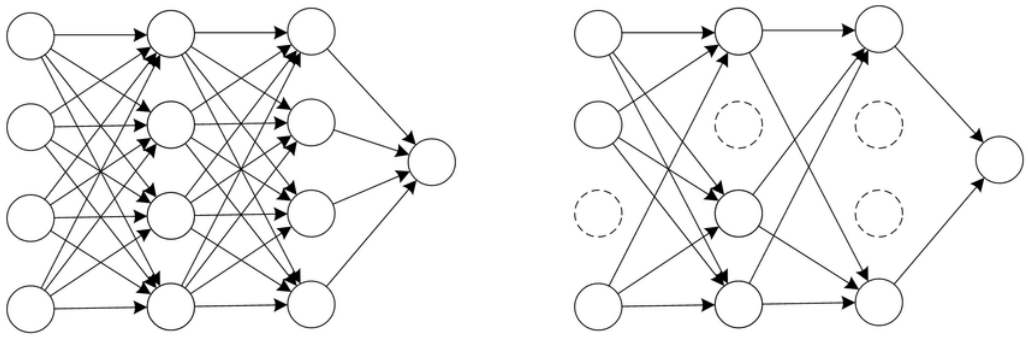
Also at compilation step, you need to specify an optimizer. Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the losses.

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 [38].

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. Figure extracted from [43]

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.

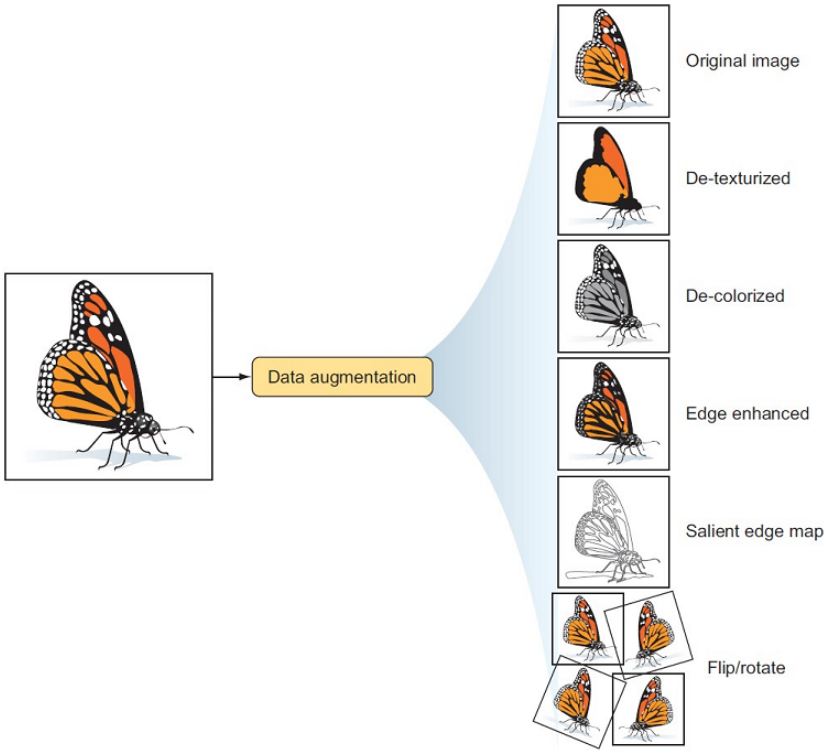


Figure 5.4: Some image data augmentation examples. Figure extracted from [32]

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.

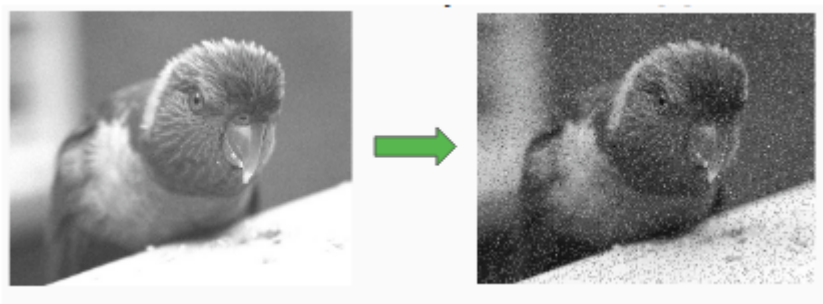


Figure 5.5: Illustration of adding Gaussian noise to an image. The image becomes blurry.

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 ≈ 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.

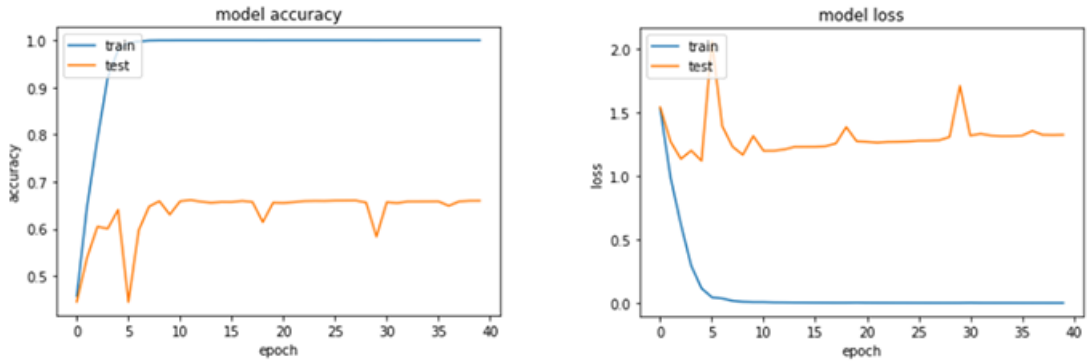


Figure 5.6: Experiment 1 result: model accuracy and lost over epochs. The blue line is the experiment on train set and the orange line on test set.

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.

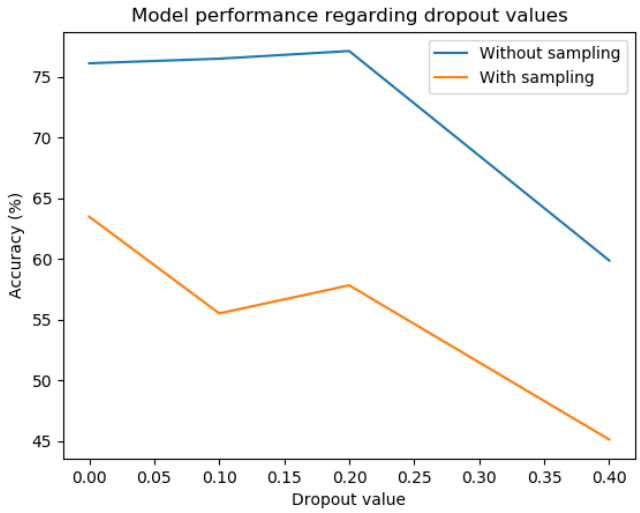


Figure 5.7: Experiment 3 result graph: the blue line is the experiment without sampling and the orange line is the one with sampling. $0.2 \rightarrow 0.5$ refers to the experiment using increasing dropout values.

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.

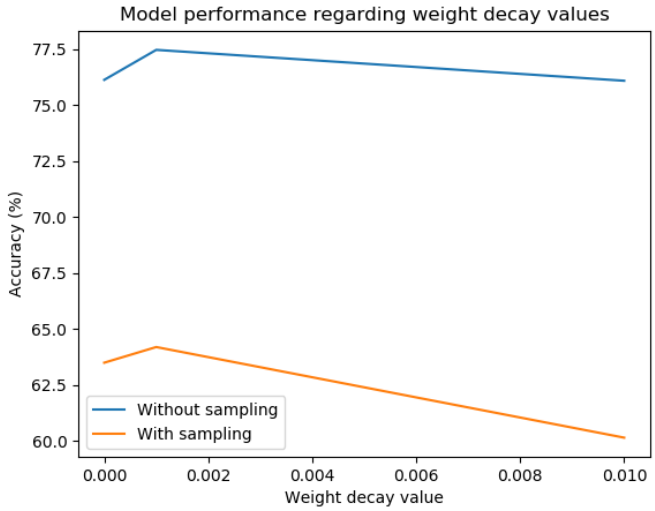


Figure 5.8: Experiment 4 result: the blue line is the experiment without sampling and the orange line is the one with sampling.

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.

Noise value	Accuracy (no sampling)	Accuracy (with sampling)
0	**76.13%**	63.490%
0.1	75.520%	**64.760%**
0.2	75.960%	60.710%
0.4	75.230%	62.080%

Table 5.5: Experiment 6 results table.

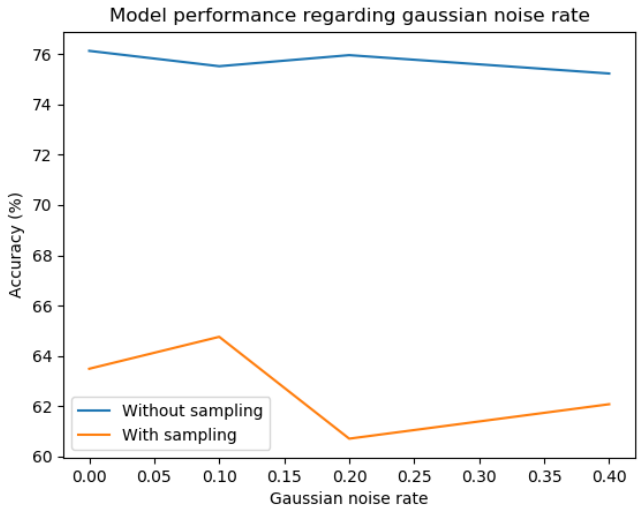


Figure 5.9: Experiment 6 result: the blue line is the experiment without sampling and the orange line is the one with sampling.

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. The best results obtained by each experiment are summarized in the Table 5.6 below. Following these results, 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.

	ACC without sampling	ACC with sampling
Exp 1 (epochs)	65%	65%
Exp 2 (optimizer)	63.490%	63.490%
Exp 3 (dropout)	78.140%	63.490%
Exp 4 (weight decay)	77.490%	64.190%
Exp 5 (data aug)	78.410%	64.810%
Exp 6 (noise)	76.130%	64.760%

Table 5.6: Summary table of the **best** results obtained for the experiments on CIFAR-10 dataset. These are corresponding to the following parameters : (Exp 1) betweeen 10 and 40 epochs ; (Exp 2) ADAGRAD ; (Exp 3) dropout=0.2, dropout=0 ; (Exp 4) weight decay = 0.001 ; (Exp 5) data aug = yes, (Exp 6) noise=0.1, noise=0. The result highlighted in green is the best result obtained. The result highlighted in red is the worst result obtained.

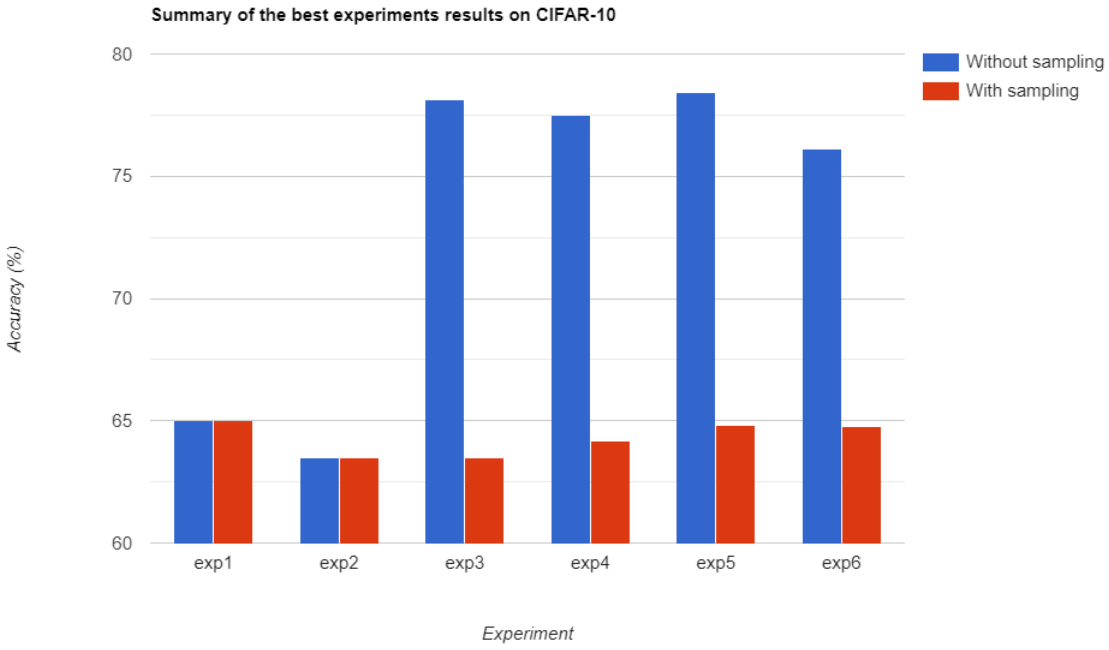


Figure 5.10: Summary graph of the **best** results obtained for the experiments on CIFAR-10 dataset. The blue bars correspond to the results without sampling and the red bars to the results with sampling.

Chapter 6

My experiments: Identification of Leukemia Subtypes from microscopic images

After experimenting the created model on a non-medical dataset, this chapter approaches the second main part of the thesis: **the application on biomedical data**. The following experiments are done on the identification of different leukemia subtypes. Leukemia is a malignant progressive disease that results in high numbers of abnormal blood cells in the bone marrow and other blood-forming organs. These abnormal blood cells suppress the production of normal blood cells leading to multiple symptoms. The main method to detect these abnormal blood cells is from microscopic images of blood's content. Moreover, leukemia is subdivided into a variety of large group which can also be detected by these microscopic images. In this chapter, we first tackle the detection of leukemia as a binary classification (ill/healthy) using the ALL-IDB dataset [40]. In a second step, the different types of leukemia make object of a multi-class image classification problem. Experiments on these different types of leukemia will use the DEMIR-LEUKEMIA dataset [29]. This allows us to **distinguish the difference between medical and non-medical images** previously studied (CIFAR10). Are the methods the same? Are the regularizers used for non-medical images relevant for the medical ones? This chapter is based on the work of Ahmed et al [29].

6.1 Subtypes of leukemia

The two main subdivisions of leukemia are acute and chronic forms and can be defined following the National Cancer Institute:

- **Acute** leukemia: A rapidly progressing cancer that starts in blood-forming tissue such as the bone marrow, and causes large numbers of white blood cells to be produced and enter the blood stream[24]. Acute leukemia is then characterized by a rapid increase in the number of immature blood cells and needs to treated quickly since the cancer can evolve quickly.
- **Chronic** leukemia: A slowly progressing cancer that starts in blood-forming tissues such as the bone marrow, and causes large numbers of white blood cells to be produced and enter the blood stream [24]. Chronic leukemia typically takes months or years to progress,

the cells are produced at a much higher rate than normal, resulting in many abnormal white blood cells and is generally not diagnosed until symptoms, such as enlarged lymph nodes, appear.

Acute and chronic leukemia can be subdivided into 2 more classes regarding which kind of blood cells is affected. These 2 classes are called lymphocitic and myelogenous leukemias.

- **Lymphocitic** leukemia: A type of cancer in which the bone marrow makes too many lymphocytes (white blood cells). - following the NCI [24] definition.
- **Myelogenous** leukemia: the cancerous change takes place in a type of marrow cell that normally goes on to form red blood cells, some other types of white cells, and platelets. - following *Wikipedia* [51] definition.

There exists many more specific leukemia types but we will only approach these 4 main subtypes in the context of this document. These 4 subtypes of leukemia are summed up on Table 6.1 and illustrated by Figure 6.1. We can see the importance of determining the leukemia subtype of the patient. A chronic leukemia is not as urgent as an acute leukemia that needs to be treated as soon as possible.

Cell type	Acute	Chronic
Lymphocitic	Acute Lymphocitic Leukemia (**ALL**)	Chronic Lymphocitic Leukemia (**CLL**)
Myelogenous	Acute Myelogenous Leukemia (**AML**)	Chronic Myelogenous Leukemia (**CML**)

Table 6.1: Summary of the 4 main leukemia subtypes tackled in the document.

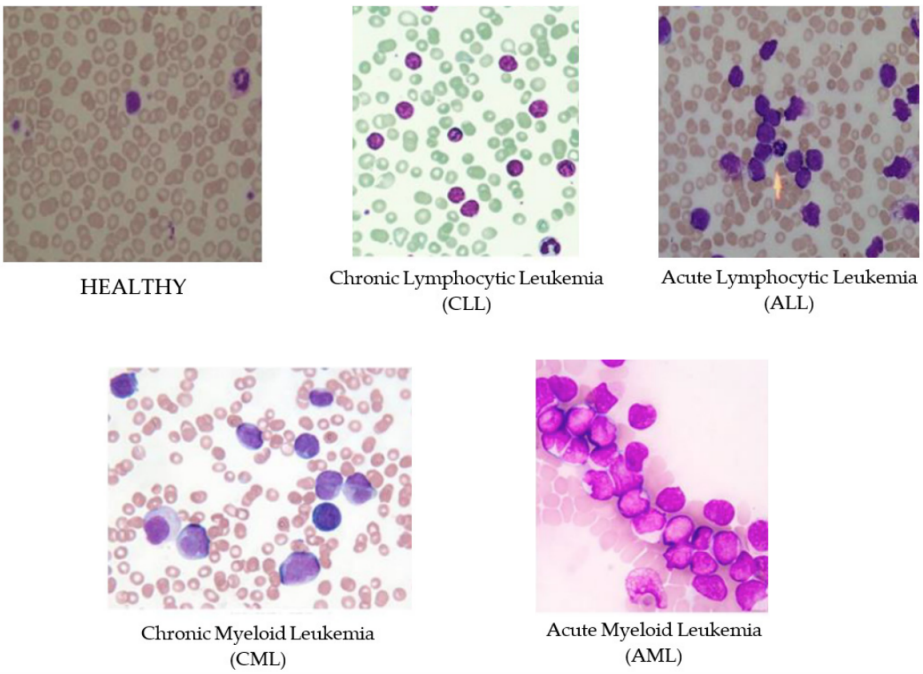


Figure 6.1: Sample images of four different types of Leukemia. From left to right, HEALTHY, Chronic Lymphocytic Leukemia (CLL), Acute Lymphocytic Leukemia (ALL), Chronic Myeloid Leukemia (CML), Acute Myeloid Leukemia (AML). Figure extracted from [29]

6.2 ALL-IDB dataset

The ALL-IDB dataset, fully named *Acute Lymphoblastic Leukemia Image Database for Image Processing*, is provided by Fabio Scotti from the University of Milan [40]. As its name indicates, the ALL-IDB dataset only contains microscopic images from patients that have the ALL type of leukemia when labelled positive to illness. For each image in the dataset, the classification/position of ALL lymphoblasts is provided by expert oncologists.

The ALL-IDB dataset is divided into 2 sets of data: ALL-IDB1 and ALL-IDB2.

- ALL-IDB1 contains large images of average size 2592 x 1944 which contains about 39000 blood elements spread into 108 images collected in September 2015. In these 108 images, 50 are labelled positive to leukemia and 58 labelled healthy. The ALL-IDB1 image files are named with the notation ImXXX_Y.jpg where XXX is a 3-digit integer counter and Y is a boolean digit equal to 0 is no blast cells are present (healthy), and equal to 1 if at least one blast cell is present in the image (ill). As the images are large, ALL-IDB1 is quite useful for testing different image preprocessing methods such as segmentation or data augmentation to see their efficiency in the final classification accuracy. As the dataset also provides the position of the blast cells in the image, this information may be used to estimate the location of the centroids of the blast cells as well as the number of ALL blast.
- ALL-IDB2 is a collection of cropped area of interest of normal and blast cells that belongs to the ALL-IDB1 dataset. These images are then somehow already preprocessed regarding the region of interest. Their size is ≈ 250 x 250. ALL-IDB2 images have similar gray level properties to the images of the ALL-IDB1, except the image dimensions. The images are annotated similarly to the images from ALL-IDB1. ALL-IDB2 contains 260 images including 130 ALL cases and 130 healthy cases.

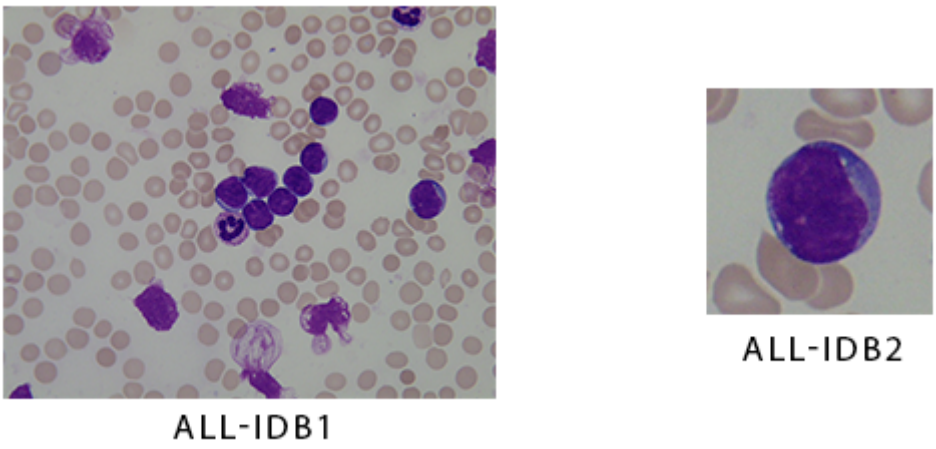


Figure 6.2: Image examples of ALL-IDB1 and ALL-IDB2.

6.3 DEMIR-LEUKEMIA dataset

The DEMIR-LEUKEMIA dataset has been created by *Ahmed et al.* [29] from the The American Society of Hematology (ASH) Image Bank [30]. Since ALL-IDB only deals with ALL leukemia subtype, the DEMIR-LEUKEMIA dataset is complementary to ALL-IDB by providing a collection of CLL, AML and CML leukemia subtypes images.

ASH Image Bank is publicly available on the Web and includes a comprehensive collection of images related to a wide range of hematological topics. In this study, *Ahmed et al.* have selected all available blood cell images annotated with leukemia, including any of CLL, AML and CML subtypes.

DEMIR-LEUKEMIA contains 177 AML, 185 CLL and 185 CML images. It contains no healthy cases. The sizes of the images are very variables, it can goes from 200 x 200 to 1000 x 1000. As *Ahmed et al.* have chosen the images with no specific criteria except the leukemia subtype, the images are very disparate. When using this dataset, the preprocessing step is then mandatory.

Table 6.2 sums up the content of the two original datasets used for leukemia subtypes detection, i.e. the number of samples.

Dataset	ALL	AML	CLL	CML	HEALTHY	Total
ALL-IDB1	50	0	0	0	58	108
ALL-IDB2	130	0	0	0	130	260
ALL-IDB	180	0	0	0	188	368
DEMIR-LEUKEMIA	0	177	185	185	0	547
ALL-IDB + DEMIR-L	180	177	185	185	188	915

Table 6.2: Summary of the ALL-IDB and DEMIR-LEUKEMIA datasets used for leukemia subtypes detection.

6.4 Data augmentation on datasets

In addition to the original datasets, these data have also been augmented using multiple minor images transformations in order to have larger datasets. Data augmentation techniques were widely utilized to increase the dataset size and avoid memorization. As the images change and the pixels are moved, the neural network would think these are distinct. The transformations that have been applied are listed below:

- **Rotation**: random rotations between -25° (left rotation) and 25° (right rotation). The rotation effect has usually good impact on the neural network as the change is very minor and then the features impacting the label can also be recognized as the original. Nevertheless, the corners of the images are black due to the rotation that can negatively impact the learning of the neural network especially if we manage small images. Indeed, the black area can dominate the image and can be considered as an important feature to determine the label. This rotation has been done using the `sk.transform.rotate()` method from `skimage` library in Python.
- **Horizontal flip**: reverse the image horizontally, the image generated is a mirror-reversal of an original across a horizontal axis image. Pixels were moved horizontally from one half of the image to the other half. In contrary to simple rotation, flipping the image does not imply black areas and is then theoretically beneficial to the learning.

