# Skin Cancer Detection Using Deep Learning Techniques

Álvaro González Jiménez

Final Degree Project

Supervisor Dr. Isabel A. Nepomuceno Chamorro


Universidad de Sevilla

May 2019

Publish on May 2019 by
Álvaro González Jiménez
https://github.com/alvarogonjim
alvgonjim1@alum.us.es

Template made by Pablo Trinidad
http://www.lsi.us.es/ trinidad/

Me, D. Álvaro González Jiménez  ID number 53585603L,

**I declare**

my authorship of the work presented in the memory of this final degree project that has the title:

*Skin Cancer Detection Using Deep Learning Techniques*

Which I sign,

Fdo. D. Álvaro González Jiménez
in Universidad de Sevilla
06/05/2019

# Contents

# List of Figures

# List of Tables

# Part I

# Introduction

# Motivation and problem definition

*In this chapter, we will present the problem that we are going to solve in the futures chapters. Furthermore, we will explain my motivation in order to choose this topic.*

## 1.1 Motivation and objectives

During my last year of degree I was wondering what I was going to work on my thesis. I knew I wanted to work on something related to health because I had always been interested in creating a product with value which would have an impact on society.

During my internship I was in one of the most important research centres in France, Naver Labs Europe, in the field of artificial intelligence. There I realized the potential of the Deep Learning technology that was emerging during these years. I could do something related to health that I had not worked previously during my studies.

As a responsible software engineer I will build a maintainable software following the rules and standards that I learned during these years, as well as learn new emerging technologies. Although the result will not be as good as recent studies because of the available resources.

The main objective is to learn about this technology that I have never studied before and achieve a scalable prototype whose precision and quality will increase in the future using different techniques. We will compare the obtained results with other algorithms and/or other works (papers, studies and so on). Finally, we are going to expose the best model in a REST API so we can build an APP which sends a picture, taken from the camera, as an input to the API and it will return the predicted results to the user.

## 1.2 Definition of the problem

"Nowadays the melanoma of the skin represents 5.3% of all new cancer cases in the U.S. In 2018, it is estimated that there will be 91,270 new cases of melanoma of the skin and an estimated 9,320 people will die of this disease" [1]. A quickly prediction of this disease will help to reduce the number of die people. In fact, most of the people do not realize that they have melanoma of the skin so it will make harder the cure process.

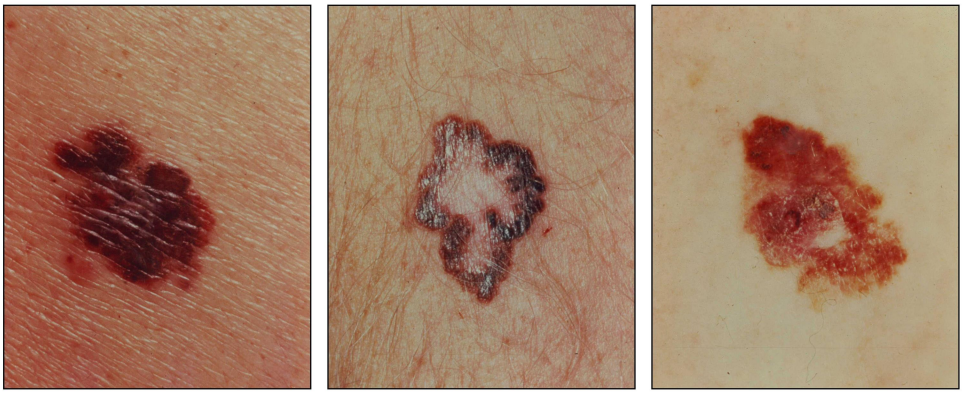


Figure 1.1: Examples of melanoma of the skin. [1]

As you can see in the figure §1.1 the melanoma presents a wide variety of characteristics, for example the pigmentation of the skin will be dark, the size is bigger than the normal pigmentation of the skin and the shape is irregular. This set of characteristics make a difficult task for human eye but an interesting research line using AI algorithms to detect the melanoma.

# AI History and State of the art

*In this chapter, we will explore a little bit about the Machine Learning world, the algorithms that people have been used for years to solve this and other types of problems. A brief summary about the current frameworks that we can use to solve this problem, theirs pros and cons. Finally we are going to explain the problems that we are going to solve.*

## 2.1 Artificial Intelligence History

Deep Learning (DL) has revolutionized industry after industry. People thinks that Deep Learning and Artificial Intelligence (AI) are synonyms, but these words are totally different. We can define the AI as *the automation of intellectual tasks normally performed by humans*. The AI started in the beginning of the 1957 when John McCarthy held the first academic conference on the subject, the AI machines were able to solve problems that were difficult for humans to solve.

One of the most important AI machines in the history is the Enigma machine built by Alan Turin at the end of World War II. This machine could crack in hours entire conversations and text made by the Nazis.

In the early years of AI, a lot of researchers believed that AI could be achieved by hard coding rules. The kind of AI is called symbolic AI and was useful in solving well-defined, logical problems but it was incapable of solving complex problems such as image recognition, object detection, object segmentation, language translation, and natural-language-understanding.

In order to understand the relationship among AI, ML and DL let's visualize them as concentric circles see in the figure §2.1.

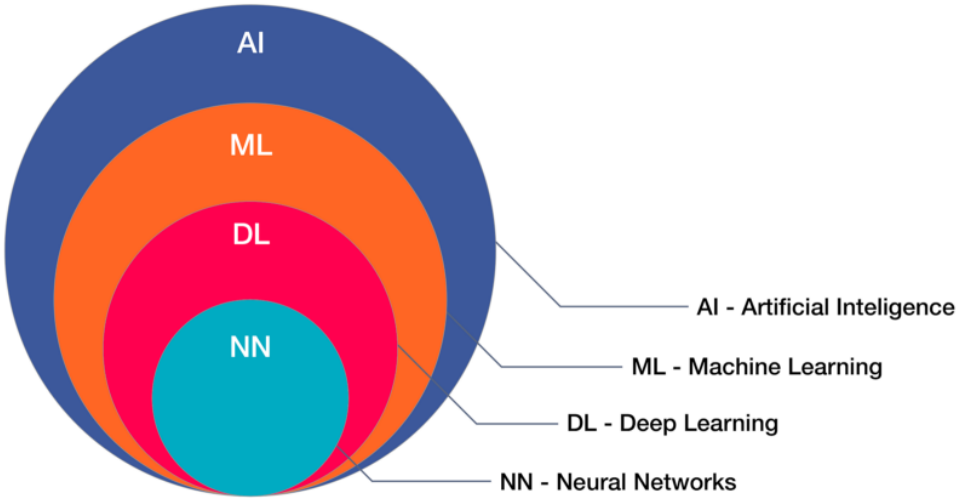



Figure 2.1: Representation AI, ML and DL [2]

The idea that came first (AI) and the largest one. Then machine learning (ML) which blossomed later, and finally DL which is driving today's AI explosion (fitting inside both).

### 2.1.1 Machine Learning

Machine Learning (ML) is a sub-field of AI and has become very popular in the last 10 years. This discipline uses mathematics and statistics with the power of the computer to build intelligent systems that can learn by themselves, identify patterns and they are able to make decision without human intervention.