- **Vertical flip**: reverse the image vertically, the image generated is a mirror-reversal of an original across a vertical axis image. Pixels were moved vertically from one half of the image to the other half.
- **Noise**: random noise added to the image. As seen in the previous experiments done on the CIFAR10 dataset (see subsubsection 5.4.2.6), gaussian noise has relatively a good regularizer impact on the network. We then use the assumption that it would also have a positive impact on medical images. This noise has been added using the `sk.util.random_noise()` method from `skimage` library in Python.
- **Uniform filter**: similarly as noise, the uniform filter that we apply there is to reduce the sharpness of the image, i.e. the image is somehow blurred and can easily be compared as noise. This uniform filter has been added using the `ndimage.uniform_filter()` method from `scipy` library in Python.
- **Random gamma**: gamma defines the relationship between a pixel's numerical value and its actual luminance. In other words, gamma impacts the contrast and the luminosity of the image. In this study, modifying the gamma could be a reliable transformation to detect features as the edges of the shapes can be made more visible. This uniform filter has been done using the `exposure.adjust_gamma()` method from `skimage` library in Python.

All these transformations are illustrated by order in Figure 6.3.

When using data augmentation, people are also usually applying shifts and crops to their images. Here, I decided to not use these two as shifting or cropping an image is more likely affected by information loss and could be then not relevant for the model's enhancement.

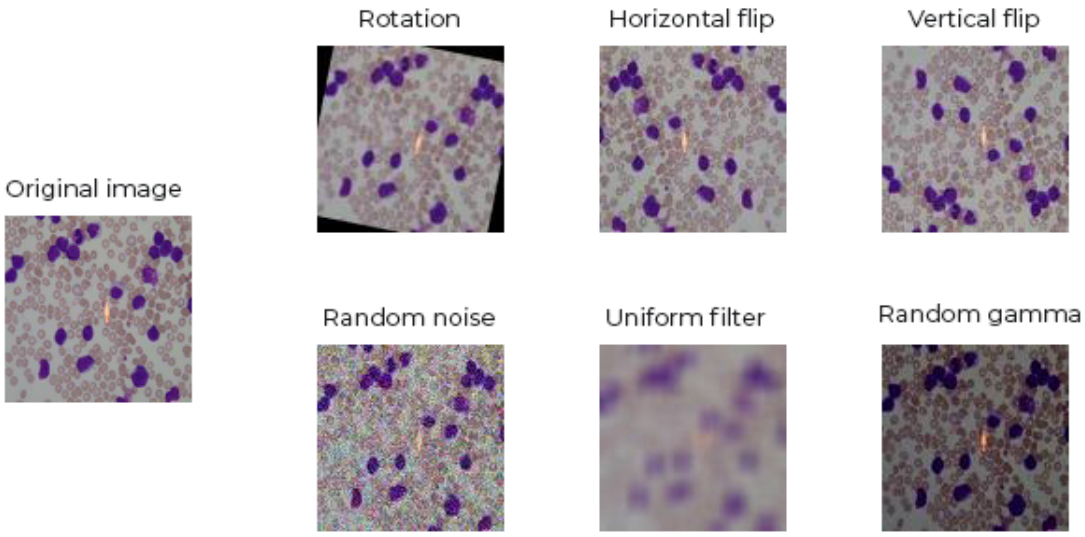


Figure 6.3: Image transformations used for data augmentation. From left to right: original, rotation, horizontal flip, vertical flip, random noise, uniform filter and random gamma.

To create the transformed dataset, we apply randomly one of the transformations described above to each image from the original dataset. Each image has been augmented, thus we obtain a stratified dataset where the proportion of each label is not impacted and is similar to the original dataset. In that case, as we are creating one "copy" of each image, we are doubling the dataset size, then the number of images assigned to each label is also doubled. (e.g. if we have 2 classes of 40 "healthy" and 30 "ill", after data augmentation, we obtain 80 "healthy" and 60 "ill")

All the datasets with their corresponding number of images per leukemia subtypes are summarized in Table 6.3.

Dataset	ALL	AML	CLL	CML	HEALTHY	Total
ALL-IDB1	50	0	0	0	58	108
ALL-IDB1 (+ data aug.)	100	0	0	0	116	216
ALL-IDB2	130	0	0	0	130	260
ALL-IDB2 (+ data aug.)	260	0	0	0	260	520
ALL-IDB	180	0	0	0	188	368
ALL-IDB (+ data aug.)	360	0	0	0	376	736
DEMIR-LEUKEMIA	0	177	185	185	0	547
DEMIR-LEUKEMIA (+ data aug.)	0	354	370	370	0	1094
ALL-IDB + DEMIR-L	180	177	185	185	188	915
ALL-IDB + DEMIR-L (+ data aug.)	360	354	370	370	376	1830

Table 6.3: Summary of the ALL-IDB and DEMIR-LEUKEMIA datasets used for leukemia subtypes detection in addition with data augmentation datasets.

6.5 Methodology

The methodology used is based on the article "Identification of Leukemia Subtypes from Microscopic Images Using Convolutional Neural Network" from *Ahmed et al.* [29]. As the article's statement indicates, the model used is a convolutional neural network and can easily be justified (cfr. subsection 5.2.1).

6.5.1 Preprocessing

In the article [29], no preprocessing phase was done to the images and are used from their original sizes and colors. In average, the images from the datasets used are quite large ($\approx$ 1500x1000) and requires a huge processing time if used at this size from a classic student laptop. Because of that, a preprocessing step is mandatory to avoid endless execution. In this case, the preprocessing is defined by:

- Resize of all the images of a fixed size 100x100, illustrated in Figure 6.5 (a).
- In some experiments, we have applied some filters before processing the images: images to **greyscale**, images' **contrast increased** by an arbitrary value, **histogram equalization** of the images which is a method of contrast enhancement. This method consists on spreading out the most frequent intensity to make lower local contrast to gain a higher contrast. The principle of histogram equalization is illustrated in Figure 6.4. A variant of histogram equalization (CLAHE) has been has been used and explained in subsection 4.2.1. These

filters have as objective to enhance the images by highlighting the important features of the images.

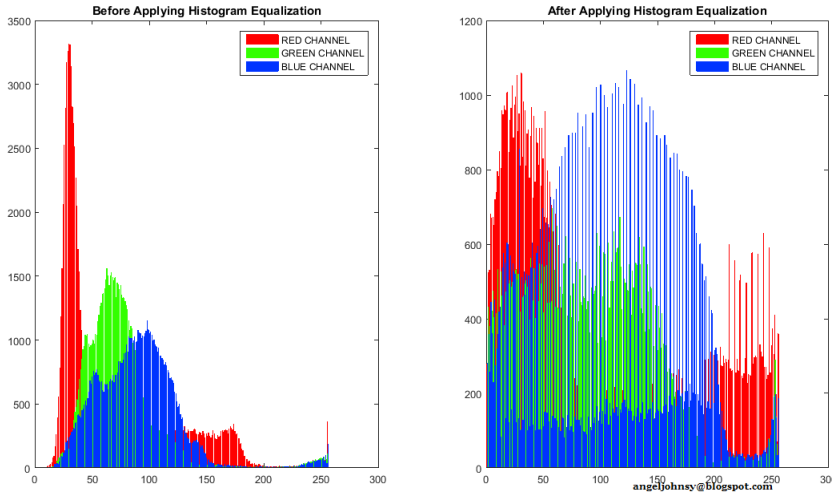


Figure 6.4: Histogram equalization principle. A color histogram of an image represents the number of pixels in each type of color component. We spread out the most frequent intensity values of each color channel. Figure extracted from [5].

These preprocessing steps are illustrated in Figure 6.5.

Moreover, multiple questions are raised from this preprocessing:

- Are informations still available from very small images? What is the capacity of the neural network to find the information?
- What informations are reliable to the training of a neural network? Is color needed to enhance the training? Is the direction of the images an impacting feature, i.e. are rotations and flips really relevant in this type of images?
- What is the best filter to apply on medical images? Which filter facilitates the model to retrieve the information on such images?
- Despite the preprocessing applied, can we still obtain good prediction rate?

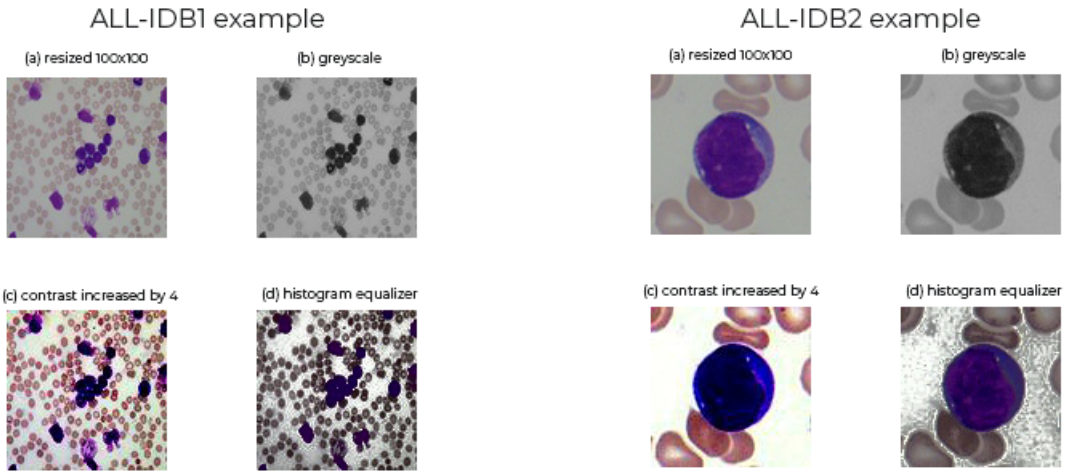


Figure 6.5: Example of preprocessing to an image of the dataset ALL-IDB1 and ALL-IDB2. (a) Image resized to 100x100 pixels, (b) grey-scale applied to the resized image, (c) image contrast increased, (d) histogram equalization applied.

6.5.2 CNN architecture

As reminder, CNN is composed of convolution layers, pooling layers, flattening, and multilayer perceptron. The number of layers and its respective parameters are chosen based on our previous study on CIFAR10 dataset done in chapter 5 and the article cited above [29]. Here below are described the choices done in the further experiments:

- **Convolutional layers:** only 2 convolutional layers as the images are very small, 32 and 64 filters of size 3x3 with ReLU as activation function.
- **Max-Pooling layers:** one pooling layer after each convolutional layer (then total of two) of size 2x2.
- **Flatten layer:** one flatten layer between the convolutional layers and the fully-connected layer to transform the 2-dimensional max-pooled matrix into one dimensional array used as input for the fully-connected layer.
- **Fully-connected layer:** composed of two hidden layers (dense layers) and an output layer. The two hidden layers are respectively composed of 128 neurons and 64 neurons. The output layer contains as much as neurons than possible outputs.

→ Each layer is followed by a Batch normalization.

6.6 Experiments and results

The following experiments are mainly based on the experiments done by Ahmed et al in their article [29]. However, some additional experiments are done to understand better how the model reacts and what are the best configuration to learn and predict better. In addition to my own results, when similar experiments were done, the results obtained by the authors of this article are also taken over in order to compare them and see the impact of the preprocessing.

6.6.1 Experiments

In this study, we have 3 different datasets available: ALL-IDB1, ALL-IDB2 - which contains ALL and HEALTHY cases (described in section 6.2 and DEMIR-LEUKEMIA - which contains AML, CLL and CML cases (described in section 6.3). Having these 3 datasets, experiments will be done separately divided in 3 cases:

- *Case A - ALL-IDB1:* in this case, we only manage a binary classification between the ALL and HEALTHY cases. The images are not segmented and groups multiples blood cells.
- *Case B - ALL-IDB2:* the second set of ALL-IDB contains segmented blood cells images and allow us to determine the potential improvement of the learning thanks to the segmentation. The question that can be raised using ALL-IDB2 is: "Does it give enough information?".
- *Case C - All datasets:* runs on set of all images available: ALL-IDB and DEMIR-LEUKEMIA. This case involves the 4 subtypes of leukemia and the HEALTHY case. There are then 5 classes.

The experiments are realised on 25 epochs and batch size of 32 with ADAM optimizer. These choices have been made by the authors of [29] after multiple experiments that have noticed that working with 25 epochs performs better than longer epoch periods. The test set is defined by 20% of the complete dataset. Each experiment has been repeated 5 times, the training set and the test set always change at each repetition. Note that this is not a 5-fold cross-validation since the train/test sets split is done randomly. The choice of not using a 5-fold cross-validation is due to the format and the order of the dataset which contains first all the ALL cases then all the AML cases then all the CLL cases and finally the CML cases. Using cross-validation would therefore not have been reliable since the test set would only contain one class.

- *Experiment 1 - Baseline:* 5 runs on the resized images of size 100x100 in color using the baseline CNN: no dropout and no data augmentation.
- *Experiment 2 - With filters:* runs on the resized images with filters applied (greyscale, contrast increased or histogram equalization) of size 100x100 using the baseline CNN: no dropout and no data augmentation. Each filter is tested separately in order to see the impact of each filter. 5 runs are then done for each filter which involves a total of 15 runs.
- *Experiment 3 - Size 200x200:* 5 runs with larger resized images of size 200x200 in color using the baseline CNN: no dropout and no data augmentation.
- *Experiment 4 - Noise:* runs on the resized images of size 100x100 with different level of noise applied. The level of noise applied are respectively $[0.1, 0.2, 0.4, 0.6, 0.8]$. We use the baseline CNN. Each noise level is ran 5 times. 25 runs are then done in total.

- *Experiment 5 - Dropout:* 5 runs on the resized images of size 100x100 in color using dropout set to 0.1 with no data augmentation.
- *Experiment 6 - Data augmentation:* 5 runs on the data augmented dataset (described in section 6.4) with no dropout.
- *Experiment 7 - Dropout and data augmentation:* 5 runs on the data augmented dataset (described in section 6.4) using dropout set to 0.1.

6.7 Results

In this section, the results of the experiments described in the previous section are presented. For each experiment, 3 results' graphs are presented:

- *the evolution of the **accuracy** on the **test set** along epochs:* the indicator of how often the classifier was correct.
- *the evolution of the value of **cost function** on the **test set** along epochs:* the cost function quantifies the error between predicted values and expected values and presents it in the form of a single real number [26], i.e. measure of how wrong the model is in terms of its ability to estimate the relationship between X and y. In our case, we want the cost function to be minimized.
- *the evolution of the **accuracy** on **train set** along epochs:* this graph is used to visualise the speed at which the model learns and converges to 100% accuracy (convergence state).

6.7.1 Case A - Results summary for ALL-IDB1 dataset experiments

⚠**The detailed results of the experiments for ALL-IDB1 dataset are available in the appendix section A.1.** It contains detailed graphics, table and description of the results of each experiment done on **ALL-IDB1 dataset**. Each parameter is approached one by one and interpreted concerning their efficiency and the impact on the model's accuracy. Therefore, note that this current section is a **summary** of all these interpretations on obtained results for the first dataset ALL-IDB1.

To summarize the experiments done using ALL-IDB1 dataset, the Table 6.4 recapitulates the average results of each experiment for the accuracy on test and train set as well as the loss value for test set.

As the dataset is very small using small images, using parameters as dropout or noise looks like inefficient given the results obtained. Using the simplest CNN is then best way to learn efficiently. To make the significant features clearer, a preprocessing step using image enhancement (contrast increasing or histogram equalization) seems to be very efficient. It highlights well the edges of the blood cell which is the main feature of these images. Hence, the best result obtained uses the baseline CNN on enhanced images by histogram equalization. We obtain an average of 90.9% accuracy on test set and a loss value of 0.28.

On the other hand, the worst result is obtained by using greyscale filter which means that color feature is quite important to detect specificities of blood cells. We obtain an average accuracy of 51.81%. This worst result is followed by the experiment where we add a huge value of noise (0.8). We obtain an average of 54.54% accuracy on test set and a loss value of 7.32. When using that large amount of noise, the model cannot detect well the different features and then has a large error rate. This average accuracy is also obtained when using dropout without data augmentation. This shows how much using dropout is bad when having such a small dataset (comparable to adding 0.8 of gaussian noise!).

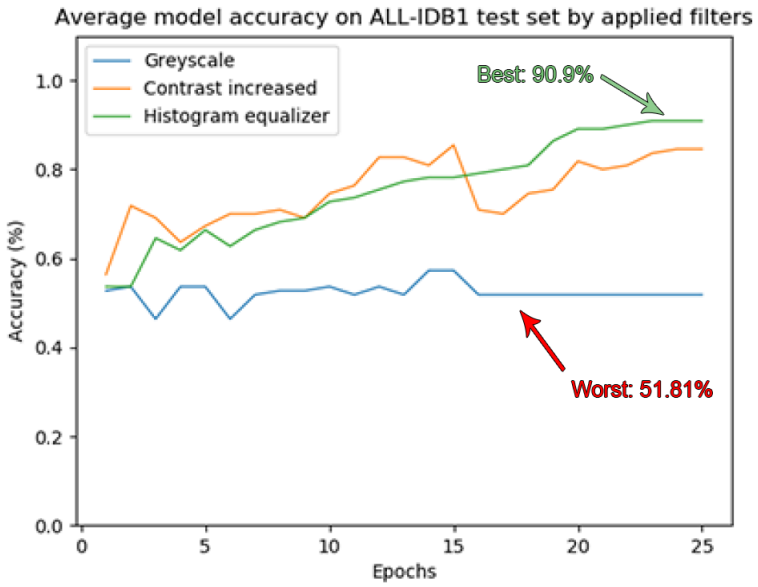


Figure 6.6: Accuracy evolution on test set of the best and the worst result for ALL-IDB1. The worst result is the experiment using greyscale filter (blue line). The best result is the experiment using histogram equalizer (green line).

	ACC-TEST	LOSS-TEST	ACC-TRAIN
Exp A.1 (baseline)	65.45%	3.46	100%
Exp A.2 (grey)	51.81%	6.09	100%
Exp A.2 (contrast)	84.54%	1.43	100%
Exp A.2 (hist. eq.)	90.9%	0.28	100%
Exp A.3 (200x200)	59.08%	2.36	100%
Exp A.4 (noise=0.1)	77.27%	0.67	100%
Exp A.4 (noise=0.2)	63.63%	1.64	100%
Exp A.4 (noise=0.4)	59.09%	1.75	97.67%
Exp A.4 (noise=0.6)	59.09%	4.33	94.18%
Exp A.4 (noise=0.8)	54.54%	7.32	84.88%
Exp A.5 (Dropout)	54.54%	5.78	61.85%
Exp A.6 (Data aug)	72.26%	1.31	99.64%
Exp A.7 (Dropout & Data aug)	65.9%	1.16	64.99%

Table 6.4: Summary table of the results obtained for ALL-IDB1. The results presented in this table are the average results for each experiment. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set. The result highlighted in green is the best result obtained. The result highlighted in red is the worst result obtained.

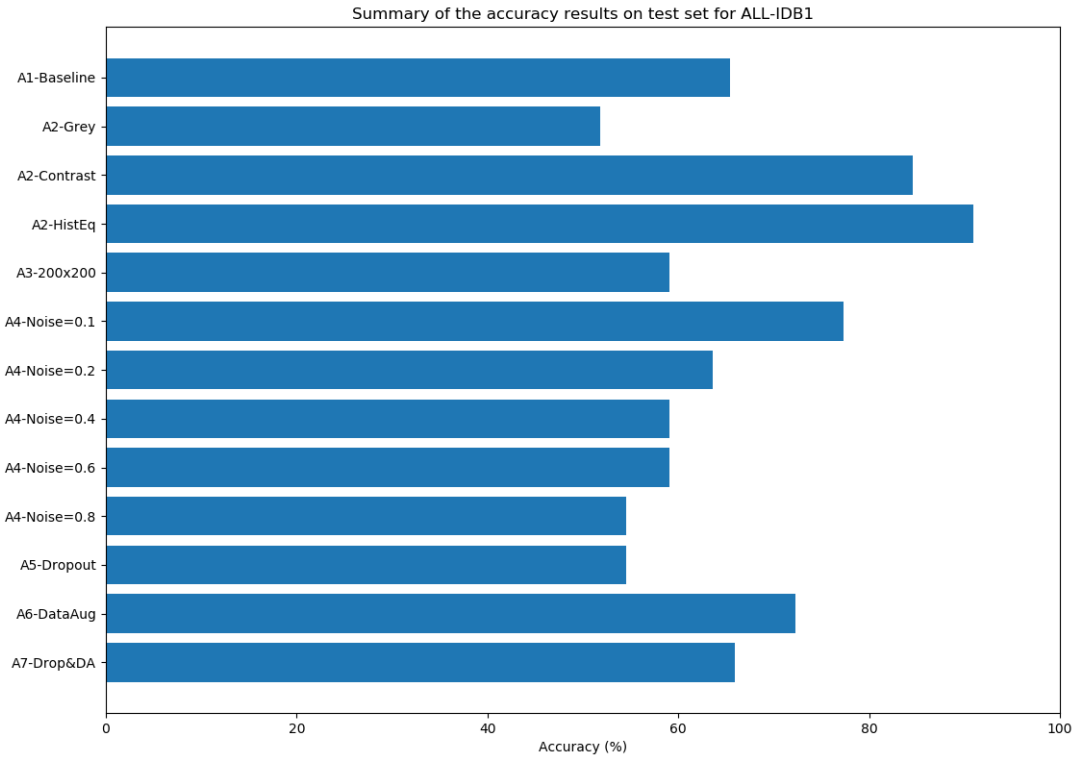


Figure 6.7: Summary graph of the accuracy results obtained on test set for ALL-IDB1.

6.7.2 Case B - Results summary for ALL-IDB2 dataset experiments

⚠**The detailed results of the experiments for ALL-IDB2 dataset are available in the appendix section A.2.** It contains detailed graphics, table and description of the results of each experiment done on **ALL-IDB2 dataset**. Each parameter is approached one by one and interpreted concerning their efficiency and the impact on the model's accuracy. Therefore, note that this current section is a **summary** of all these interpretations on obtained results for the second dataset ALL-IDB2.

To conclude the different experiments done using ALL-IDB2 dataset, the preprocessing step using image segmentation do not show a clear advantage in comparison with the experiments done using ALL-IDB1 with no segmentation. Overall, the results are quite equivalent.

Using contrast augmentation (whether with arbitrary value or with histogram equalization) is the best image enhancement technique to have great performance. Same as ALL-IDB1, using such technique increases the visibility of edges and then facilitates the model's learning. On the other hand, adding a huge value of noise still gets the worst results for the same reasons as ALL-IDB1. This conclusion is then corresponding to using the non-segmented dataset (ALL-IDB1).

However, some clear differences can be observed on some experiments such as the experiment A.2 or A.3 which demonstrate way more good performances by using segmentation. Indeed, the images of ALL-IDB1 are containing more edges and details to highlight as we don't segment them. Therefore, using the experiment 2 (which brings out these edges) or using the experiment 3 (that enlarges these images and then makes the details more visible) are more efficient for the non-segmented dataset ALL-IDB1.

The Table 6.5 summarizes all the average results obtained during the experiments done with ALL-IDB2.