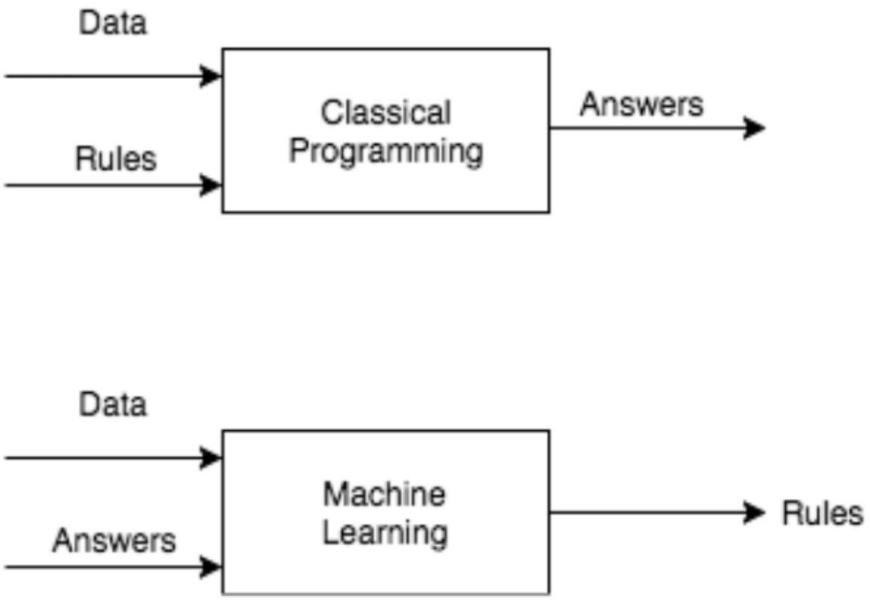



Figure 2.2: ML vs Traditional programming

At a high level, machine learning systems took at tons of data and come up with rules to predict outcomes for unseen data as you can see in the figure §2.2. Some examples of machine learning systems in the real life are:

- Google Photos uses a specific form of machine learning for grouping photos.
- Recommendations systems which are family of ML algorithms. We can see this kind of recommendations systems in many famous apps like Spotify (music), Netflix (movies) and Amazon (products).

### 2.1.2 Classical Machine Learning

The first methods came from pure statistics in the '50s. They solved formal math tasks — searching for patterns in numbers, evaluating the proximity of data points, and calculating vectors' directions [29].

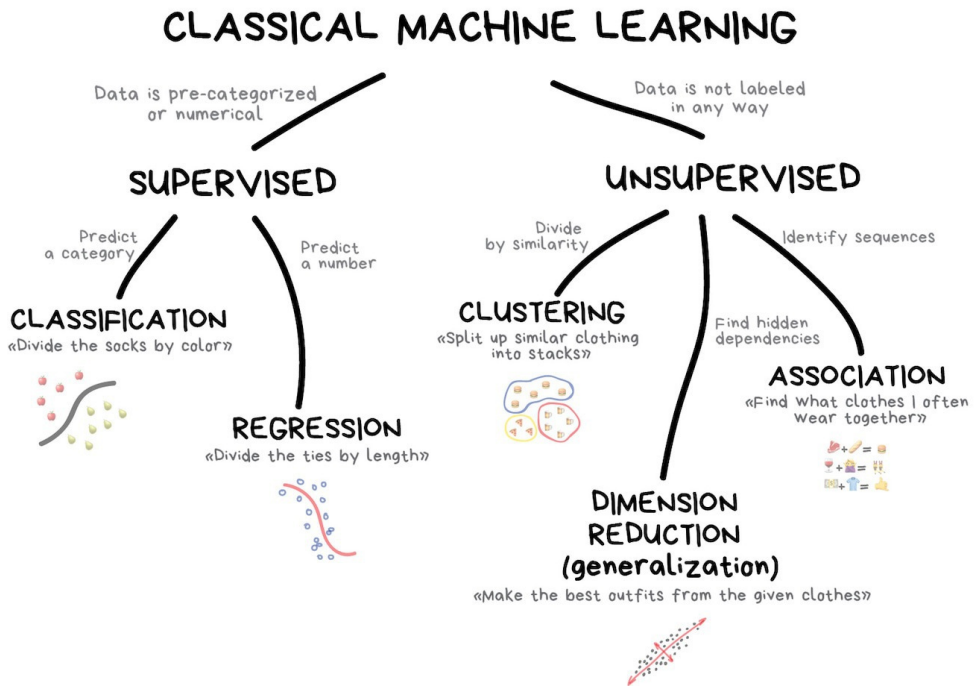



Figure 2.3: Classical Machine Learning algorithms [29]

As you can see in the image §2.3 there are two main groups in the classical machine learning, the first one called supervised because we teach the algorithm. The data that we use in this type of learning is labelled and they try to predict a category (classification) or a number (regression) from the previous data and experience.