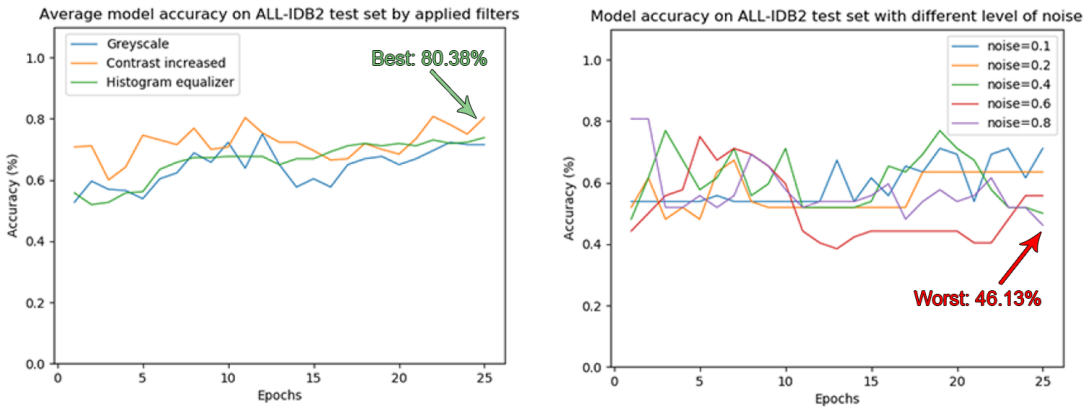


Figure 6.8: Accuracy evolution on test set of the best and the worst result for ALL-IDB2. The worst result is the experiment with noise=0.8 (purple line). The best result is the experiment using constrast increased filter (orange line).

	ACC-TEST	LOSS-TEST	ACC-TRAIN
Exp B.1 (baseline)	61.91%	1.86	100%
Exp B.2 (grey)	71.53%	1.09	99.9%
Exp B.2 (contrast)	80.38%	0.47	100%
Exp B.2 (hist. eq.)	73.84%	0.95	100%
Exp B.3 (200x200)	66.92%	0.804	99.902%
Exp B.4 (noise=0.1)	71.15%	1.41	100%
Exp B.4 (noise=0.2)	63.46%	2.34	99.5%
Exp B.4 (noise=0.4)	50.00%	3.60	98.55%
Exp B.4 (noise=0.6)	59.09%	4.33	97.55%
Exp B.4 (noise=0.8)	46.13%	7.12	94.71%
Exp B.5 (Dropout)	55.37%	1.9	68.12%
Exp B.6 (Data aug)	71.91%	1.198	99.9%
Exp B.7 (Dropout & Data aug)	64.61%	0.97	65.24%

Table 6.5: Summary of the results obtained for ALL-IDB2. The results presented in this table are the average results for each experiment. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set. The result highlighted in green is the best result obtained. The result highlighted in red is the worst result obtained.

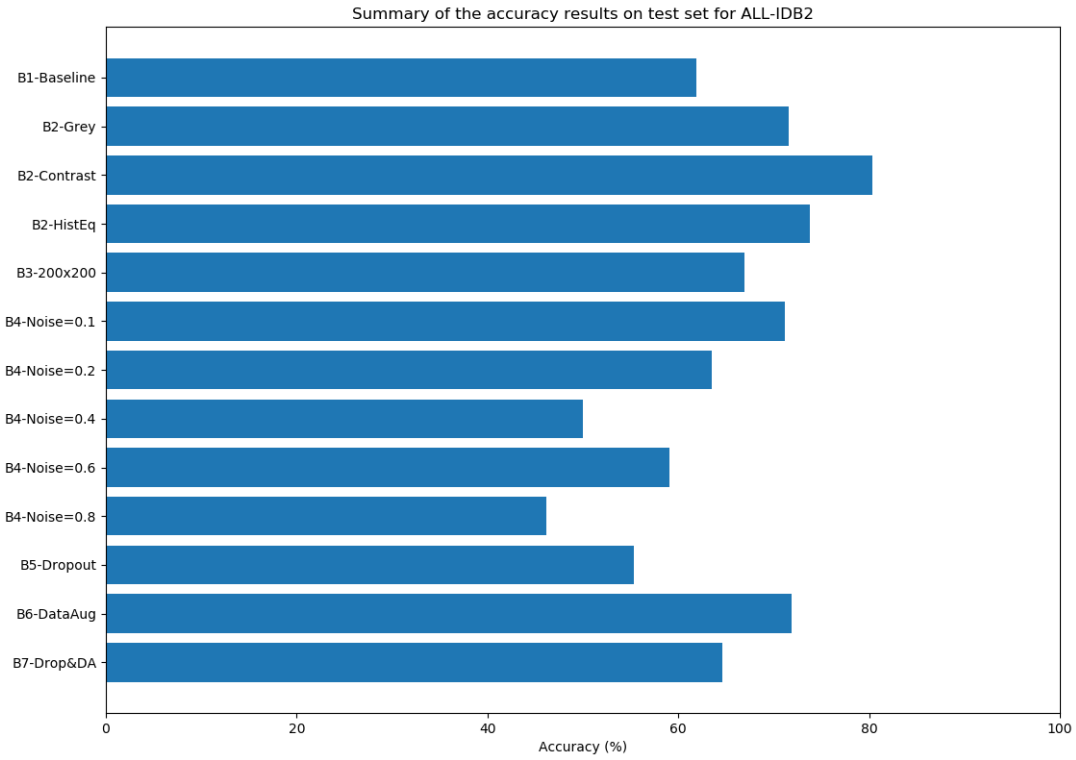


Figure 6.9: Summary graph of the accuracy results obtained on test set for ALL-IDB2.

6.7.3 Case C - Results summary for ALL-IDB and DEMIR-LEUKEMIA datasets experiments

⚠**The detailed results of the experiments for ALL-IDB and DEMIR-LEUKEMIA datasets are available in the appendix section A.3.** It contains detailed graphics, table and description of the results of each experiment done on **ALL-IDB and DEMIR LEUKEMIA datasets**. Each parameter is approached one by one and interpreted concerning their efficiency and the impact on the model's accuracy. Therefore, note that this current section is a **summary** of all these interpretations on obtained results for the datasets ALL-IDB1 and DEMIR-LEUKEMIA.

The Table 6.6 summarizes the results obtained for the identification of leukemia subtypes experiments. This time, the results obtained by using data augmentation get the highest performance which confirms the efficiency of data augmentation technique. Nevertheless, the experiments using image enhancement are also efficient whether by adding filters (greyscale, contrast increase or histogram equalization) or by increasing the size of the images. Following the three tackled cases, increasing the sample's size and using filters should then be validated methods to increase the performance of the model. One more time, it is the experiment C.4 where we add a large noise value that shows the worst performance. Remind that in this case, an accuracy of 20% signifies the randomness since there are 5 classes. This shows that the performance is worst when adding that large noise value than a model that picks a class randomly. Moreover, it seems that the dataset is still not big enough for dropout to be an efficient technique.

Thus, overall there is no real difference that we can observe from the experiments done on ALL-IDB (binary classification) and the identification of leukemia subtypes (5 classes). In both cases, data augmentation and image enhancement are the techniques that show the best performances.

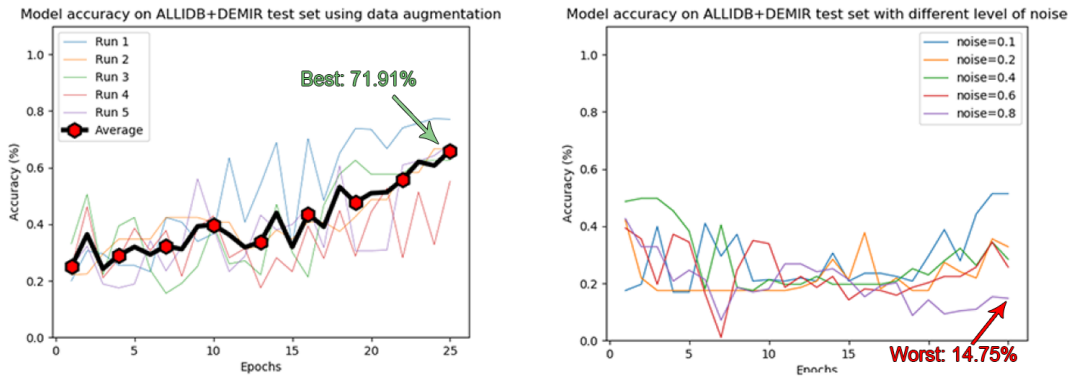


Figure 6.10: Accuracy evolution on test set of the best and the worst result for ALL-IDB and DEMIR-LEUKEMIA. The worst result is the experiment with noise=0.8 (purple line). The best result is the experiment using data augmentation (**black** line).

	ACC-TEST	LOSS-TEST	ACC-TRAIN
Exp C.1 (baseline)	49.61%	2.93	99.69%
Exp C.2 (grey)	59.01%	2.29	99.86%
Exp C.2 (contrast)	58.68%	2.05	99.72%
Exp C.2 (hist. eq.)	61.74%	1.63	99.72%
Exp C.3 (200x200)	59.99%	1.29	99.69%
Exp C.4 (noise=0.1)	51.36%	2.83	99.59%
Exp C.4 (noise=0.2)	32.78%	4.23	98.22%
Exp C.4 (noise=0.4)	28.41%	4.92	97.81%
Exp C.4 (noise=0.6)	25.68%	5.64	96.17%
Exp C.4 (noise=0.8)	14.75%	8.80	94.67%
Exp C.5 (Dropout)	25.89%	3.74	49.58%
Exp C.6 (Data aug)	71.91%	1.556	97.2%
Exp C.7 (Dropout & Data aug)	31.41%	1.874	53.79%

Table 6.6: Summary of the results obtained for ALL-IDB + DEMIR-LEUKEMIA datasets. The results presented in this table are the average results for each experiment. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set. The result highlighted in green is the best result obtained. The result highlighted in red is the worst result obtained.

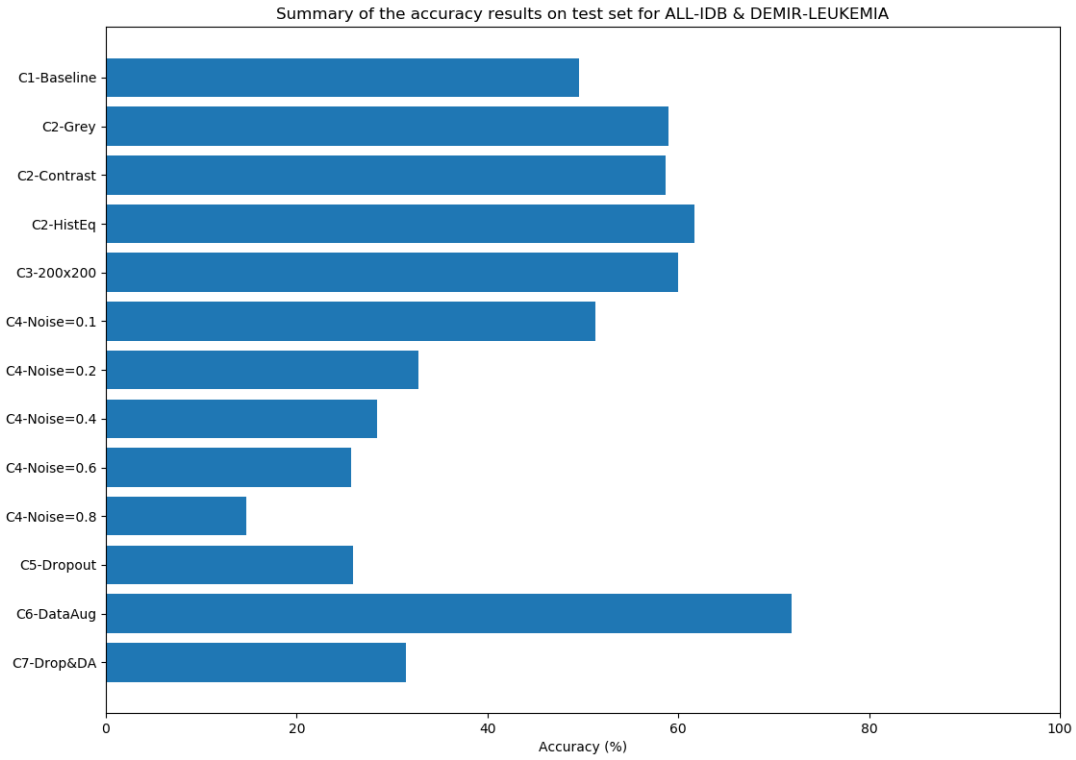


Figure 6.11: Summary graph of the accuracy results obtained on test set for ALL-IDB and DEMIR-LEUKEMIA.

6.8 Conclusion

To conclude the study of identification of leukemia subtypes from the multiple experiments done, we can highlight the **importance of having a good dataset**. In our experiments, the datasets that we have used were generally too small and did not contain enough samples to allow the model to learn perfectly. In addition to this, we have resized the images to a very small size (100x100) to allow a classic laptop to run the model on it. That resize is one more proof that the work on the dataset constitutes a huge percentage of the model's learning. Moreover, some images of DEMIR-LEUKEMIA datasets were affected by bad quality images with added features to the original images (watermarks, bad resolution, etc). Thus, before using the CNN to learn, it is important to clean the data to have a readable and reliable dataset to use.

Nevertheless, even if our datasets were not perfectly cleaned and formatted, we have obtained some convincing and motivating results referring to the results obtained by the authors of the article [29]. Overall, the experiments that performed best were obtained by using image enhancement and data augmentation. These techniques were used to highlight the edges and important features of the images or even to increase the number of samples of the dataset. These experiments were one more time some processing on the dataset and not on the model. It reinforces again the assertions on the importance of having a good dataset made previously. In contrary to this, the worst performance were obtained when using dropout on the CNN model. But again, it was due to the dataset that were not conform to the use of dropout regularization and not due to the technique itself (for example, we have seen that dropout was one of the best regularization technique on CIFAR-10 dataset constituted of 60000 images).

Finally, what we can conclude from all these experiments is that the architecture of the CNN constitutes in the end a low impact on the total performance. Of course, it is still important to have an architecture that suits well with the type of dataset but the impact of the structure and the content of the dataset dominates and constitutes a huge percentage of the factors that make the model performing good.

However, how can we enhance these results and make it more efficient? Is there further methods that can make it more reliable? The next chapters discuss about that.

Chapter 7

Discussion: Interactive Image Labelling and Active Learning

The results previously obtained may be convincing, though the objective is to improve again and again. Nowadays in the medical field, image recognition is mostly used as computer-aided detection in addition to an expert point-of-view. Nevertheless, the final goal is that the computer detection can be used independently and actually it constitutes a big challenge. The idea is that we use the professional expertise to enhance the model abilities to recognize images. We then introduce the concept of *Interactive Image Labelling* and its relative method called *Active Learning*. In this chapter, we will then first define those concepts and its relative areas. We will also discuss about how it works and the methodology of active learning. Finally, we will describe how it actually performs and what are the limitations of such techniques.

7.1 What is interactive labelling?

In the image classification field, we know that the the tasks of image labelling constitutes the basis of the machine learning model as it performs with supervised learning. However, we need this labelled data and it can become a real difficulty to obtain it (and sufficiently). Interactive labelling and active learning allows us to use unlabelled data combined to labelled data. This type of learning is called *semi-supervised learning*. This combination of data is basically used to reinforce the learning of the model via the interaction with the experts. Active learning is then fully-related to *Reinforcement Learning*. In this section, we will then describe the concepts of active learning, semi-supervised learning and reinforcement learning.

7.1.1 Active learning definition

Basically, Active Learning is a machine learning subfield where the learning method can **interactively** query an *oracle* which can be any type of information source (human or machine) to annotate new data that are not labelled. More formally, following *Settles* [42]:

Definition 7.1.1. *Active learning* (sometimes called “query learning” or “optimal experimental design” in the statistics literature) is a subfield of machine learning and, more generally, artificial intelligence. The key hypothesis is that if the learning algorithm is allowed to **choose the data** from which it learns, it will perform better with less training. [42]

The efficiency of active learning can then be explained by the fact that it uses less labelled training data (that are difficult to gather in large amount) and is compensated by the oracle information. This can be easily illustrated by an example. Let's take the breast cancer case that we have tackled in chapter 4. When studying this case, the training set was composed of multiple types of images from different sort of patients. By filtering the data thanks to the oracle information – which can be for instance the age or the general health state of the patient – we can select patients based on certain criteria and focus the training data on which the model has difficulty to learn. For example, if the model has difficulty to learn and predict the disease on old patients then we will focus on giving more old patients samples to the model so that he can improves its learning of this case. From the information of the oracle about the age, we can then improve the model.

The process of active learning is illustrated by the Figure 7.1. This process is called the pool-based active learning cycle. We start from labelled data to initially train the machine learning model then the oracle (also called annotator) select queries/criteria (step explained subsection 7.2.2) on unlabelled data and provides it to the labelled data to allows the model to learn on the new labelled dataset.

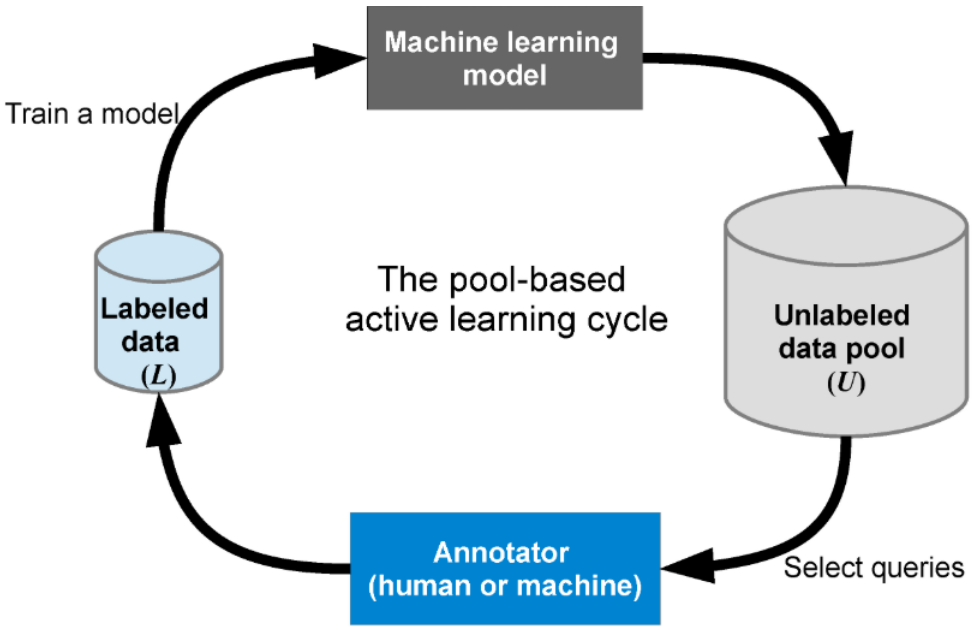


Figure 7.1: The pool-based active learning cycle. Figure extracted from [53]

7.1.2 Semi-supervised learning

Following the definition of *Chapelle et al.* [12],

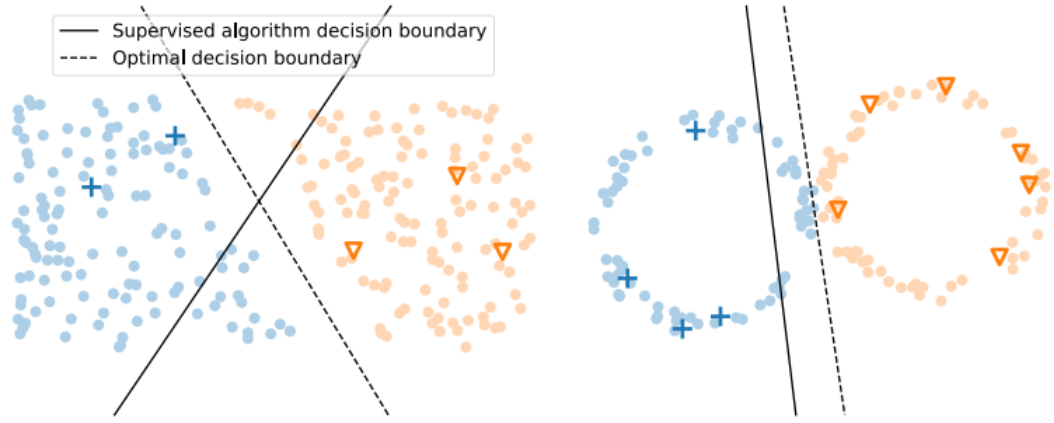
Definition 7.1.2. Semi-supervised learning (SSL) is halfway between supervised and unsupervised learning. In addition to unlabeled data, the algorithm is provided with some supervision information – but not necessarily for all examples [12].

As we use labelled and unlabelled data, active learning is then in a semi-supervised learning case. Semi-supervised learning requires an external information (i.e. the oracle) in addition to the labelled data. Active learning and semi-supervised learning techniques are then quite

similar and we may even consider active learning as a special case of semi-supervised learning. Nevertheless, some differences can be considered between both cases. While semi-supervised methods exploit what the learner thinks it knows about the unlabelled data, active methods attempt to explore the unknown aspects [42]. It means that the objective of active learning is to try to find out labelling which small number of them will get much better performance (e.g. the age criteria on our breast cancer detection example) while general semi-supervised learning techniques focus on the knowledge of the remaining unlabelled data to enhance the performance.

But why does semi-supervised learning techniques work? Following the survey of *van Engelen et al.* [48], there exists three assumptions of semi-supervised learning (and then active learning) that can proves the efficiency of such techniques. These assumptions are illustrated in Figure 7.2. We consider that a point in the 2D-area is representing an image for the image classification problem.

- **Smoothness assumption:** for two input points $x, x' \in X$ that are close by in the input space, the corresponding labels y, y' should be the same. [48] That means that unlabelled points near a labelled point should have the same label as the labelled point.
- **Low-density assumption:** (can be considered as the counterpart of the smoothness assumption) the decision boundary of a classifier should preferably pass through low-density regions in the input space [48].
- **Manifold assumption:** first, manifold are topological spaces that are locally Euclidean. This assumption states that the data lie approximately on a manifold of much lower dimension than the input space [51]. For instance, when we consider a 3-dimensional input space where all points lie on the surface of a sphere, the data can be said to lie on a 2-dimensional manifold [48].



(a) Smoothness and low-density assumptions. **(b)** Manifold assumption.

Figure 7.2: Illustration of semi-supervised learning assumptions. (a) Smoothness and low-density assumptions (b) Manifold assumption. The filled line represents the classification of a supervised learning algorithm while the dotted line represents the classification of a optimal decision boundary that approximately represents a semi-supervised learning algorithm. The + represents labelled data of blue class while ▽ represents labelled data of orange class. Circle symbols represents unlabelled data from respective classes regarding its color. Figure extracted from [48].

7.1.3 Reinforcement Learning

In active learning, the model interacts with an oracle to reinforce its decisions. That sounds a lot like *Reinforcement Learning* where the learner (i.e. agents) interacts with the world via "actions" and tries to find an optimal policy of behaviour with respect to "rewards" it receives from the environment. The objective in active learning is somehow to gain "rewards" via the interaction with the oracle to learn based on the unknown criteria to gain more information and then make better decisions and predictions. In parallel, the reinforcement learner take risks and try out actions for which it is uncertain about the outcome (corresponding to the active learning that requests labels on the unknown). This process is illustrated in Figure 7.3. This method is called "exploration-exploitation" and is applied by multiple reinforcement learning algorithms such as Q-learning or Monte Carlo. Reinforcement learning and active learning are then related as the learner **must be proactive** in order to perform well.

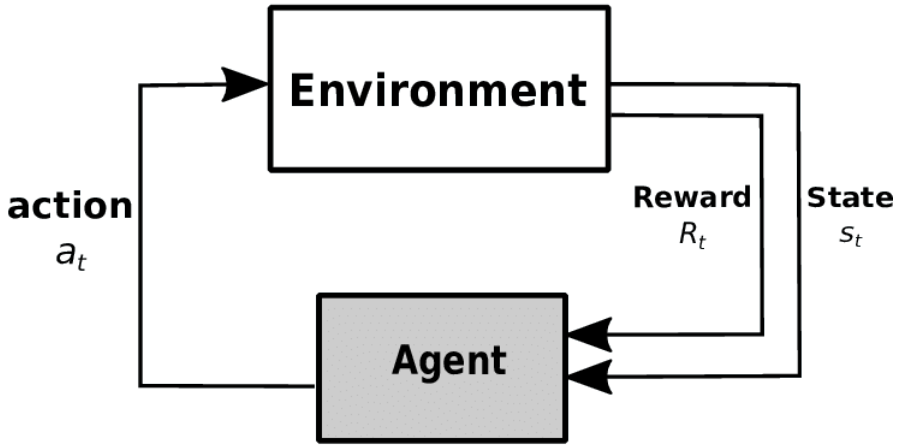


Figure 7.3: Reinforcement learning cycle. Figure extracted from [4]

7.2 Methodology of Active Learning

In this section, we present the methodology of active learning and how does it work. More precisely, the scenarios where active learning acts are covered as well as the different query strategies.