On the other hand we have unsupervised learning. It is very hard to find labelled data, in the internet all the data that we can find are not labelled so we can spend days putting a label to the data one by one or we can use this types of algorithms to find relations and categories by themselves.

The most important algorithms in the unsupervised learning are clustering to divide objects based on unknown features. Dimensionality reduction that assembles specific features into more high level ones and associated rule learning that looks for patterns in the order's stream [29].

### 2.1.3 Ensemble Methods

This technique are being used because of the accuracy that they can produce. The idea is quite simple to understand, just take some algorithms (regression, decision trees, kNN, Naive Bayes and so on) and put it together to get a better result.

You'll get even better results if you take the most unstable algorithms that are predicting completely different results on small noise in input data [**?** ]

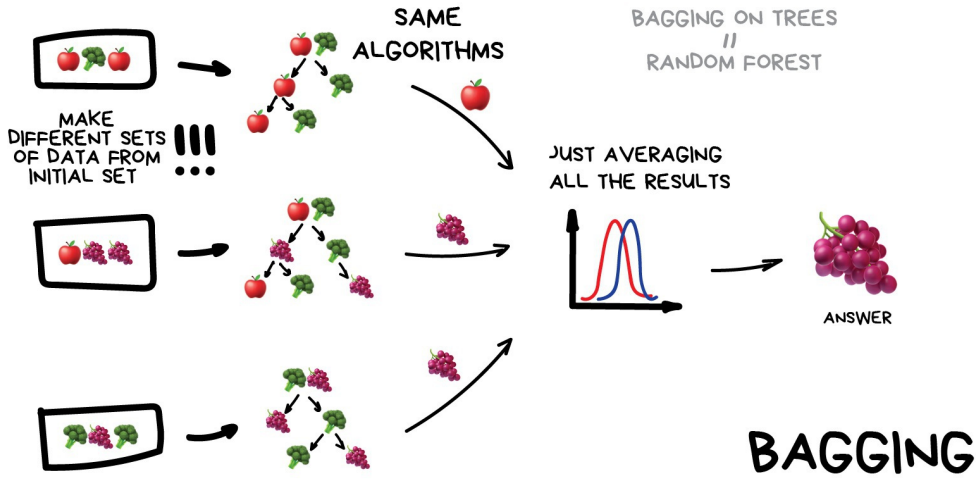



Figure 2.4: Explanation of the bagging algorithm [29]

The previous image §2.4 represents one of the most famous ensemble methods, the random forest, which is simply bagging on the decision trees. Although you have other types of ensemble methods like Stacking or Boosting.

### 2.1.4 Deep Learning

Deep Learning is a sub-field of ML, that uses specific algorithms to extract features from images. They are based on neuronal networks whose inputs are each pixels of the image and they predict a label as an output.

For example, we can use the image of a number, the neuronal network will identify the borders of the number passing each pixel through the neuronal network's layers and it will determine which number is the image.

The use of DL has grown tremendously in the last few years with the rise of GPUs, big data, cloud providers (Amazon Web Services) and frameworks such as Torch, Tensorflow or Pytorch.

### 2.1.5 Reinforcement Learning

Reinforcement Learning is a sub-field of machine learning which addresses the problem of automatic learning of optimal decisions over time. This is a general and common problem studied in many scientific and engineering fields.[9]

Reinforcement Learning (RL) is the third camp and lays somewhere in between full supervision and a complete lack of predefined labels. On the one hand, it uses many well-established methods of supervised learning such as deep neural networks for function approximation, stochastic gradient descent, and back-propagation, to learn data representation. On the other hand, it usually applies them in a different way. We need an agent that takes actions in some environment.

A good example of this is a robot mouse in a maze as you can see in the figure §2.5. Its environment is a maze with food at some points and electricity at others. The robot mouse can take actions such as turn left/right and move forward. Finally, at every moment it can observe the full state of the maze to make a decision about the actions it may take.

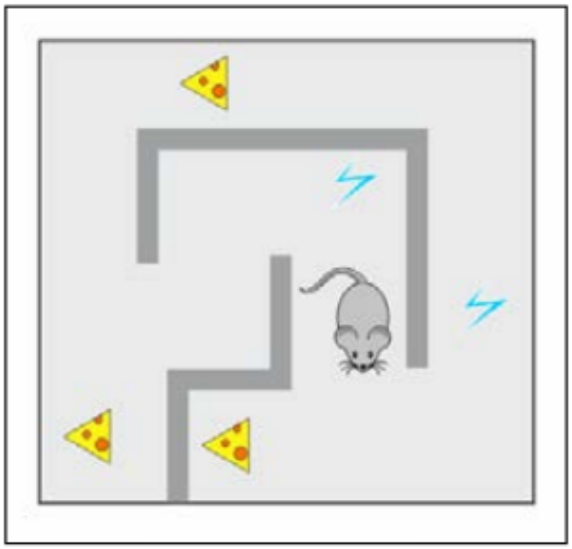


Figure 2.5: Example of the Robot Mouse scenario [9]

It is trying to find as much food as possible, while avoiding an electric shock whenever possible. These food and electricity signals stand as a reward given to the agent by the environment as additional feedback about the agent's actions.

## 2.2 STATE OF THE ART

### 2.2.1 Neuronal network

A neuronal network is a mathematical model that try to represents how our mind works. The first mathematical model was presented in 1943 called Perceptron this model could do several simple tasks [3].

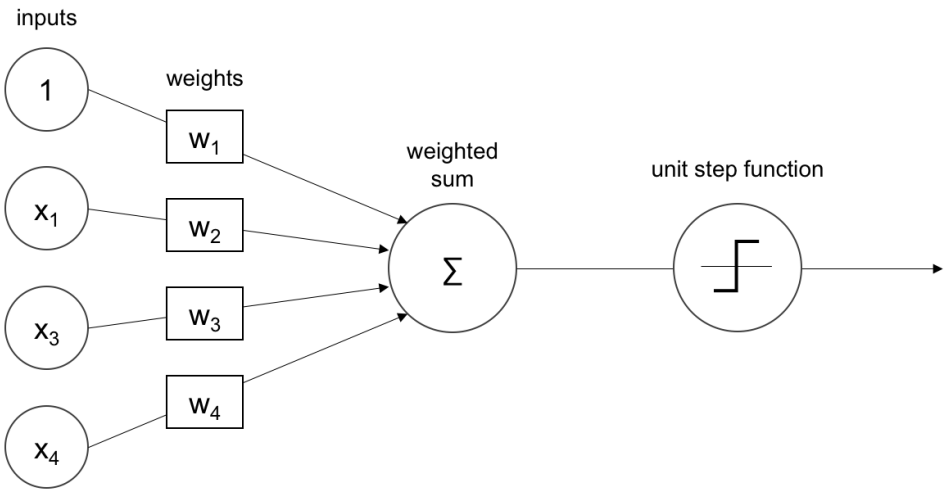



Figure 2.6: Example of a perceptron [5]

The figure §2.6 represents a perceptron which has a set of inputs that are going to be multiplied by their weights. Then the perceptron will perform a weighted summation to produce an output. [4] Finally the output of the perceptron can be passed through an activation function or transfer function.[5] The process that the weight of the inputs changes in order to improve the output of the perceptron is called training.

An Artificial Neuronal Network (ANN) is a collection of perceptrons as you can see in the figure §2.7 and activation functions. The perceptrons are connected to form hidden layers or units. The hidden units form the non-linear basis that maps the input layers to output layers in a lower-dimensional space, which is also called artificial neural networks. ANN is a map from input to output. The map is computed by weighted addition of the inputs with biases. The values of weight and bias values along with the architecture are called model.