7.2.1 Active learning scenarios

Three typical scenarios are known in active learning in which the learner will query the labels of instances. These three scenarios are called "Membership Query Synthesis", "Stream-Based Selective Sampling" and "Pool-Based sampling". Each scenario will be illustrated with an image classification problem example.

7.2.1.1 Membership Query Synthesis

In this scenario, to make a query about the label of an unlabelled data sample (i.e. an image), the learner will generate an instance of this data sample and send it to the oracle so that he can label it. This is basically the most simple scenario where the learner asks directly the label of

an unlabelled data to the oracle. For example, if the problem treats about the CIFAR10 dataset presented in section 5.1 and the learner has an image of car that is unlabelled, the model will generate a similar image (a generalization of this car image) that will be sent to the oracle in order to label it. So that, the model has henceforth the label for such images. The problem that can be encountered in such scenario is that the model cannot generate well a similar image and then the query images generated by the learner are not recognizable for the oracle. This problem is due to the efficiency and capabilities of the model to first recognize the features of the image and in a second step to generate them.

7.2.1.2 Stream-Based Selective Sampling

From the instance selected **one by one** from the unlabelled data pool, the learner here determines whether he wants to query the label of the instance or not regarding the information that he knows about this. Indeed, the main purpose of active learning is to focus on the unknown to learn more about it in order to have a more reliable model on each case. Thus, if the model has already a good knowledge and has enough learned about the instance's data, the decision of not querying the label to the oracle is more relevant and optimized. The informativeness of a query is determined by **query strategies** tackled in subsection 7.2.2. Following the same example than the previous scenario, if the pool of unlabelled data contains car and airplanes images, knowing that the model has enough information about airplanes, the model will discard the instance when the model encounters an instance of airplane image and then will make queries when the instance is covering an unlabelled car image.

7.2.1.3 Pool-Based sampling

Similarly to the Stream-Based Selective Sampling scenario, we assume that we have a large pool of unlabelled data. Here we make a ranking on **all** the instances of the pool according to the informativeness of each instance and we simply select the instances that are best ranked. In contrary to Stream-Based Selective Sampling scenario, the main difference lays in the fact that pool-based sampling evaluates and ranks the entire collection before selecting the best whereas the former scans through the data sequentially in and makes query decisions individually [42] in the previous scenario. Pool-Based sampling seems then to be the most common scenario but Stream-Based selective sampling can be a good alternative in some situations where memory or processing are limited. Discussing about the same examples then previous scenarios, if images with cars/airplanes/birds appear in the top of the informativeness ranking, we will then pick the related images and make label queries on it.

The Figure 7.4 summarizes the three scenarios discussed above.

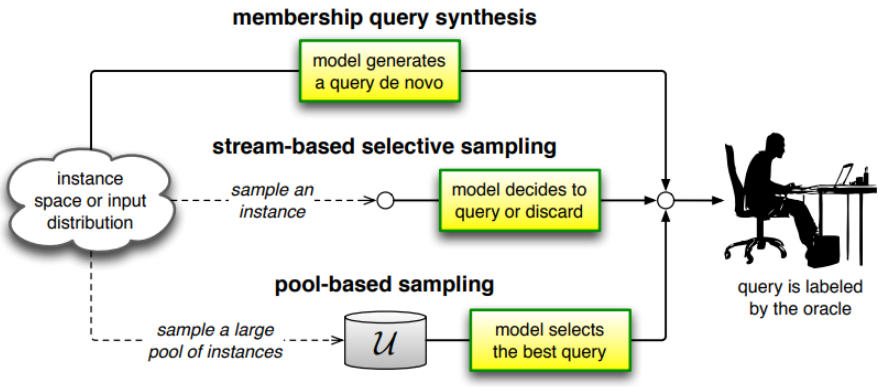


Figure 7.4: Diagram illustrating the three main active learning scenarios. Figure extracted from [42]

7.2.2 Query Strategies

As discussed in the scenarios, the learner needs to determine the relevance of making queries to the oracle. This is based on the informativeness given by querying the label such that we pick the most informative queries to label. To measure this informativeness and to determine whether or not it is relevant to query the unlabelled data, we need to use query strategies. Multiple query strategies exist and most of them are covered by the works of *Settle* [42] or *Akusok et al.* [1]. In the context of this section, some popular query strategies are presented below.

7.2.2.1 Uncertainty Sampling

In this strategy, the measure of the information rate of the query is based on the uncertainty of the data. As we want to learn more on unknown data, this approach allows to make label queries on the least uncertain instances. The uncertainty measure used is the **entropy** which is the most popular measure in information theory developed by *Shannon*. Entropy is defined by

$$E(S) = -\sum_{j=1}^{k} p_j log_2(p_j)$$

where S is the source that produces k messages (i.e. labels), p_j is the probability that that the j^{th} element appears in the dataset.

The uncertainty sampling strategy can also consider the **least confident** where the learner selects the instance for which it has the least confidence in its most likely label. The least confident instance is defined by the class label with the highest posterior probability[1] under the model.

However, using the least confident strategy may be too poor since it only takes in consideration the most probable label and disregards the other label probabilities. To overcome this problem, the uncertainty sampling strategy can also be based on **margin** that selects the instance that has the smallest difference between the first (i.e. the one found by the least confident strategy) and second most probable labels.

[1] Following *Wikipedia* [51], the posterior probability of a random event or an uncertain proposition is the conditional probability that is assigned after the relevant evidence or background is taken into account.

7.2.2.2 Query-by-committee

The Query-by-committee strategy is aptly named since it involves training multiple models (called the committee), and selecting samples that this committee disagree about the most. The committee is trained on the same current labelled set but chooses different hypothesis where each member of the committee votes and disagrees for these hypothesis and the relevance on the labellings of query candidates. The most informative query is considered to be the instance about which they most disagree. The Query-by-committee strategy is known to have less computation than other query strategies.

Moreover, to create the committee, we can also use *query-by-boosting* and *query-by-bagging* which use ensemble learning methods "boosting" and "bagging" in order to have diversity among committee members. Indeed, the strategy is most efficient when the models are varied and diversified to have different hypothesis. To make it more clear, an analogy about a survey about "the frequency of sport performed by people" can be done where we obtain more relevant and reliable results when we collect the opinions of different types of person (young, old, ...). Otherwise, if we only collect information with young people (similar models), the results may be biased and are relatively high while if we collect results on young and old people (diversified models) the results should be more accurate and representative.

7.2.2.3 Expected model change

This strategy is based on the changes that would impact the model if the label of the query is found. Then this is the instance in which the label found would lead to the greatest positive change in the model that is chosen to query. This strategy has a huge computational cost since you need to uses methods to compute the gradients for a loss function that is averaged over all label possibilities for the given instance [31]. The cost of using this strategy is then very high since you need to try every combinations of label to the instance to see its expected impact on the model.

Similarly to it, it exists an alternative strategy that observe the model's generalization error instead of the model's changes. That strategy is called *Expected error reduction*. To decide the instance to query, we take the one that reduce the most the generalization error of the model. It uses the same principle than expected model change strategy by computing the change in the validation error averaged over the class labels. It has even more computational cost and then is not recommended to use.

7.3 Performance and limitations of Active Learning

7.3.1 Performances of AL: hands-on experiments

To evaluate the performance of the active learning technique, multiple researchers did experiments and have written papers on it. In this section, we will be based on a paper written by *Guo-Jun Qi et al.* named *"Two-Dimensional Active Learning for Image Classification"* [34].

In this paper, the authors consider a variant of the traditional active learning method called the *two-dimensional active learning*[2]. This method considers the sample dimension but also the

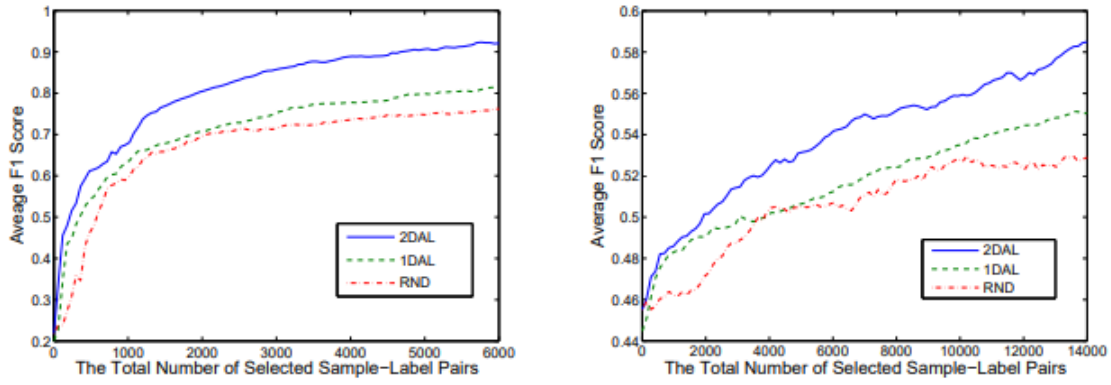
[2] For more information and details about the 2D active learning method, take a look on the section 2 of the main paper [34].

label dimension in particular to compute uncertainty. This new AL method as well as the traditional AL (1DAL) will be performed on image classification problems to compare them with a classical supervised learning strategy (random selection), more precisely on natural scene dataset (with beach, sunset, foliage, fields, ... images) and gene dataset (with genes images belonging to 14 different functional classes). Concerning the query strategies, the authors use the *maximum entropy model* and the *expected model change* variant, both explained in subsection 7.2.2.

The results obtained using active learning methods on the described datasets are illustrated by the Figure 7.5. The left graph is the performance on the natural scene dataset and the right graph is the performance on the genes dataset.

We can observe that the behaviour of the curves are quite the same for both datasets and show the efficiency of active learning methods (blue and green lines) compared to a supervised learning method with random selection. In average, the F1-score for the first dataset does not exceed 0.7 using random strategy while it can reach over 0.8 using simple active learning (1DAL) and even 0.92 using the improved AL strategy submitted by the authors of the paper. For the second dataset, the results are less significant but show the same trends.

Thus, what the results prove is that the **selection of samples to label** has an important weight on the performance of the model by enhancing it as expected. Moreover, the authors of this paper provide added value on the traditional active learning method by also considering the label (and not only the sample) to compute the uncertainty which allows us to **query the most reliable samples**. That points out that multiple variant of active learning exist which demonstrates the wide range of the active learning technique which can still be improved.



(a) *Scene* (b) *Yeast*

Figure 7.5: Performance of active learning methods on image classification problem. The y-axis corresponds to the F1-score and the x-axis to the number of selected sample pairs. 2DAL (blue line) = two dimensional AL, 1DAL (green pointed line) = one-dimensional AL, RND (red pointed line) = random strategy. (a) left: natural scene dataset ; (b) right: gene dataset. Figure extracted from [34]

7.3.2 Summary and conclusion: what are the limitations of active learning?

7.3.2.1 Performance

In the last chapter approaching the identification of Leukemia Subtypes, what we had conclude is the importance of having a clean and reliable dataset first in order to correctly perform our model on it. Thus, we have introduced the concept of active learning which is a technique that allow us to improve the labelling of our dataset samples. The experiments done by the authors of the paper [34] [3] shows motivating results concerning active learning techniques to procure better labelled datasets and then reinforces the idea of the importance of the dataset. All that confirms that **data labelling** is the central core of a good model performance for an image classification problem. In this situation, we are then not talking about a problem of labelling speed but more about the intelligence of labelling. Of course it goes hand in hand: the smarter the labelling, the faster it will be done.

The Figure 7.6 from *Jennifer Prendki* [33] shows the expected efficiency of active learning on the model's performance. When comparing the results obtained by *Guo-Jun Qi et al.* [34] on image classification problems[3], we can observe the same curves' behaviour when using active learning against using the supervised learning.

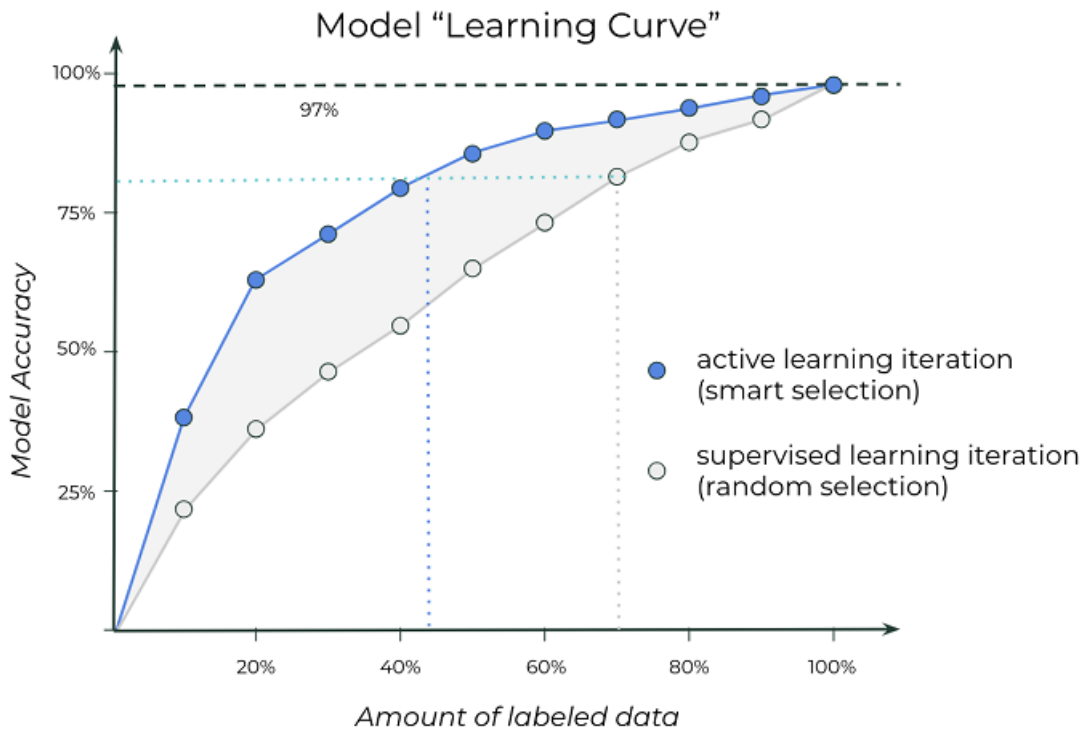


Figure 7.6: Expected results using active learning compared to supervised learning. With Active Learning, a given accuracy can be reached with significantly less data; in this example, a 80% accuracy can be achieved with only 45% of the total volume of available data as opposed to 70% in the case of regular supervised learning. Figure extracted from [33]

[3] cfr. subsection 7.3.1

7.3.2.2 Limitations

Active learning seems flawless but in reality this is a concept that is not totally understood. Previously we have seen that there exists multiple variants of active learning such as Two-Dimensional Active Learning [34] which means that other variants may exists and probably not even discovered.

Moreover, we have related active learning to semi-supervised learning which is a category of algorithms that is **vulnerable to biases** since it is based on assumptions which are in fact not always verified[4].

Last but not least, active learning requires the intervention of an annotator to label the data which is **expensive**.

Nevertheless, it is important to remind that results that have been obtained by multiple researchers are very encouraging for the future of active learning technology.

[4] cfr. subsection 7.1.2

Chapter 8

Conclusion and further work

Image classification, artificial intelligence, machine learning, convolutional neural networks, performance, ... All these terms that have been approached and have constituted the keys of the thesis. To conclude the document, let's make an overview of what has been done as well as what could be done for a future work in the domain of image classification whether on classical data or on medical data.

8.1 Thesis summary

8.1.1 Image classification problem summary [1]

The thesis approaches the problem of image classification also called "image identification" or "image recognition". Identifying the content of an image automatically is nowadays a very challenging problem and concerns all fields in the world. This image classification problem takes on greater importance when it is linked to the medical domain by, for instance, detecting cancer tumours from microscopic images. In this case, the human health comes into play and makes the business more serious. Nevertheless, for the medical field, automated image recognition technology is only used as computer-aided detection to the experts for the time being. Of course, the final objective is that the computer has the capability to be autonomous and reaches an accuracy of 100%. Thus, identify an image's content requires an accurate methodology and each optimization that can be done to identify better or/and faster an image is welcome.

The image recognition methodology is described by machine learning and deep learning methods such as neural networks, in particular convolutional neural networks. We have observed that this kind of model cannot work alone and also needs a huge preprocessing work on images such as image enhancement or segmentation to extract interesting features. Thus, to do this pre-processing steps, we need to acquire conform datasets with enough data to train our model. All this proves the complexity of the image classification problem because all these elements need to be taken in account and cannot be ignored to have reliable results.

[1] Corresponds to the summary of chapter 1 (Introduction) to chapter 4 (Breast cancer detection) included.

8.1.2 Contribution summary [2]

How to optimise a CNN model? How to get the best results? These are the main purposes of the thesis. We have discussed about the parameters that could improve the accuracy of a CNN learning model first by experimenting our model on classical images dataset (CIFAR-10). The model that we have used was built from the simplest CNN structure so that changes made to the parameters are significantly observable. Before using our model, the dataset has been preprocessed using sampling and one hot encoding. Sample preprocessing step allowed us to observe how the results differ when using smaller dataset. Indeed, for the number of classes considered, using only one batch reduces the model's accuracy by an average of 10%. The impact of sampling is then notable. Afterwards, CNN model's parameters that we have taken in account were the epochs number, the optimizer, dropout values, weight decay values, the use of data augmentation and the presence of Gaussian noise. Data augmentation and dropout have demonstrated the best results.

Subsequently, we have made experiences on medical images in particular on microscopic images of bone marrows and other blood-forming organs to detect abnormal blood cells in order to classify different leukemia subtypes (ALL, AML, CLL, CML). The parameters that we have considered this time are filters (contrast enhancement, histogram equalizer, greyscale), image's size, presence of noise, dropout and use of data augmentation. Same as CIFAR-10 dataset, data augmentation has shown the better result but this time filters has also proved its effectiveness by highlighting edges of tumours. Moreover, we have denoted the importance of the dataset structure and content which probably constitutes the most important step to obtain reliable results. That proves the significance of preprocessing steps in addition to a good CNN architecture.

8.1.3 Limitations summary [3]

Using exclusively the CNN model reaches limitations very quickly. The purpose of machine learning method on medical image recognition problem is to be pulled away from its status of "computer-aided detection". Thus, we have approached a new method called "Active Learning" that can enhance the interactive image labelling capacity of a CNN model. This method gathers the informations obtained by the model and the informations of an oracle (most of the time, a human expert) in order to improve the model by querying to the oracle new informations that could help on labelling new data. Active learning then uses labelled and unlabelled data for the purpose of reinforce the model's knowledge. That can make you think about semi-supervised learning and reinforcement learning. We have defined the most common strategies and query methods. Some papers have worked on it and have proven the efficiency of such a method compared to a classical supervised learning method by convincing results. Nevertheless, active learning remains very fuzzy by its vulnerability to biases and its expensiveness.

8.2 Further work: neural networks extension

8.2.1 Multiple inputs and mixed data

The content of the thesis focuses only on the image variable to make predictions on a problem but we have to note that **the image is one variable among many others**. For instance, if

[2] Corresponds to the summary of chapter 5 (Model prototype) and chapter 6 (Leukemia Subtypes identification).
[3] Corresponds to the summary of chapter 7 (Active Learning).

we come back to the breast cancer detection problem, in the chapter 4 we have only taken in account the mammograms images ignoring others variables that can influence and may have a huge impact on the prediction result (in this case, if the patient has breast cancer or not). These variables can be the age, the global health, the height, the weight, gender, ... of the patient which are in particular **numerical and categorical data**. Thus, in addition to images data, we can add informations by using these data. The objective is to keep the efficiency of the CNN architecture used for images and extend it by adding this new information from non-image data.

When giving multiple types of independent data to our machine learning model, we call them **mixed data**. In the previous example of breast cancer detection, the mixed data is composed by numerical data (age, height, weight,...), categorical data (gender) and of course image data (mammogram radiography images). Non-image data may be used such as time series or texts data. For example, the problem of emotion recognition involves multiple types of data: facial expressions (visual), the voice (audio), or the meaning of words (text). The 2 first can be translated into images and managed by CNN but the meaning of words needs to be processed by techniques such as Natural Language Processing (NLP).

Using mixed data is then obviously challenging since it is needed to use more preprocessing steps for non-image data such as normalization or scaling in addition to those done for images. Mixed data is simply illustrated in the Figure 8.1.

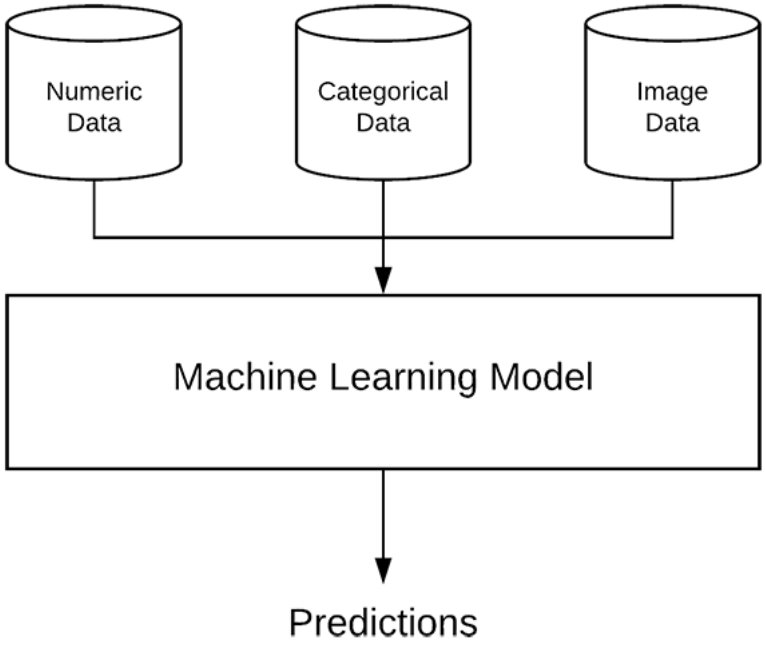


Figure 8.1: Scheme of mixed data given to machine learning model to enhance predictions. Figure extracted from [35]

8.2.2 How and why would it work?

To use mixed data, we need to change the architecture of the machine learning model. Overall, the idea is to use two layers to manage on one side image data and on the other non-image data which creates multiple branches in the model that merges to produce one (or multiple)

output(s). Thus, data must be loaded and preprocessed in order to be applied to the multi-input network.

We have seen that CNN is optimized for images and changing the CNN architecture directly would have been a loss of time and performance. The idea is then to use multiple models: one for image data which will be obviously a CNN as well as one for non-image data. Concerning this last, a multi-layer perceptron (MLP) is usually used when using numerical data but can differs depending of data type (e.g. NLP model for texts). The model is then defined by two branches including a CNN and another network (e.g. MLP, NLP). Their respective outputs are fed into a common set of Dense layers and are then combined using for example concatenation to finally obtain a multi-inputs network model. The Figure 8.2 recapitulates and illustrates the scheme of the Machine Learning model architecture.

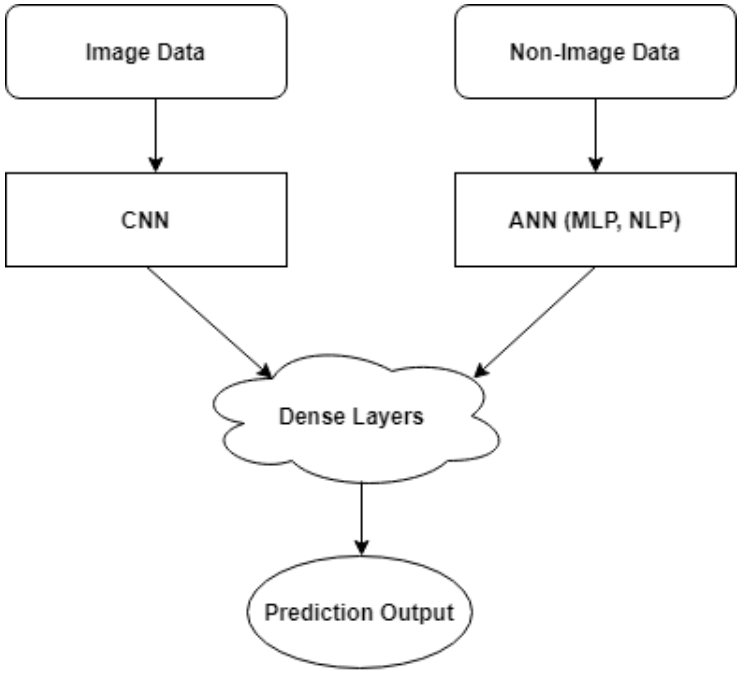


Figure 8.2: Scheme of a multi-inputs network model.

8.2.3 Further work purpose

We have noticed all along the thesis that the image classification problem concerns a large area of fields. From social to medical domain, we have focused on the image variable but we have seen the importance of other types of variables such as numerical or categorical data. However, combining multiple types of data is still a very touchy and in full expansion subject.

Nowadays, few papers and articles approach this while it can greatly improve the efficiency of machine learning techniques. For instance, *Yu Sun et al.* are the authors of one of the few articles on this subject called "Multi-Input Convolutional Neural Network for Flower Grading" [44] written recently in May 2020. This article approaches this extension of CNN using a Multi-Input CNN for flowers grading and shows the improvement of the prediction's accuracy that goes from 89% to 94% when using this Multi-Input CNN. Therefore to complete this thesis, further work on the combination of multiple inputs data by using multiple models may be a great improvement topic to discuss in particular for critical fields such as medical area.

Bibliography

[1] AKUSOK, A., EIROLA, E., MICHÉ, Y., GRITSENKO, A. et LENDASSE, A. (2017). Advanced query strategies for active learning with extreme learning machines. *In 25th European Symposium on Artificial Neural Networks, ESANN 2017, Bruges, Belgium, April 26-28, 2017*.

[2] ALEX KRIZHEVSKY, Ilya Sutskever, G. H. (2012). Imagenet classification with deep convolutional neural networks. *University of Toronto*.

[3] ALEX KRIZHEVSKY, V. N. et HINTON, G. (2009). The cifar-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html.

[4] AMIRI, R., MEHRPOUYAN, H., FRIDMAN, L., MALLIK, R. K., NALLANATHAN, A. et MATOLAK, D. (2018). A machine learning approach for power allocation in hetnets considering qos. *In 2018 IEEE International Conference on Communications, ICC 2018, Kansas City, MO, USA, May 20-24, 2018*, pages 1–7. IEEE.

[5] ANGEL, A. (2015). Color histogram equalization. https://www.imageeprocessing.com/2015/11/color-histogram-equalization-matlab-code.html.

[6] 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/.

[7] 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.

[8] BOSER, B. E., GUYON, I. et VAPNIK, V. (1992). A training algorithm for optimal margin classifiers. *In* HAUSSLER, D., éditeur : *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992, Pittsburgh, PA, USA, July 27-29, 1992*, pages 144–152. ACM.

[9] 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/.

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

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

[12] CHAPELLE, O., SCHÖLKOPF, B. et ZIEN, A., éditeurs (2006). *Semi-Supervised Learning*. The MIT Press.

[13] CHOLLET, F. (2015). Keras documentation. https://keras.io/.

[14] COLE, M. (2017). Building a facial recognition pipeline with deep learning in tensorflow. https://hackernoon.com/building-a-facial-recognition-pipeline-with-deep-learning-in-tensorflow-66e7645015b8.

[15] DINA A. RAGAB, Maha Sharkas, S. M. et REN, J. (2019). Breast cancer detection using deep convolutional neural networks and support vector machines. *PeerJ*.

[16] FOGARTY, T. (2018). Regression or classification? linear or logistic? https://towardsdatascience.com/regression-or-classification-linear-or-logistic-f093e8757b9c.

[17] 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*.

[18] GONZALEZ, R. C. et WOODS, R. E. (2008). *Digital Image Processing, 3rd edition*. Prentice Hall.

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

[20] GUY SHTAR, S. M. (2017). Clustering and dimensionality reduction: Understanding the "magic" behind machine learning. https://www.imperva.com/blog/clustering-and-dimensionality-reduction-understanding-the-magic-behind-machine-learning/.

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

[22] HANIEH, B. (2014). Artificial neural network part 1.

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

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

[25] JOHN PAUL MUELLER, L. M. (2016). *Machine Learning for Dummies*. Learning Made Easy.

[26] KRZYK, K. (2018). Coding deep learning for beginners — linear regression (part 2): Cost function. https://towardsdatascience.com/coding-deep-learning-for-beginners-linear-regression-part-2-cost-function-49545303d29f.

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

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

[29] NIZAR AHMED, Altug Yigit, Z. I. A. A. (2019). Identification of leukemia subtypes from microscopic images using convolutional neural network. *MDPI*.

[30] of HEMATOLOGY, T. A. S. (2020). Ash image bank. https://imagebank.hematology.org/.

[31] PETROVA, O. (2019). Active learning: The theory. https://blog.scaleway.com/2020/active-learning-some-datapoints-are-more-equal-than-others/.

[32] POATEK (2019). Data augmentation. https://www.poatek.com/2019/10/07/how-to-get-even-more-performance-from-transfer-learning/data-augmentation/.

[33] PRENDKI, J. (2018). Introduction to active learning. https://www.kdnuggets.com/2018/10/introduction-active-learning.html.

[34] QI, G., HUA, X., RUI, Y., TANG, J. et ZHANG, H. (2008). Two-dimensional active learning for image classification.

[35] ROSEBROCK, A. (2019). Multiple inputs and mixed data. https://www.pyimagesearch.com/2019/02/04/keras-multiple-inputs-and-mixed-data/.

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

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

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

[39] SAHA, S. (2018). A comprehensive guide to convolutional neural networks ? the eli5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[40] SCOTTI, F. (2020). Acute lymphoblastic leukemia image database for image processing. https://homes.di.unimi.it/scotti/all/.

[41] SEIF, G. (2018). Deep learning for image recognition: why it's challenging, where we've been, and what's next. https://towardsdatascience.com/deep-learning-for-image-classification-why-its-challenging-where-we-ve-been-and-what-s-next-93b56948fcef.

[42] SETTLES, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

[43] 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.

[44] SUN, Y., ZHU, L., WANG, G. et ZHAO, F. (2017). Multi-input convolutional neural network for flower grading. *J. Electr. Comput. Eng.*, 2017:9240407:1–9240407:8.

[45] 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*.

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

[47] 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.*

[48] van ENGELEN, J. E. et HOOS, H. H. (2020). A survey on semi-supervised learning. *Mach. Learn.*, 109(2):373–440.

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

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

[51] WIKIPEDIA (2019). Wikipedia, the free encyclopedia. https://en.wikipedia.org/.

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

[53] YANG, L., MACEACHREN, A. M., MITRA, P. et ONORATI, T. (2018). Visually-enabled active deep learning for (geo) text and image classification: A review. *ISPRS Int. J. Geo-Information*, 7(2):65.

[54] 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*.

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

Appendix A

Detailed results of Leukemia Subtypes identification experiments (chapter 5)

This appendix contains all the results of the chapter 5 experiments concerning the identification of Leukemia Subtypes. These results are structured by a detailed description and interpretation, the table containing the values results obtained and their corresponding graphs. In each result graph, the result of each run is drawn as well as the average of the 5 runs highlighted.

A.1 Case A - ALL-IDB1

For the first case, we use the ALL-IDB1 dataset which contains a total of 108 images. We then use 86 images for training and 22 images for testing when not using data augmentation. Otherwise, we have 172 images for training and 44 images for testing. As reminder, ALL-IDB1 dataset contains only 2 classes: HEALTHY (label 0) vs ILL (label 1) and contains a large view of blood cells of the patient. ALL-IDB dataset is explained in details in section 6.2.

A.1.1 Experiment A.1 - Baseline

This experiment consists of trying the simplest version of the CNN and observe how it reacts to the resized original dataset. To make a general idea about the efficiency of our model, we can refer to the results obtained by Ahmed et al on his work [29]. They obtained an average of 81.16% accuracy on test set, 1.3093 for loss value and 99.55% for train set without applying any image transformations.

Generally, the results obtained are not so far than the results obtained by the reference article [29] knowing that we have used images resized to 100x100 (= 10000 pixels). Thus, the images that we have used have **22 times less pixels** since the original images are sized of ≈ 1700x1300 (= 221000 pixels).

We can certainly assure that this resizing impacts the performance of our CNN model and can cause some troubles in its capacity to learn and retrieve the information. Nevertheless, despite this huge inconvenience, the CNN is still capable of learning some features otherwise we would have an average accuracy of 50% in this type of binary classification. We can observe that the CNN converges very quickly – no more than 5 epochs, to its convergence state.

The general behaviour when looking to the results graphs on Figure A.1 is that the accuracy tends to increase slowly and stabilize at a certain time. Same for the loss value, it tends to decrease along time.

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**
ACC-TEST	68.18%	63.63%	59.09%	63.63%	72.72%	65.45%
LOSS-TEST	3.53	5.86	6.55	0.55	0.82	3.46
ACC-TRAIN	100%	100%	100%	100%	100%	100%

Table A.1: Experiment A.1 (On ALL-IDB1, with Baseline) - Train set = 86 images, Test set = 22 images - Detailed results of each run after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

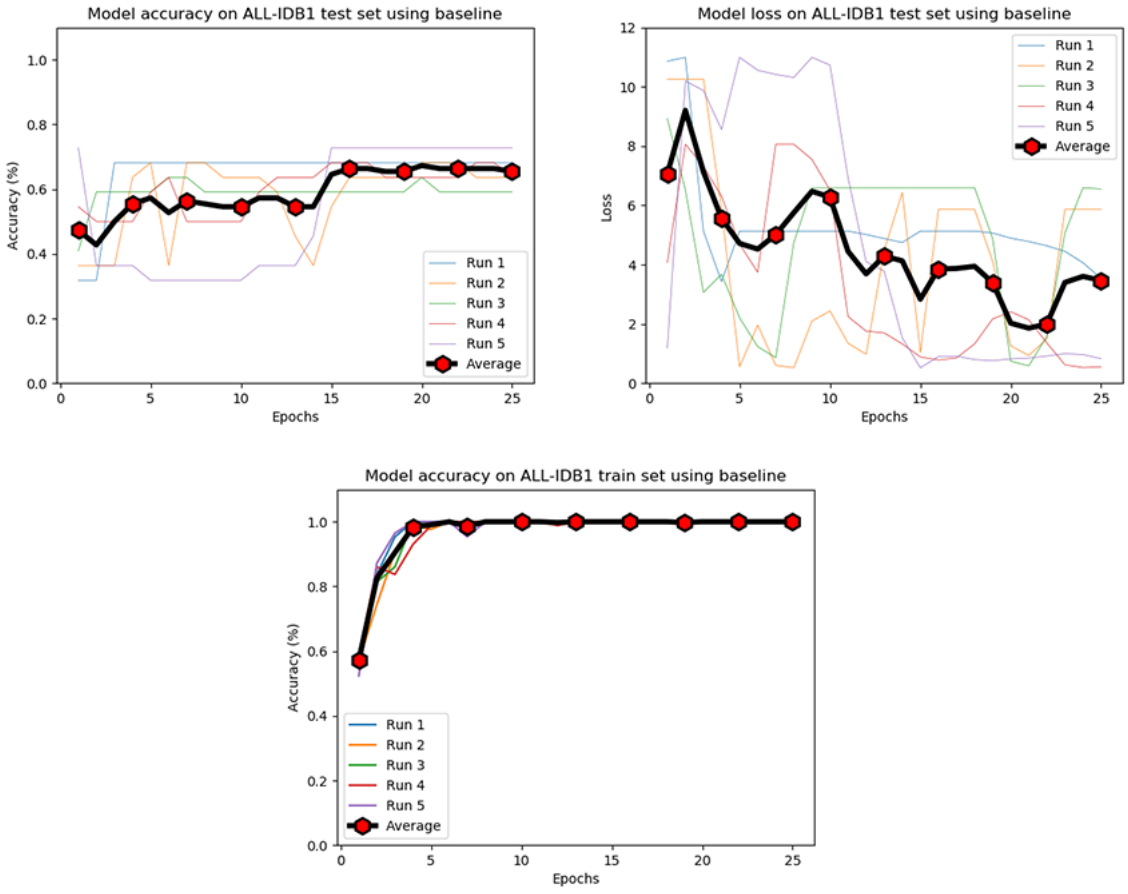


Figure A.1: Results of experiment A.1 (On ALL-IDB1, with Baseline) - The average result of the 5 runs after 25 epochs: (1) Accuracy on test set: **65.45%** (2) Loss value on test set: **3.46** (3) Accuracy on train set: **100%**

A.1.2 Experiment A.2 - Filters

In this experiment, we keep the baseline CNN but we preprocess the images of the resized dataset by applying some filters **separately**.

Greyscale

First filter that we apply is transforming the images into greyscale. The results obtained when using greyscale images is very bad. It stucks to 50% which means a totally random prediction. This issue can be due the fact that images are already very small and we suppress the color feature that seems to be important in such images. In fact, the original images (with color) of ALL-IDB1 dataset are at first sight very similar. One of the main feature that distinct them is color. Adding such greyscale filter transforms the images into "mirror" images that causes a total fuzziness whether for human eyes or the CNN model.

Contrast increased

The second filter we apply is to increase the contrast of the images by an arbitrary value. In our case, we will increase the contrast by 4. As illustrated in Figure 6.5(b), the color and the edges of blood cells are more highlighted. These features constitute important information for such classification. Bringing them to light can therefore only be a good idea. This affirmation is well confirmed by the results obtained. In average, we obtain an accuracy of 84.54% after 25 epochs which is even more than the baseline result of the reference article presented in the previous section. Hence, the size of the images seems to not be the most important feature in such images as we obtain in average better results by increasing the images' contrast to highlight the blood cells color and edges on 100x100 images. However, we will take a look on the size feature on subsection A.1.3.

Histogram equalization

Finally, the last filter that we apply is a well-known image enhancement technique called *histogram equalization* and previously explained in subsection 4.2.1. In summary, what histogram equalization do is increasing the contrast of an image by effectively spreading out the most frequent intensity values of the histogram. This technique is known to be efficient in image recognition problems. This allows for areas of lower local contrast to gain a higher contrast. The results obtained (average of 90.9%) do not betray the reputation for image enhancement of the histogram equalization technique. In contrary to the previous filter where we apply the same contrast increase on the entire image, here we only increase the contrast where it is needed. Therefore, we obtain clearer images that may highlight efficiently affected blood cells and help for the model's learning.

	Greyscale	**Increased contrast**	**Histogram equalization**
Accuracy on test set	51.81%	84.54%	90.9%
Loss value on test set	6.09	1.43	0.28
Accuracy on train set	100%	100%	100%

Table A.2: Experiment A.2 (On ALL-IDB1, with filters applied) - Train set = 86 images, Test set = 22 images - Average results for each filters applied separately after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

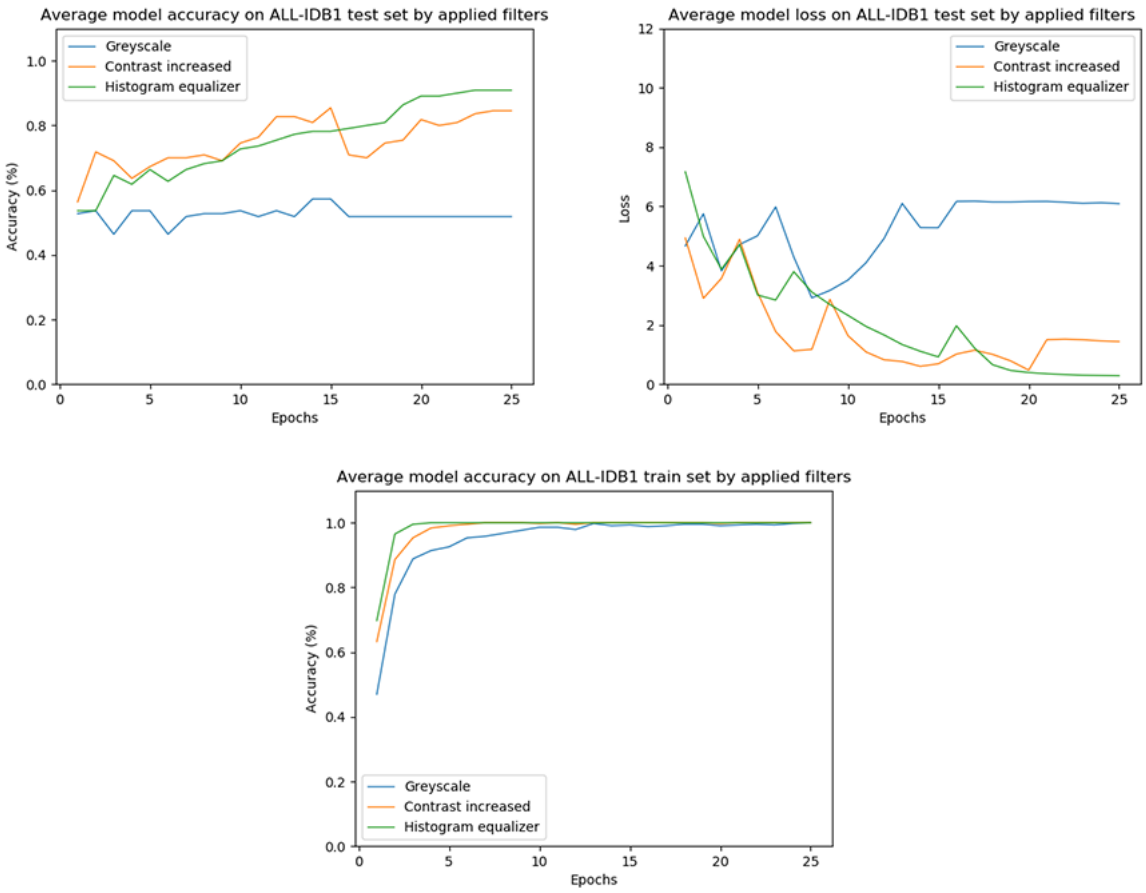


Figure A.2: Results of experiment A.2 (On ALL-IDB1, with filters applied). The results presented are the average of 5 different runs for each filter applied.

A.1.3 Experiment A.3 - Size 200x200

The results that we have obtained using the baseline CNN and a resize of the images by 100x100 were not as good as the reference article [29] even though they were still correct. An hypothesis of this issue would have been the size of these images. This experiment aims to test the size feature for the learning of the model. To do that, we have then use the same baseline CNN as before but here with a preprocessing of 200x200. The images contains now 40000 pixels which is 4 times more than in experiment A.1.

Nevertheless, what we can observe from the results is that it does not enhance the accuracy on test set and has lower results in average. These results thus does not allow to draw a positive conclusion about the effectiveness of using larger images. Looking to the graphs, we can observe that the accuracy has a continuous behaviour and the model seems not to learn more

along epochs. This can be due to the number of images of our training set (86 images) which does not allow the model to learn efficiently larger images.

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**
ACC-TEST	63.63%	40.9%	54.54%	77.27%	59.09%	59.08%
LOSS-TEST	2.47	5.94	1.96	0.51	0.96	2.36
ACC-TRAIN	100%	100%	100%	100%	100%	100%

Table A.3: Experiment A.3 (On ALL-IDB1, with larger resize) - Train set = 86 images, Test set = 22 images - Detailed results of each run after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

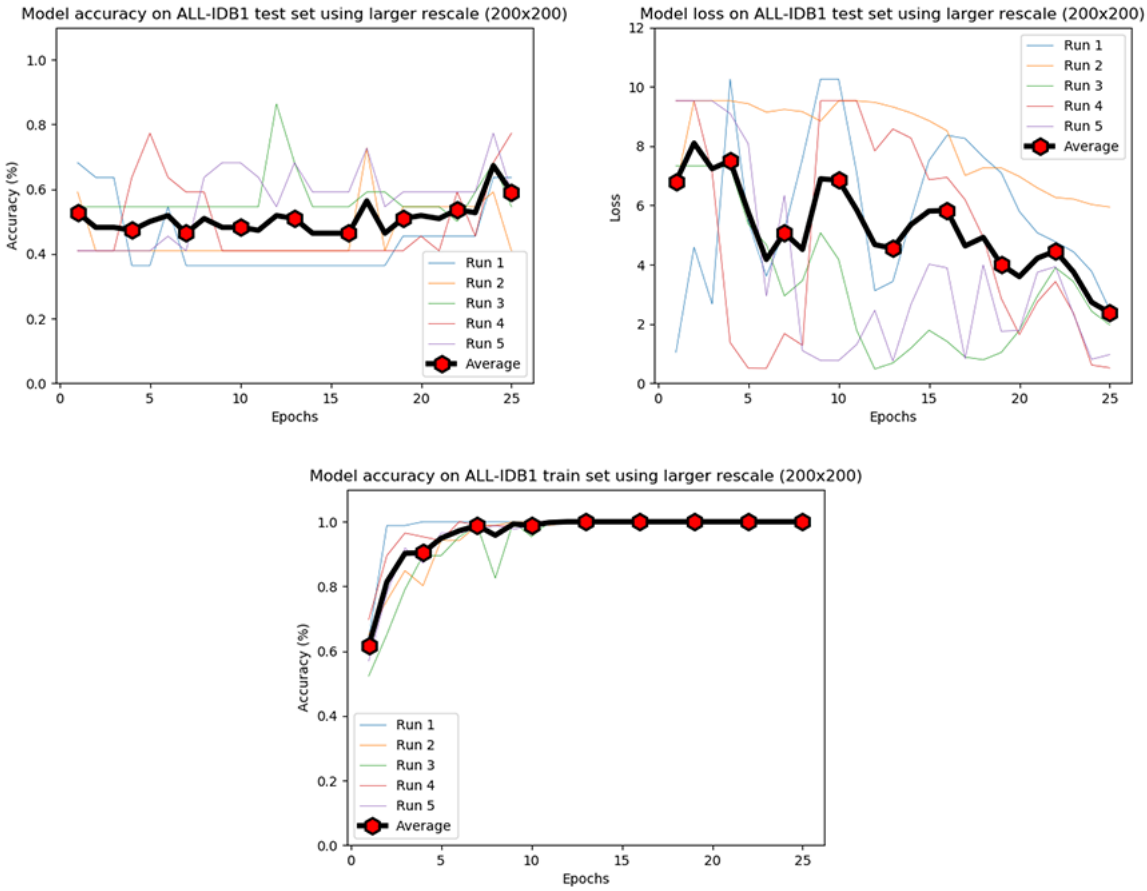


Figure A.3: Results of experiment A.3 (On ALL-IDB1, with larger resize 200x200) - The average result of the 5 runs after 25 epochs: (1) Accuracy on test set: **68.45%** (2) Loss value on test set: **0.992** (3) Accuracy on train set: **100%**

A.1.4 Experiment A.4 - Noise

A recurrent question was about how the model reacts to noise. In this experiment, we try different level of gaussian noise added. Respectively, we try the following values of noise: [0.1,0.2,0.4,0.6,0.8]. In subsection 5.3.6, we have seen that noise can be used as regularization if used sparingly. Is that also valid for medical images?

Following the results, we can mention that this regularization effect is still valid if a very small value of noise is applied, not more than 0.2. From 0.4, we begin to see bad impact of noise such that the model is not even capable to reach 100% accuracy on training set after 25 epochs. Moreover, on the third graph representing the learning speed of the model (accuracy on train set) we can observe the more noise applied, the slower the model is to converge. Thus, the threshold noise value to have a regularization effect should be 0.2. Above this, the value of cost function is increasing constantly which means that the model is making lot of error between predicted values and expected values. The model has wrong abilities to estimate the relationship between the label and the input.

This is also verified when looking to the accuracy results on test set. From `noise=0.4`, we approach to a prediction rate of 50% the total prediction randomness.

	noise=0.1	**noise=0.2**	**noise=0.4**	**noise=0.6**	**noise=0.8**
ACC-TEST	77.27%	63.63%	59.09%	59.09%	54.54%
LOSS-TEST	0.67	1.64	1.75	4.33	7.32
ACC-TRAIN	100%	100%	97.67%	94.18%	84.88%

Table A.4: Experiment A.4 (On ALL-IDB1, with different level of noise) - Train set = 86 images, Test set = 22 images - Average results depending on noise level. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

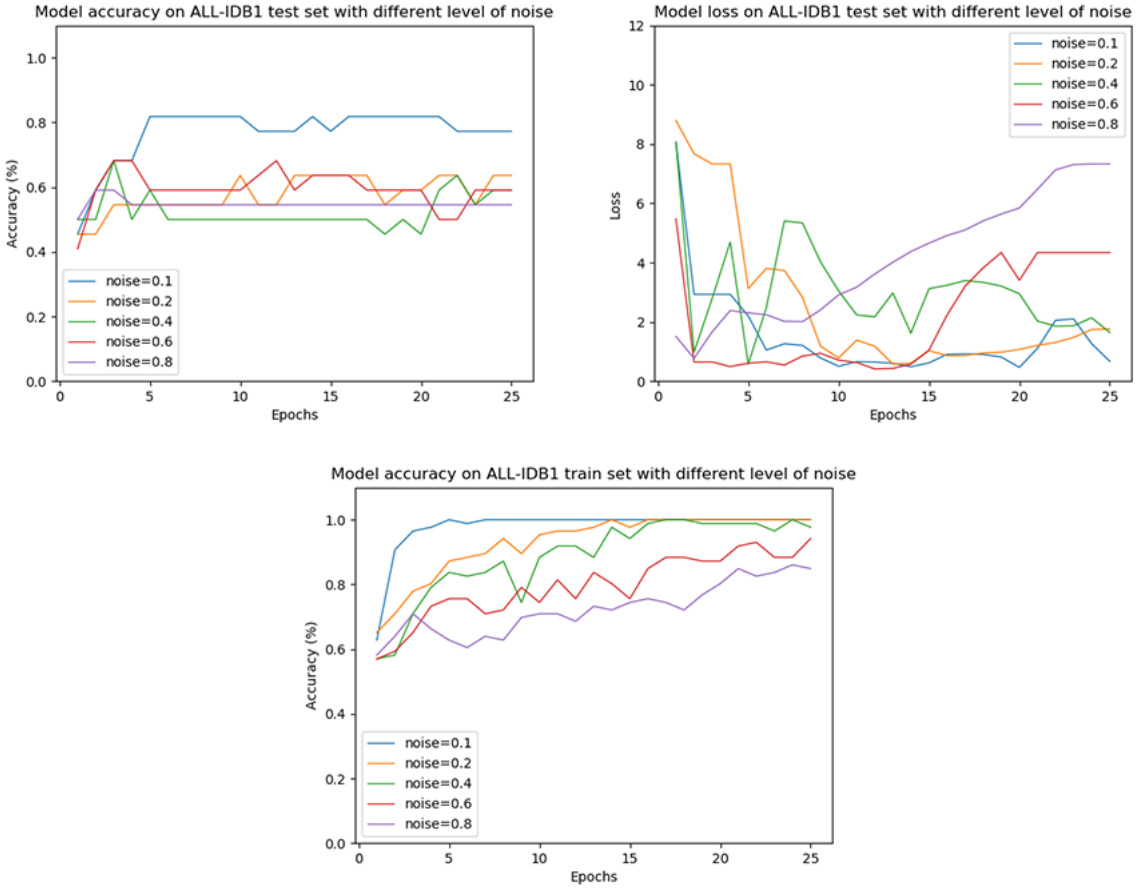


Figure A.4: Results of experiment A.4 (On ALL-IDB1, with different level of noise). The results presented are the average of 5 different runs for each noise level.

A.1.5 Experiment A.5 - Dropout

As reminder, dropout is a regularization technique that consists to drop some neurons during the training phase (see subsection 5.3.3 for more information). It is generally used to counter overfitting. When we have used it on CIFAR-10 dataset constituted of 50000 images for training, it had very good impact on the learning phase. Here, we have only 86 images for training which is very low, will it be as efficient as using it on large dataset such as CIFAR-10? Obviously no. On such a small dataset, the model does not suffer from overfitting but more about underfitting. Using dropout is then a very bad idea since it takes out neurons to the model and make it way more difficult to learn. The results obtained show it well. We have high losses values and bad accuracy rates. Looking to this last on test set graph, we observe a continuous behaviour around 50% rate. Same for the accuracy on train set, it reaches in average 61.85% accuracy after 25 epochs. That shows the disability for the model to learn correctly.

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**
ACC-TEST	59.09%	40.9%	50.00%	59.09%	63.63%	54.54%
LOSS-TEST	1.07	9.52	8.05	6.11	4.15	5.78
ACC-TRAIN	62.79%	58.13%	54.65%	60.46%	73.25%	61.85%

Table A.5: Experiment A.5 (On ALL-IDB1, with dropout=0.1) - Train set = 86 images, Test set = 22 images - Detailed results of each run after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

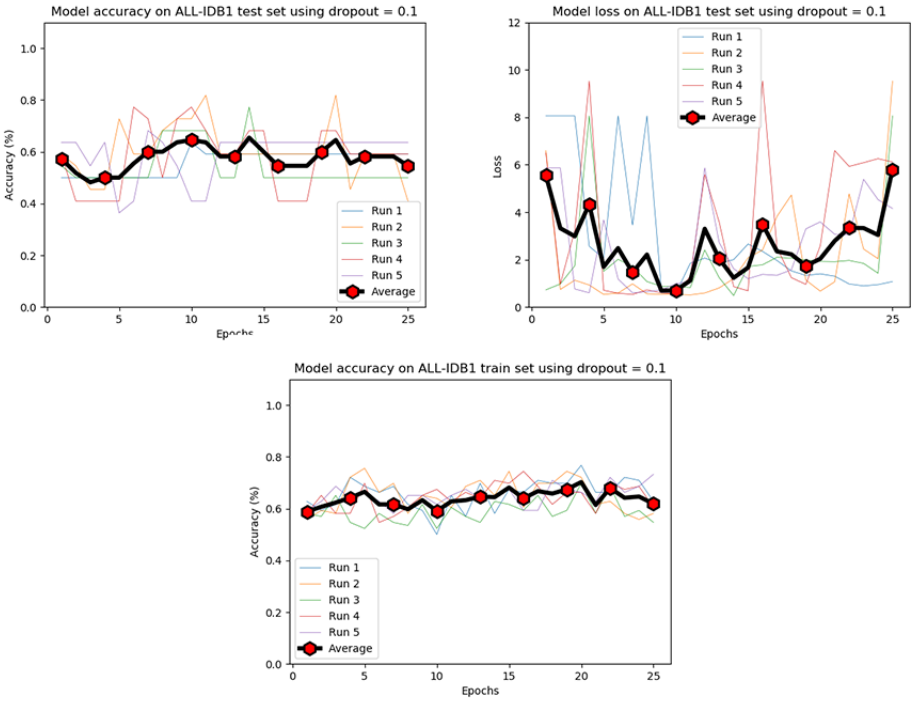


Figure A.5: Results of experiment A.5 (On ALL-IDB1, with dropout=0.1) - The average result of the 5 runs after 25 epochs: (1) Accuracy on test set: **54.54%** (2) Loss value on test set: **5.78** (3) Accuracy on train set: **61.85%**

A.1.6 Experiment A.6 - Data augmentation

In this experiment, we have used data augmentation in order to expand the dataset. The method used is explained in section 6.4. The results obtained are quite good and show well the efficiency of using data augmentation. In average, we gain 7% accuracy and have low loss value when using data augmentation. The images' direction and sense seems to not disturb the model's learning and looks like it is not an impacting feature to determinate the label.

Nevertheless, some runs as run 4 show an unconvincing performance (63.63%). This can be due to a type of image augmentation used: rotations. In fact, rotations create some black area and represents a considerable percentage of the image. Thus, the model could have taken it in account and considers it as a feature of the corresponding label which confuses the learning of the model. As data augmentation is done randomly between all the types of image augmentation explained previously, this run could have been implied by this issue and then impact negatively the learning and prediction.

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**
ACC-TEST	72.72%	81.81%	65.9%	63.63%	77.27%	72.26%
LOSS-TEST	1.17	0.46	1.25	2.76	0.91	1.31
ACC-TRAIN	98.83%	100%	100%	99.41%	100%	99.64%

Table A.6: Experiment A.6 (On ALL-IDB1, with data augmentation) - Train set = 172 images, Test set = 44 images - Detailed results of each run after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

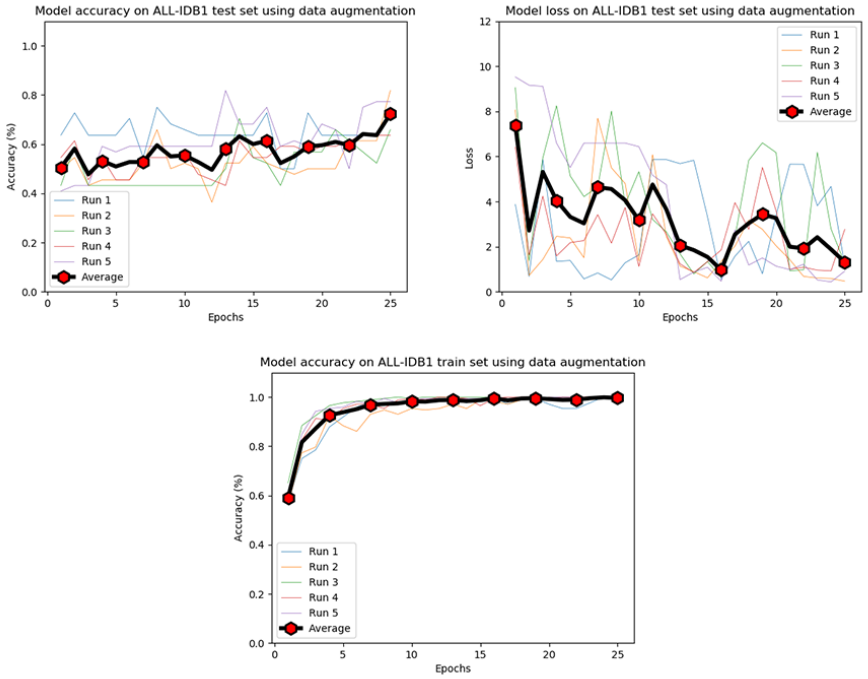


Figure A.6: Results of experiment A.6 (On ALL-IDB1, with data augmentation) - The average result of the 5 runs after 25 epochs: (1) Accuracy on test set: **72.26%** (2) Loss value on test set: **1.31** (3) Accuracy on train set: **99.64%**

A.1.7 Experiment A.7 - Dropout and data augmentation

A problem of using dropout was the size of our training set. If we use larger dataset using data augmentation, will dropout be more efficient this time? In this experiment, we try to combine dropout and data augmentation. However, from what we can observe by the results, dropout is still counterproductive for the model's learning. Even if the results are more encouraging than experiment A.5 where we use only dropout, they are still not as good as using only data augmentation. That shows that the model is still not overfit and need way more data to suffer from it in order to have regularization effect while using dropout.

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**
ACC-TEST	54.54%	59.09%	77.27%	65.9%	72.72%	65.9%
LOSS-TEST	1.13	1.95	0.84	1.21	0.7	1.16
ACC-TRAIN	64.53%	65.11%	68.60%	58.72%	68.02%	64.99%

Table A.7: Experiment A.7 (On ALL-IDB1, with data augmentation and dropout=0.1) - Train set = 172 images, Test set = 44 images - Detailed results of each run after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

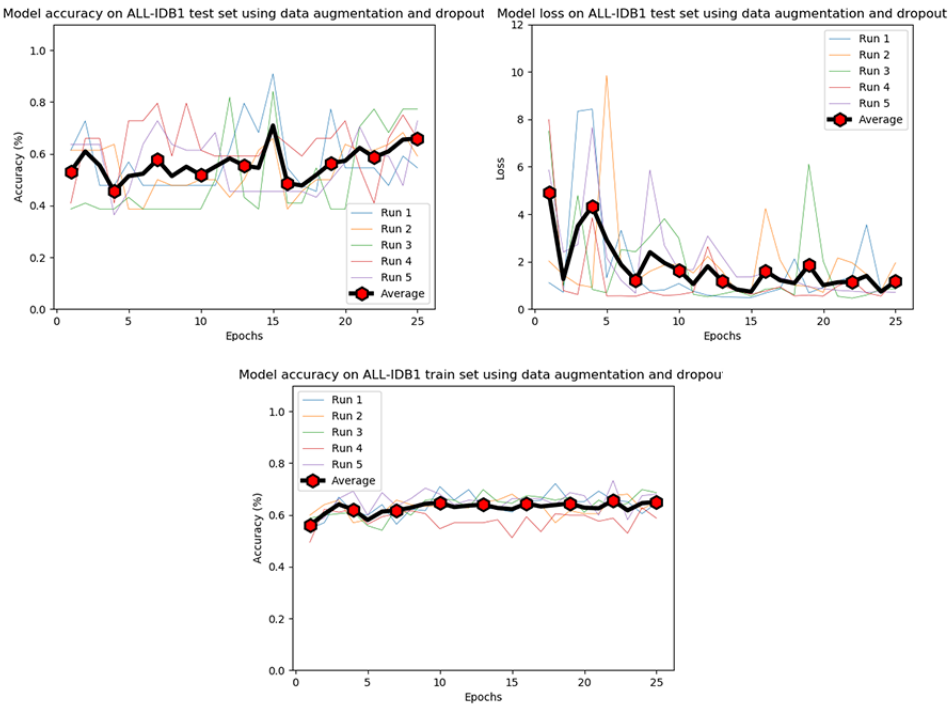


Figure A.7: Results of experiment A.7 (On ALL-IDB1, with data augmentation and dropout=0.1) - The average result of the 5 runs after 25 epochs: (1) Accuracy on test set: **65.9%** (2) Loss value on test set: **1.16** (3) Accuracy on train set: **64.99%**

A.2 Case B - ALL-IDB2

Compared to ALL-IDB1, ALL-IDB2 has been already preprocessed by image segmentation on the region of interest (ROI). This dataset contains 260 images that focus on an unique blood cell. We then have 208 images for training and 52 images for testing when not using data augmentation. Otherwise, we have 416 images for training and 104 for testing. Again, more information are described in section 6.2. We aim then to compare results without segmentation (ALL-IDB1) and results with segmentation (ALL-IDB2) to observe the efficiency of such preprocessing. To do that, we will do the same experiments as ALL-IDB1. The images of the dataset have also been resized to 100x100, their original sizes were 257x257.

A.2.1 Experiment B.1 - Baseline

When using baseline CNN on the original resized dataset, the results seems to be worse than when we use ALL-IDB1 but still around 62-65% accuracy. At first sight, determinate if the patient is affected by the disease by using only an unique blood cell is more difficult as we use a more global view of multiple blood cells to determinate if the patient is affected by the disease or not. The features that determinates the label may be more accurate when the only information given is a blood cell, i.e blood cells are very similar to each other. Therefore, we need a larger dataset to obtain better results. Nevertheless can we make these features more impacting to the model? That is what we will observe in the next section by using some filters.

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**
ACC-TEST	63.46%	63.46%	67.3%	55.76%	59.61%	61.91%
LOSS-TEST	3.22	0.65	0.79	2.59	2.05	1.86
ACC-TRAIN	100%	100%	100%	100%	100%	100%

Table A.8: Experiment B.1 (On ALL-IDB2, with Baseline) - Train set = 208 images, Test set = 52 images - Detailed results of each run after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

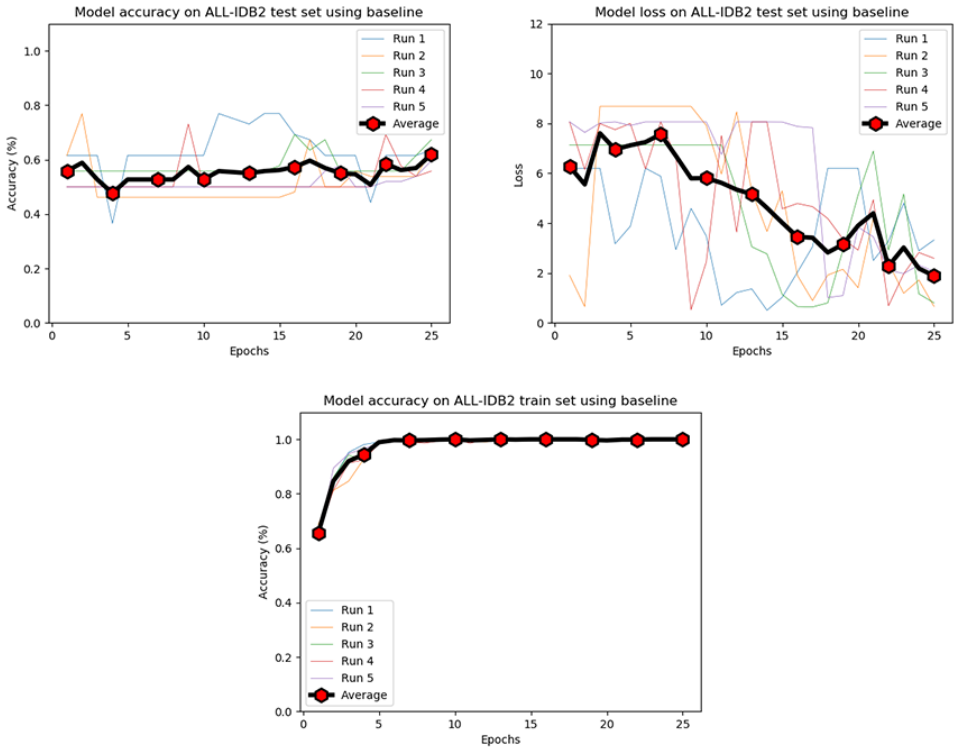


Figure A.8: Results of experiment B.1 (On ALL-IDB2, with Baseline) - The average result of the 5 runs after 25 epochs: (1) Accuracy on test set: **61.91%** (2) Loss value on test set: **1.86** (3) Accuracy on train set: **100%**

A.2.2 Experiment B.2 - Filters

Same as ALL-IDB1, we will experiment 3 different filters: transforming images into greyscale, increasing images' contrast with an arbitrary value of 4 and using histogram equalization on the images.

Greyscale

In contrary to ALL-IDB1, here we obtain more convincing results. As we manage only one blood cell, transforming the images into greyscale allow the model to highlight the shapes/edges of the cell and its content more clearly.

Contrast increased

Same as ALL-IDB1, increase the contrast to the images seems to be a good image enhancement technique. Again, edges and shapes are highlighted and facilitates the model's learning.

Histogram equalization

This time, histogram equalization does not give the best results even though it still gives very good ones. Histogram equalization technique is then a safe method to enhance correctly the image without making it unreadable unlike if we put a too large contrast increase value.

	Greyscale	Increased contrast	Histogram equalization
Accuracy on test set	71.53%	80.38%	73.84%
Loss value on test set	1.09	0.47	0.95
Accuracy on train set	99.9%	100%	100%

Table A.9: Experiment B.2 (On ALL-IDB2, with filters applied) - Train set = 208 images, Test set = 52 images - Average results for each filters applied separately after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

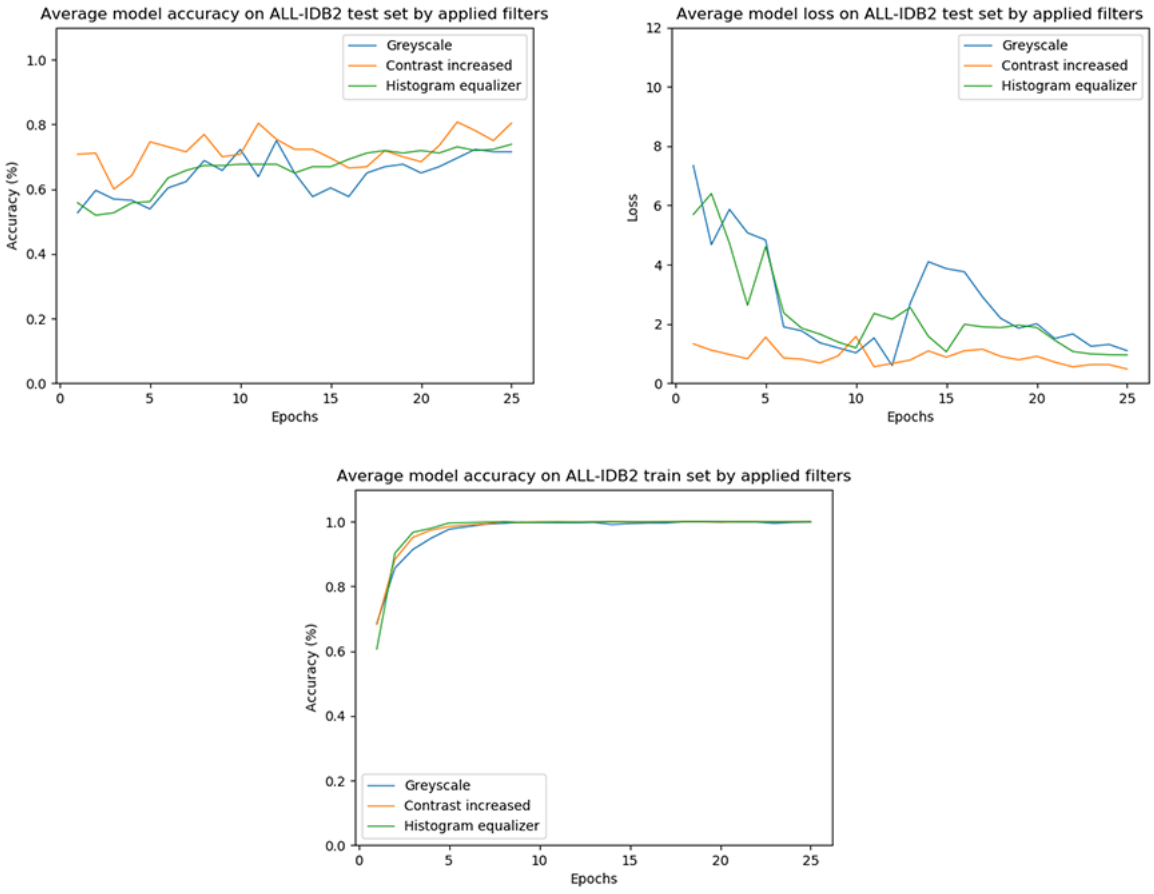


Figure A.9: Results of experiment B.2 (On ALL-IDB2, with filters applied). The results presented are the average of 5 different runs for each filter applied.

A.2.3 Experiment B.3 - Size 200x200

The original images from ALL-IDB2 dataset have a size of 257x257. In this section, we will use 200x200 images and can then be considered as the original size. For ALL-IDB1, we have seen that enlarge the images was not the best method to increase the performance due to the very small numbers of samples contained in the dataset. Here for ALL-IDB2, we have 2 times more images and we use the original size of the dataset. So that, we can observe that we obtain slightly better results. Using larger images when having much samples seems then to be a good idea and constitutes a good asset to feature detection.

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**
ACC-TEST	69.23%	65.38%	75.00%	67.3%	57.69%	66.92%
LOSS-TEST	0.52	0.68	0.65	0.94	1.23	0.804
ACC-TRAIN	100%	100%	100%	99.51%	100%	99.902%

Table A.10: Experiment B.3 (On ALL-IDB2, with larger resize) - Train set = 208 images, Test set = 52 images - Detailed results of each run after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

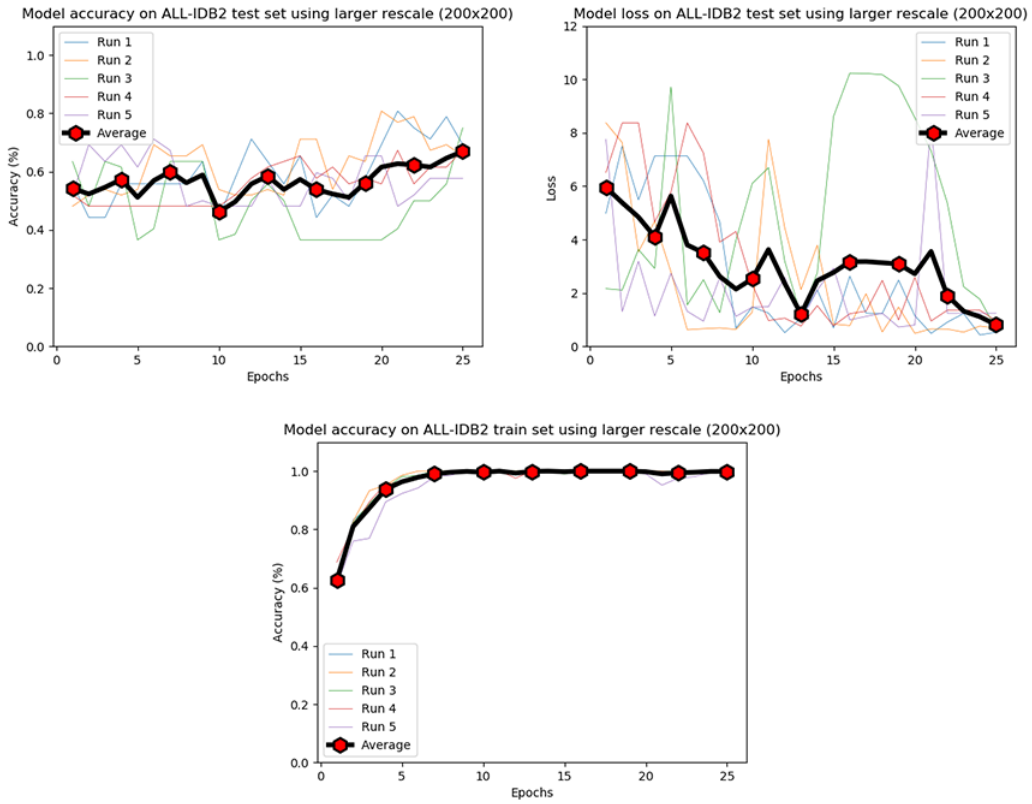


Figure A.10: Results of experiment B.3 (On ALL-IDB2, with larger resize 200x200) - The average result of the 5 runs after 25 epochs: (1) Accuracy on test set: **68.92%** (2) Loss value on test set: **0.804** (3) Accuracy on train set: **99.902%**

A.2.4 Experiment B.4 - Noise

Overall, the results seems to have the same behaviour than the results when using ALL-IDB1 dataset. At first, we have a regularization effect when using small noise value. But when using a too large value – here still 0.4, noise has a very bad impact on the model's learning. At this time, loss values are exploding and almost grow exponentially. The explanations from subsection A.1.4 are also valid for ALL-IDB2.

	noise=0.1	**noise=0.2**	**noise=0.4**	**noise=0.6**	**noise=0.8**
ACC-TEST	71.15%	63.46%	50.00%	55.76%	46.13%
LOSS-TEST	1.41	2.34	3.60	6.45	7.12
ACC-TRAIN	100%	99.5%	98.55%	97.55%	94.71%

Table A.11: Experiment B.4 (On ALL-IDB2, with different level of noise) - Train set = 208 images, Test set = 52 images - Average results depending on noise level. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

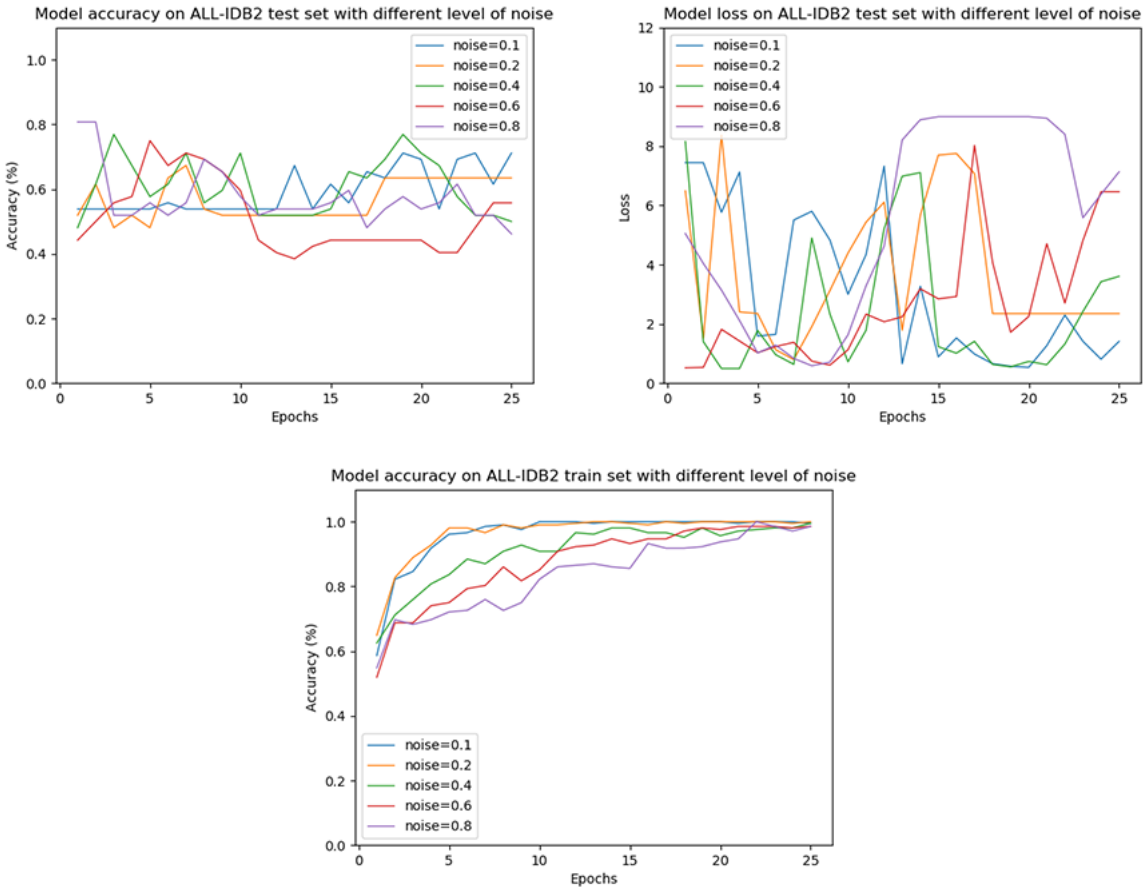


Figure A.11: Results of experiment B.4 (On ALL-IDB2, with different level of noise). The results presented are the average of 5 different runs for each noise level.

A.2.5 Experiment B.5 - Dropout

Using dropout when your dataset is very small is a technique to forget: that is the conclusion that we have made after the experiment A.5 using dropout on ALL-IDB1 that contains only 108 images. Here we have a little bit much images on ALL-IDB2 and we can remark that overall it still has bad results but we can see a little augmentation on average training set accuracy results. That reinforces the idea that dropout begins to have a regularization effect when we have very large datasets. That is an additional proof of this assertion that we have done previously in experiment A.7 (see subsection A.1.7). Of course, for this case, we are far from having a dataset size conductive to dropout as regularizer. That explains the bad results.

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**
ACC-TEST	55.76%	53.84%	48.07%	51.92%	67.30%	55.37%
LOSS-TEST	1.25	1.58	2.40	3.57	0.7	1.9
ACC-TRAIN	73.07%	70.67%	64.42%	62.5%	70.0%	68.12%

Table A.12: Experiment B.5 (On ALL-IDB2, with dropout=0.1) - Train set = 208 images, Test set = 52 images - Detailed results of each run after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

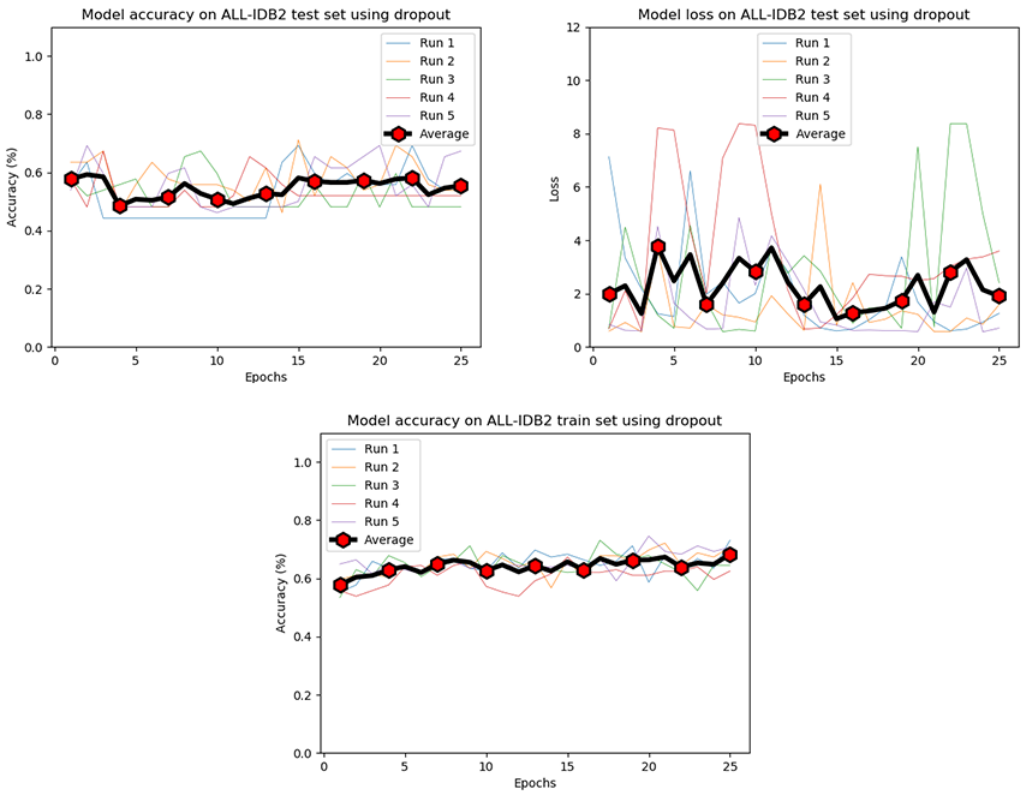


Figure A.12: Results of experiment B.5 (On ALL-IDB2, with dropout=0.1) - The average result of the 5 runs after 25 epochs: (1) Accuracy on test set: **55.37%** (2) Loss value on test set: **1.9** (3) Accuracy on train set: **68.12%**

A.2.6 Experiment B.6 - Data augmentation

One more time, data augmentation shows having a good impact on model's performance. These experiments give some indications about the efficiency of data augmentation especially when using small datasets like ALL-IDB1 or ALL-IDB2. This data augmentation techniques give many more samples to the dataset and allow the model to learn more in order to make better predictions.

Furthermore, the different image manipulations that we have done during this data augmentation seems to not be impacting negatively the feature detection step whether for segmented and non-segmented images. Thus, data augmentation constitutes a must-use asset to increase the performance of the model.

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**
ACC-TEST	64.42%	75.96%	72.11%	67.30%	79.80%	71.91%
LOSS-TEST	1.44	0.67	1.30	1.67	0.91	1.198
ACC-TRAIN	99.75%	99.75%	100%	100%	100%	99.9%

Table A.13: Experiment B.6 (On ALL-IDB2, with data augmentation) - Train set = 416 images, Test set = 104 images - Detailed results of each run after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

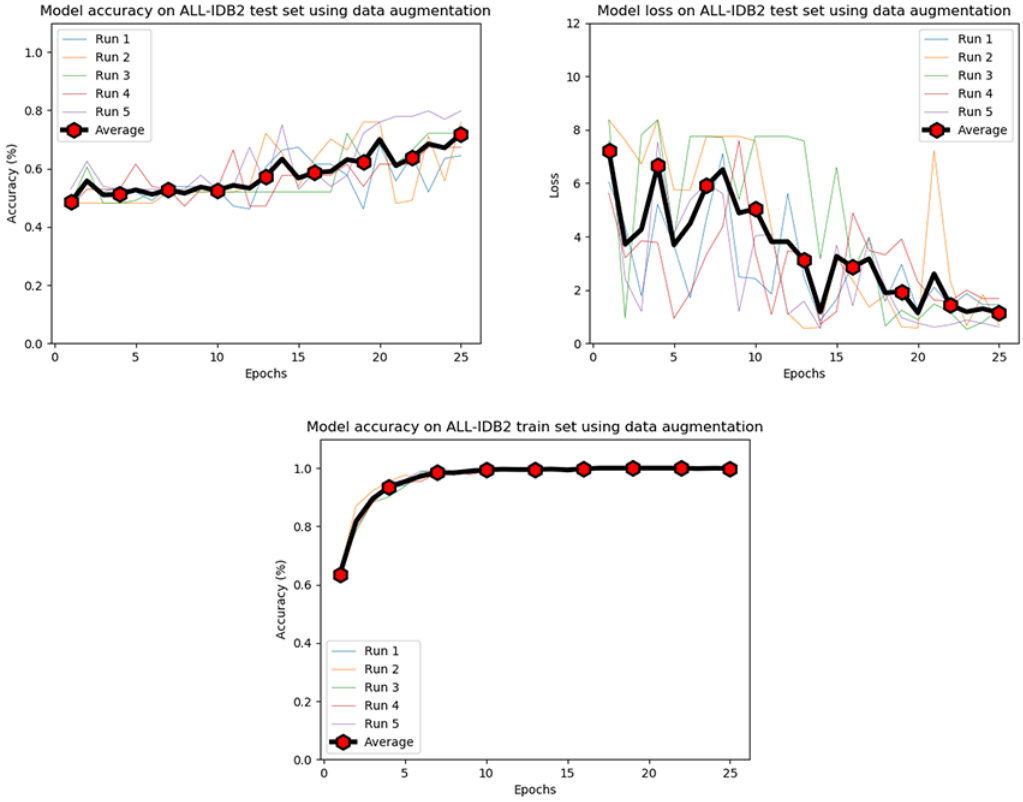


Figure A.13: Results of experiment B.6 (On ALL-IDB2, with data augmentation) - The average result of the 5 runs after 25 epochs: (1) Accuracy on test set: **71.91%** (2) Loss value on test set: **1.198** (3) Accuracy on train set: **99.9%**

A.2.7 Experiment B.7 - Dropout and data augmentation

Similarly to the experiments on dropout, we observe the same issue as usual when using dropout on small datasets. The model cannot learn correctly to the neurons removed and cannot reach more than 70% accuracy on training set. As this issue was explained before, refer to experiments A.5, A.7 and B.5 for more explanations.

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**
ACC-TEST	66.34%	64.42%	58.65%	65.38%	68.26%	64.61%
LOSS-TEST	0.79	0.7	0.93	1.53	0.926	0.97
ACC-TRAIN	67.22%	66.74%	64.33%	62.16%	65.78%	65.24%

Table A.14: Experiment B.7 (On ALL-IDB2, with data augmentation and dropout=0.1) - Train set = 416 images, Test set = 104 images - Detailed results of each run after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

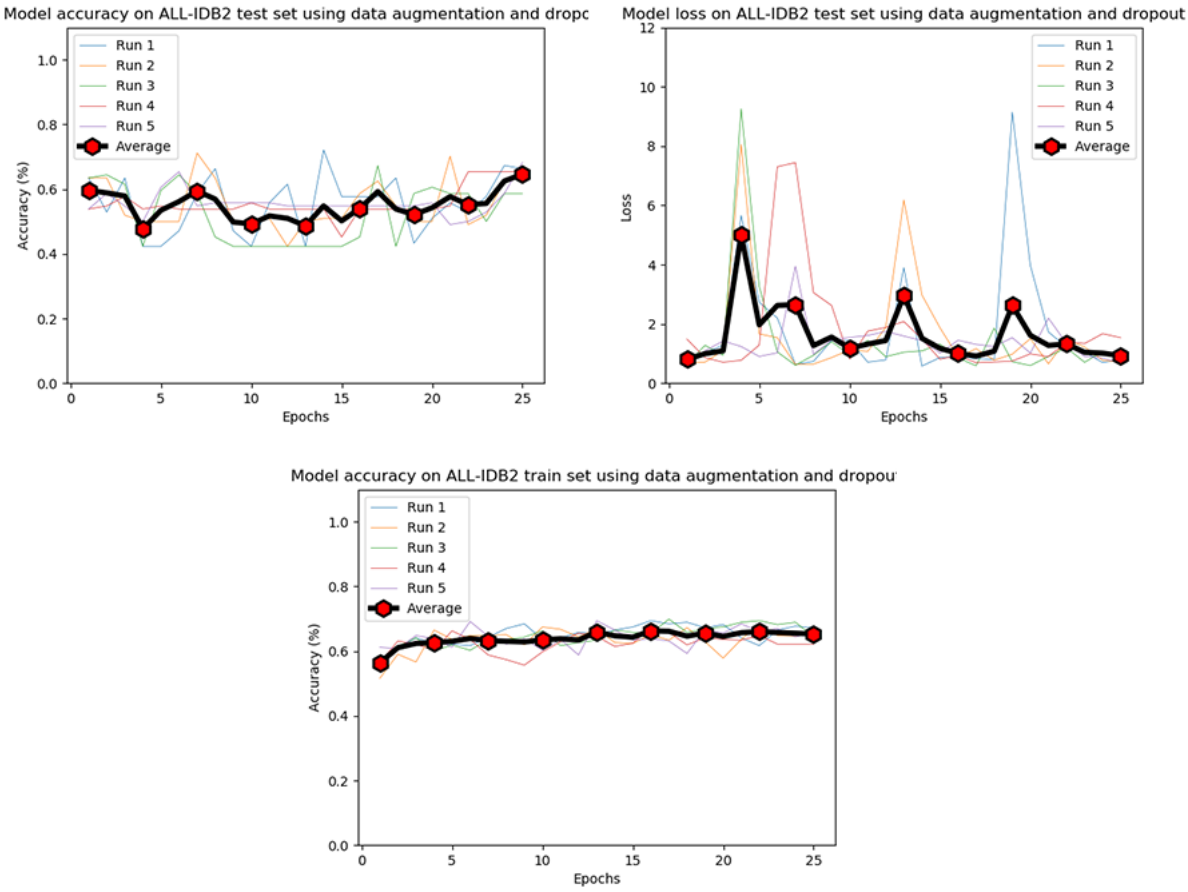


Figure A.14: Results of experiment B.7 (On ALL-IDB2, with data augmentation and dropout=0.1) - The average result of the 5 runs after 25 epochs: (1) Accuracy on test set: **65.9%** (2) Loss value on test set: **1.16** (3) Accuracy on train set: **64.99%**

A.3 Case C - ALL-IDB and DEMIR-LEUKEMIA datasets

The last case deals with ALL-IDB and DEMIR-LEUKEMIA datasets which involves 5 classes: HEALTHY (0), ALL (1), AML (2), CLL (3), CML (4). When not using data augmentation, we consider 732 images for training set and 183 for test set. Otherwise, we have 1464 images for training set and 366 for test set. That makes way more samples than two previous cases but here we also have 3 more classes to predict. The prediction accuracy rate that defines the randomness of the model is then 20% (not 50% like a binary classification). In the reference article [29], they have obtained in average an accuracy of 81.74% for test set, 1.14 of loss value and 99.34% for train set after 25 epochs using data augmentation and dropout, with SGD optimizer and no preprocessing step (no resize, no filter).

A.3.1 Experiment C.1 - Baseline

Compared to the reference article [29] results mentioned above, we are far from the results obtained. As matter of fact, we only obtain 49.61% accuracy on test set. Nevertheless, we have to keep in mind that the reference result is obtained with no resizing. When processing a large amount of images and labelled with 5 classes, size matter a lot and allow the model to learn more. Moreover, the images provided in DEMIR-LEUKEMIA dataset contains a lot of variety of images such that some of them contain features that may confuse the learning of the model such as watermarks or bad resolution. Thus, the results obtained with our images resized at 100x100 is not that bad as it is very far from a totally random prediction. It means that the model learns well even if we use very small images. To enhance those results, are more preprocessing steps necessary or do we have to use more images? That is the subject of the next sections.

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**
ACC-TEST	56.83%	39.89%	44.8%	51.36%	55.19%	49.61%
LOSS-TEST	2.38	6.59	2.32	2.14	1.26	2.93
ACC-TRAIN	99.59%	99.72%	99.72%	99.86%	99.59%	99.69%

Table A.15: Experiment C.1 (On ALL-IDB+DEMIR, with Baseline) - Train set = 732 images, Test set = 183 images - Detailed results of each run after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

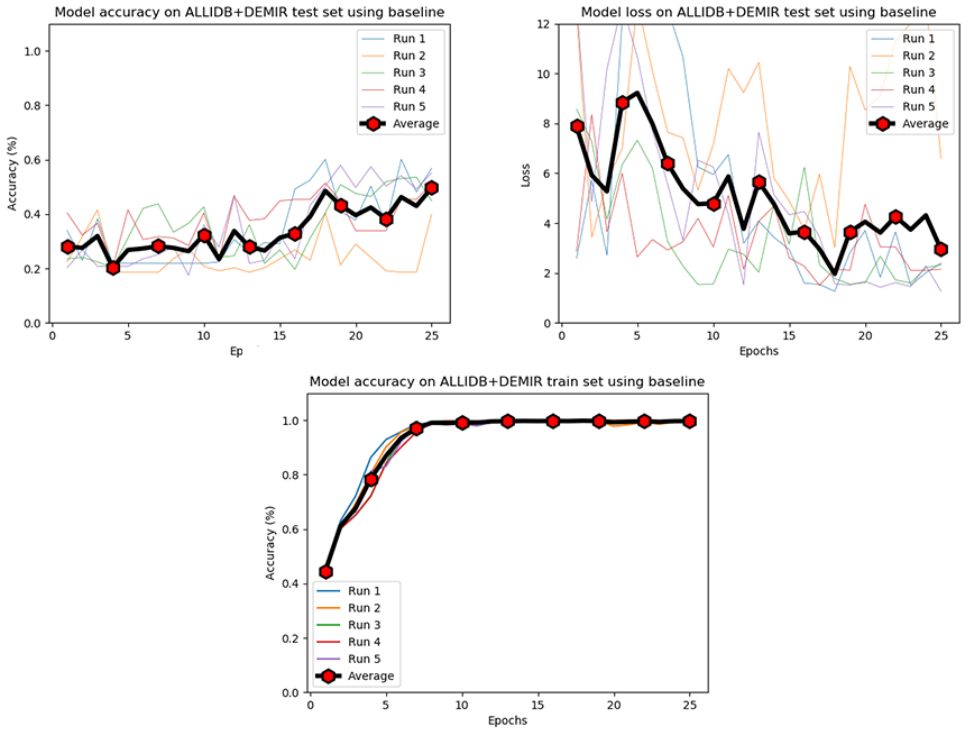


Figure A.15: Results of experiment C.1 (On ALL-IDB+DEMIR, with Baseline) - The average result of the 5 runs after 25 epochs: (1) Accuracy on test set: **49.61%** (2) Loss value on test set: **2.93** (3) Accuracy on train set: **99.69%**

A.3.2 Experiment C.2 - Filters

As we have seen for experiments done for ALL-IDB dataset, image enhancement using filters like greyscale or contrast enhancement seemed like an efficient technique to increase the performance of our model. This observation is confirmed while looking to the results obtained. We really have convincing results and the image enhancement step looks like being mandatory to gain accuracy rate. In this case, the three filters tested achieve quite similar results even if histogram equalization still have slightly better results.

	Greyscale	**Increased contrast**	**Histogram equalization**
Accuracy on test set	59.01%	58.68%	61.74%
Loss value on test set	2.29	2.05	1.63
Accuracy on train set	99.86%	99.72%	99.72%

Table A.16: Experiment C.2 (On ALL-IDB+DEMIR, with filters applied) - Train set = 732 images, Test set = 183 images - Average results for each filters applied separately after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

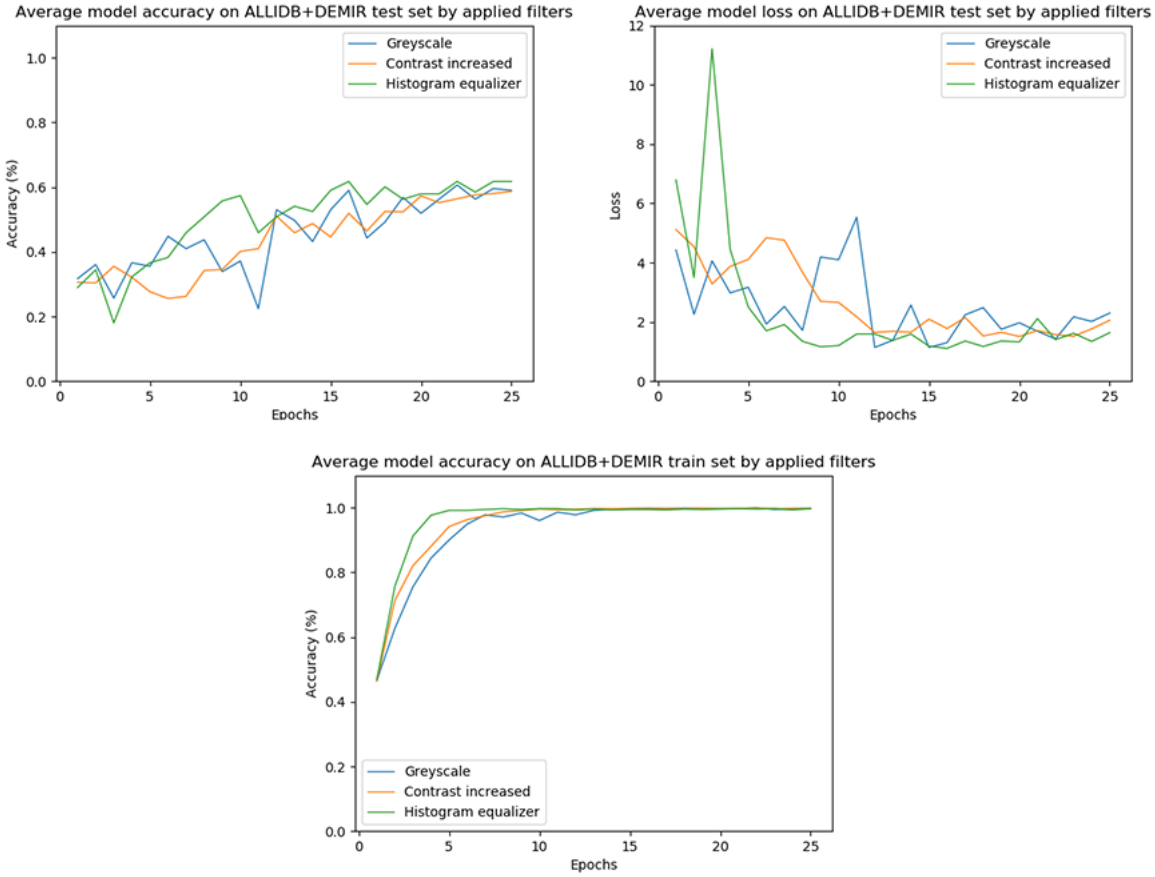


Figure A.16: Results of experiment C.2 (On ALL-IDB+DEMIR, with filters applied). The results presented are the average of 5 different runs for each filter applied.

A.3.3 Experiment C.3 - Size 200x200

To have an idea about the importance of using larger images in a case where we have more than 2 classes and more images, we do an experiment with the images of the dataset resized to 200x200. We observe a better performance in average and also a very low loss value. That shows the efficiency of using larger images and explain the very good results obtained by the authors of the reference article [29]. Moreover, we can observe that the model arrives to a convergence state as fast as using 100x100 images which means that it is not really constraining for the model's learning. However, the huge drawback of using larger images is the execution time that grow exponentially depending to the images' sizes.

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**
ACC-TEST	65.02%	67.75%	56.28%	54.09%	56.83%	59.99%
LOSS-TEST	0.97	1.36	1.65	1.45	1.04	1.29
ACC-TRAIN	99.72%	99.86%	99.31%	99.72%	99.86%	99.69%

Table A.17: Experiment C.3 (On ALL-IDB+DEMIR, with larger resize) - Train set = 732 images, Test set = 183 images - Detailed results of each run after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

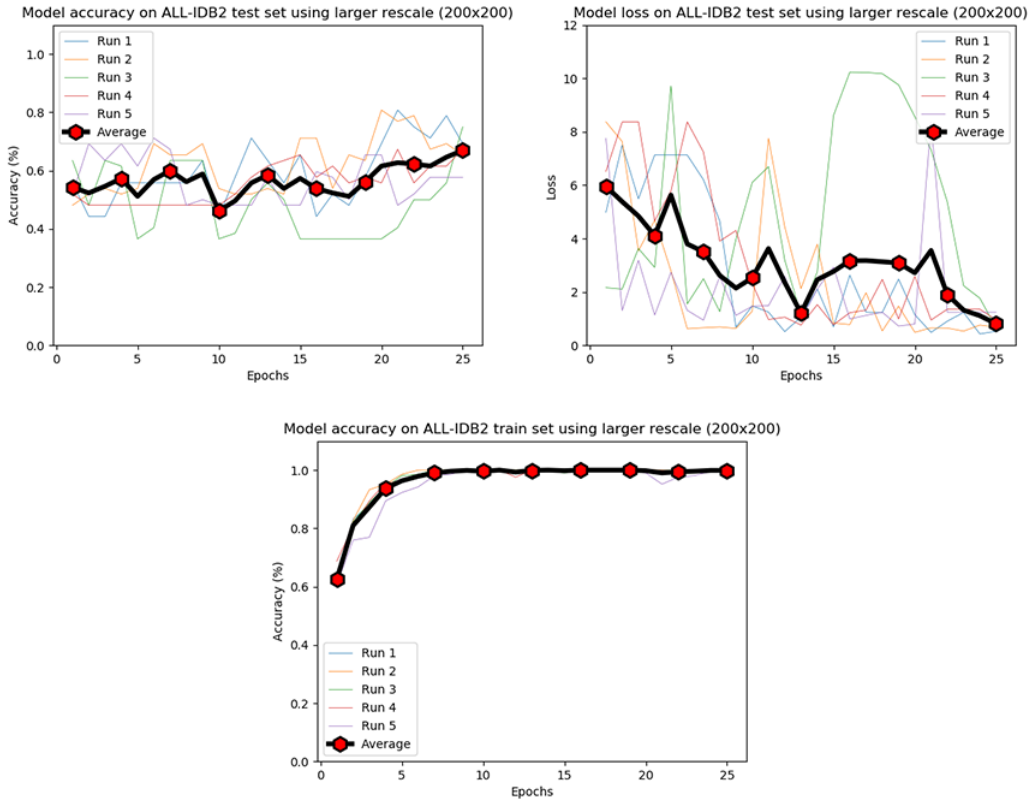


Figure A.17: Results of experiment C.3 (On ALL-IDB+DEMIR, with larger resize 200x200) - The average result of the 5 runs after 25 epochs: (1) Accuracy on test set: **59.99%** (2) Loss value on test set: **1.29** (3) Accuracy on train set: **99.69%**

A.3.4 Experiment C.4 - Noise

When testing the ability of the model to learn when noise is added, the previous experiments on ALL-IDB dataset have shown that the performance decreases drastically when adding more than 20% of noise. In this case, we see that this performance is badly affected already at 20% of noise added. Indeed, as we have way more classes and then way more features to learn, the model's ability to learn is more impacted. That is well illustrated in the third graph showing the evolution of accuracy rate on training set where we can observe that more noise added, more the model takes time to reach convergence. Of course, that proportionality impact the accuracy on test set and the cost of loss function that increases depending of noise added which means that the model is getting harder and harder to make the link between the input image and its label.

	noise=0.1	noise=0.2	noise=0.4	noise=0.6	noise=0.8
ACC-TEST	51.36%	32.78%	28.41%	25.68%	14.75%
LOSS-TEST	2.83	4.23	4.92	5.64	8.80
ACC-TRAIN	99.59%	98.22%	97.81%	96.17%	94.67%

Table A.18: Experiment C.4 (On ALL-IDB+DEMIR, with different level of noise) - Train set = 732 images, Test set = 183 images - Average results depending on noise level. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

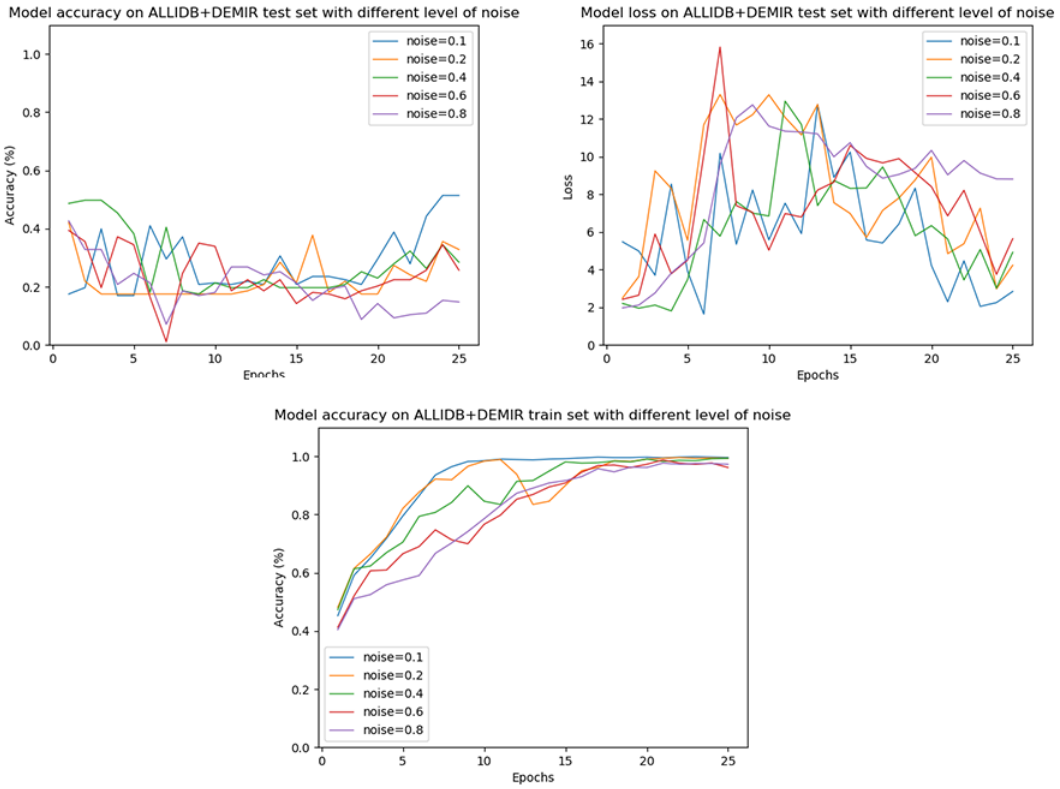


Figure A.18: Results of experiment C.4 (On ALL-IDB+DEMIR, with different level of noise). The results presented are the average of 5 different runs for each noise level.

A.3.5 Experiment C.5 - Dropout

The conclusion that arise from the previous experiments on dropout efficiency were that it does not make sense to use it when we have so few samples in our dataset. Here we can think that as we have more images, dropout will show a regularization effect. But here we also have 3 more classes. Having 3 times more images therefore amounts to the same conclusion than previous experiments. Thus, this assertion is verified by the results obtained in this experiment. In average we obtain 25.89% accuracy on test set which is very near from the total randomness prediction. Moreover, this is also observed by the third graph showing the evolution of accuracy on train set which is almost constant and does not evolve along epochs. That shows the

difficulty of the model to learn. The results show then the same behaviour as experiments A.5 and B.5. The conclusion on dropout done previously is then also valid for this case.

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**AVG**
ACC-TEST	18.57%	23.49%	21.85%	28.96%	36.61%	25.89%
LOSS-TEST	4.77	5.63	4.75	2.23	1.36	3.74
ACC-TRAIN	46.03%	45.62%	51.63%	53.27%	51.36%	49.58%

Table A.19: Experiment C.5 (On ALL-IDB+DEMIR, with dropout=0.1) - Train set = 732 images, Test set = 183 images - Detailed results of each run after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

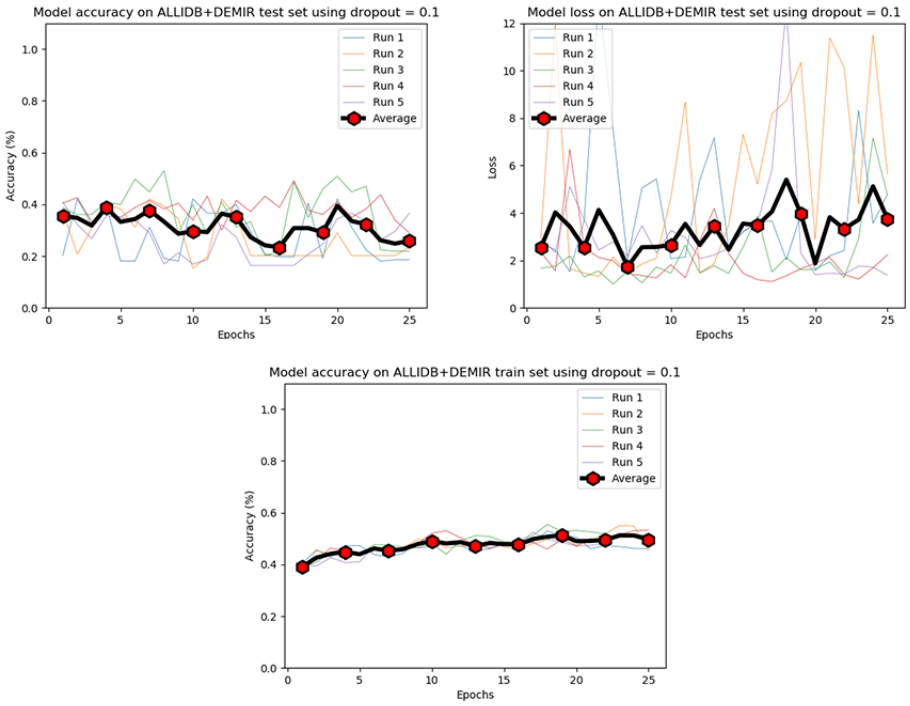


Figure A.19: Results of experiment C.5 (On ALL-IDB+DEMIR, with dropout=0.1) - The average result of the 5 runs after 25 epochs: (1) Accuracy on test set: **25.89%** (2) Loss value on test set: **3.74** (3) Accuracy on train set: **49.58%**

A.3.6 Experiment C.6 - Data augmentation

The good results obtained by the authors of [29] were obtained using data augmentation. Does data augmentation really make the performance of the model increasing? Theoretically from what we have experiment with ALL-IDB dataset and from the results of [29], it is one of the must-use technique to have great performance. The results from this experiment do not make exception of this assertion. Even if we use resized images, we nearly obtain the same results as if we have use the original size of the images. Indeed, we have obtained an average of 71.91% accuracy on test set while the result obtained by the reference article [29] with the original size was about 81%. That really show the efficiency of data augmentation in this type of experiment.

	Run 1	Run 2	Run 3	Run 4	Run 5	AVG
ACC-TEST	77.04%	66.66%	62.56%	55.19%	68.30%	71.91%
LOSS-TEST	0.74	2.75	1.26	1.64	1.39	1.556
ACC-TRAIN	97.33%	96.92%	93.98%	98.01%	99.79%	97.20%

Table A.20: Experiment C.6 (On ALL-IDB+DEMIR, with data augmentation) - Train set = 1464 images, Test set = 266 images - Detailed results of each run after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

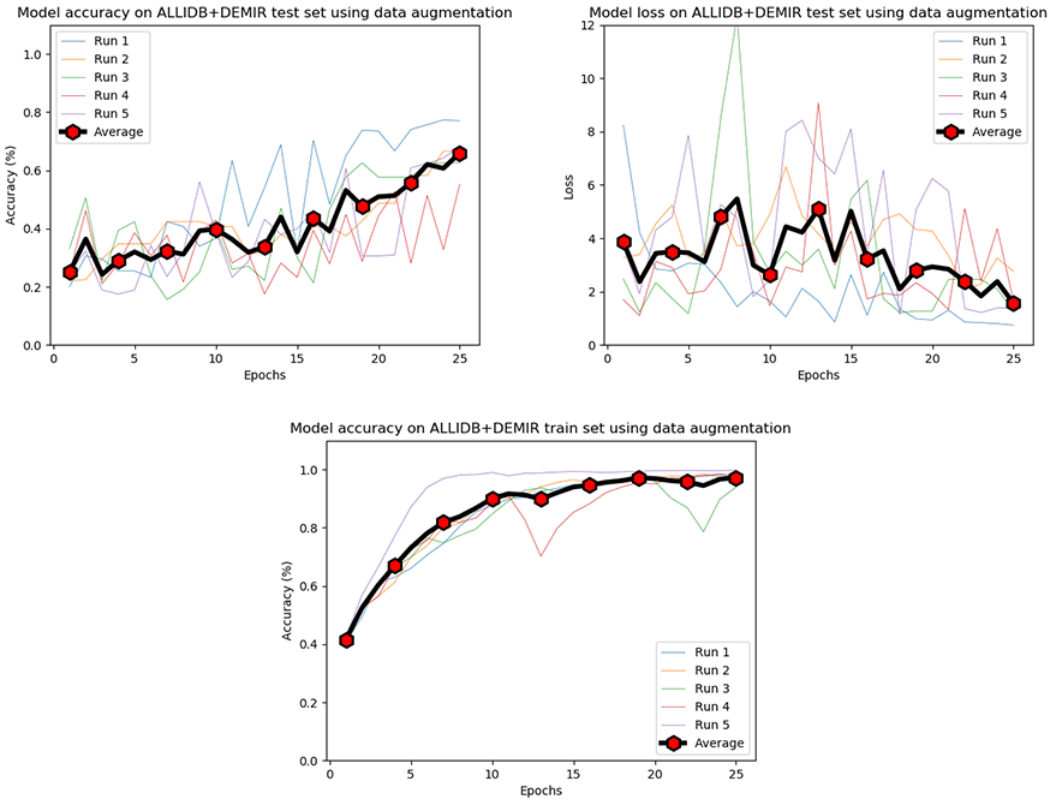


Figure A.20: Results of experiment C.6 (On ALL-IDB+DEMIR, with data augmentation) - The average result of the 5 runs after 25 epochs: (1) Accuracy on test set: **71.91%** (2) Loss value on test set: **1.556** (3) Accuracy on train set: **97.20%**

A.3.7 Experiment C.7 - Dropout and data augmentation

Last experiment is testing dropout with the data augmented dataset. As experiments A.7 and B.7, we see a little improvement compared to the respective experiments of using dropout without data augmentation. That demonstrates one more time that dropout needs to be use with a larger dataset and/or with larger images to counter overfitting. If we only make conclusion from our own experiments, we see that dropout is to avoid but this is generally not the case. Indeed, when having a large dataset with large images, using dropout allows the model to counter overfitting and then increase the performance of the model such as the reference article [29] that have used this configuration.

	Run 1	Run 2	Run 3	Run 4	Run 5	AVG
ACC-TEST	42.62%	22.4%	28.96%	34.42%	28.68%	31.41%
LOSS-TEST	1.35	2.76	1.88	1.36	2.02	1.874
ACC-TRAIN	57.44%	44.60%	55.19%	52.09%	59.63%	53.79%

Table A.21: Experiment C.7 (On ALL-IDB+DEMIR, with data augmentation and dropout=0.1) - Train set = 1464 images, Test set = 266 images - Detailed results of each run after 25 epochs. ACC-TEST is the accuracy result on test set, LOSS-TEST the value of cost function on test set and ACC-TRAIN the accuracy result on train set.

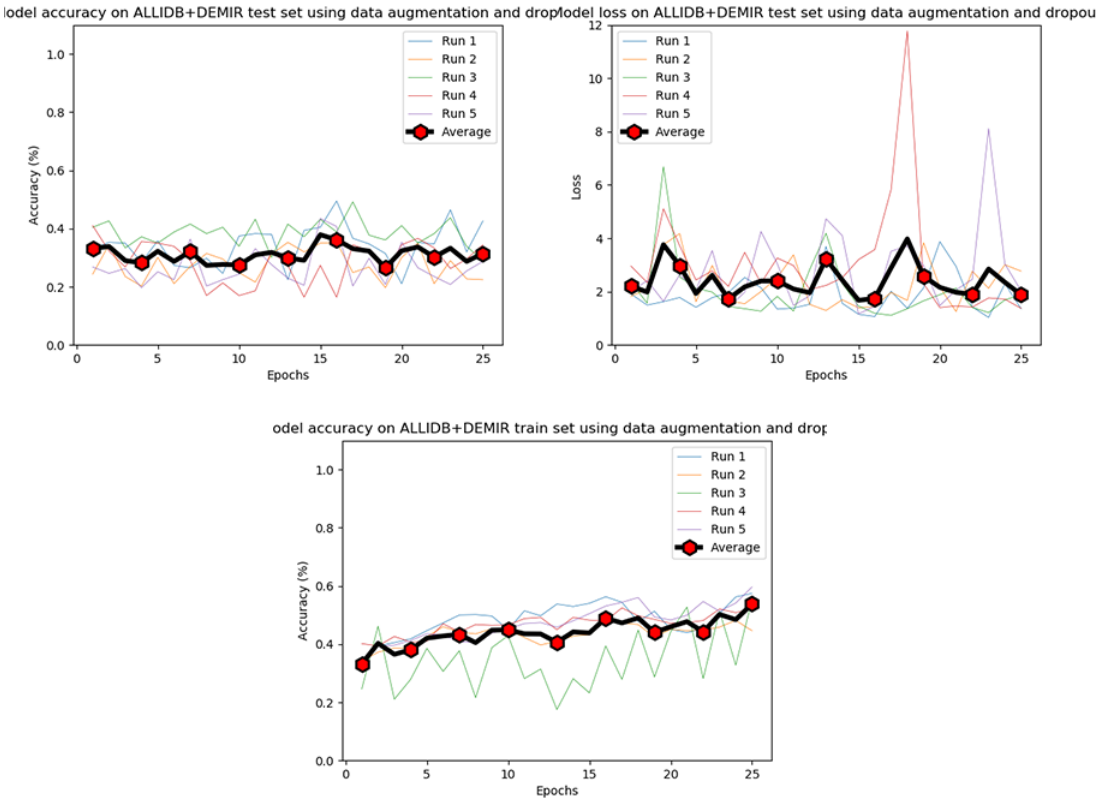


Figure A.21: Results of experiment C.7 (On ALL-IDB+DEMIR, with data augmentation and dropout=0.1) - The average result of the 5 runs after 25 epochs: (1) Accuracy on test set: **31.41%** (2) Loss value on test set: **1.87** (3) Accuracy on train set: **53.79%**