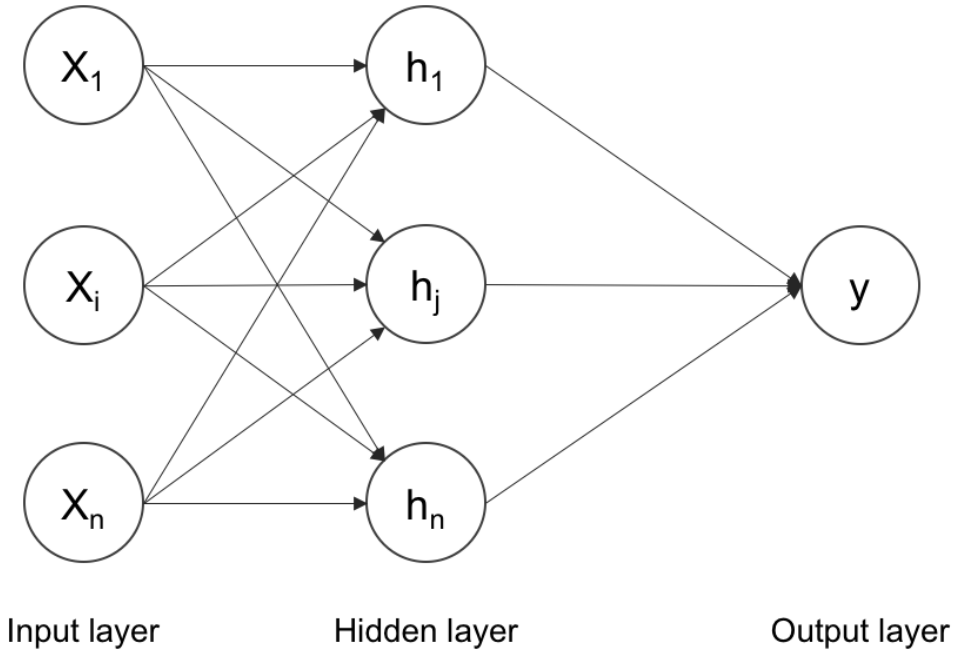



Figure 2.7: Example of an Artificial Neuronal Network (ANN) [5]

The ANN contains several parameters to optimize. The figure §2.8 represents the procedure of updating the weights, this procedure is called back-propagation. The weights are updated from backward based on the error calculated. The procedure to minimize the error is called optimization.

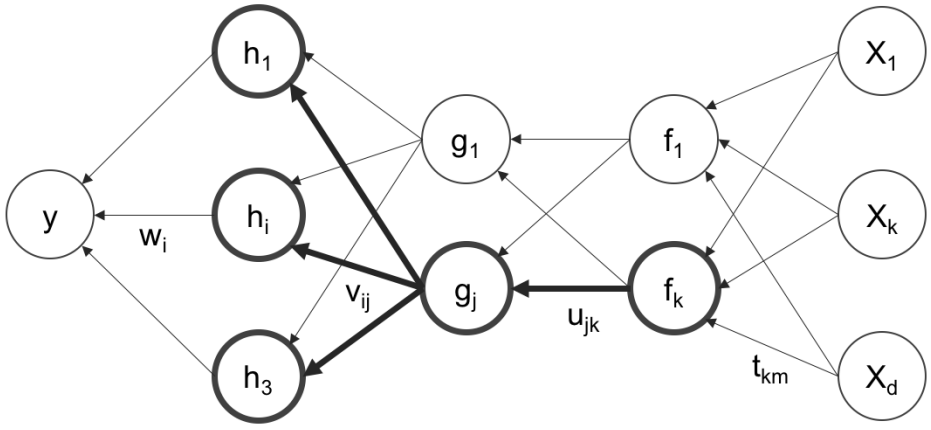



Figure 2.8: Example of back-propagation procedure [5]

A few different ANN models are available among them the Convolutional Neural

Network (CNN) and Recurrent Neural Network (RNN). They have made some revolutionary improvements in the data analysis field.

### 2.2.2 Unsupervised Pretrained Networks

#### Autoencoders

The figure §2.9 represents an autoencoder. It is an architecture for unsupervised learning. The main purpose of this architecture is to reduce the dimensionality of the dataset and it learns directly from the input data. The structure is very simple, the input layer followed by the bottleneck where the encoder and decoder operation are done and finally the output layer that contains the same number of units as the input layer does [21].

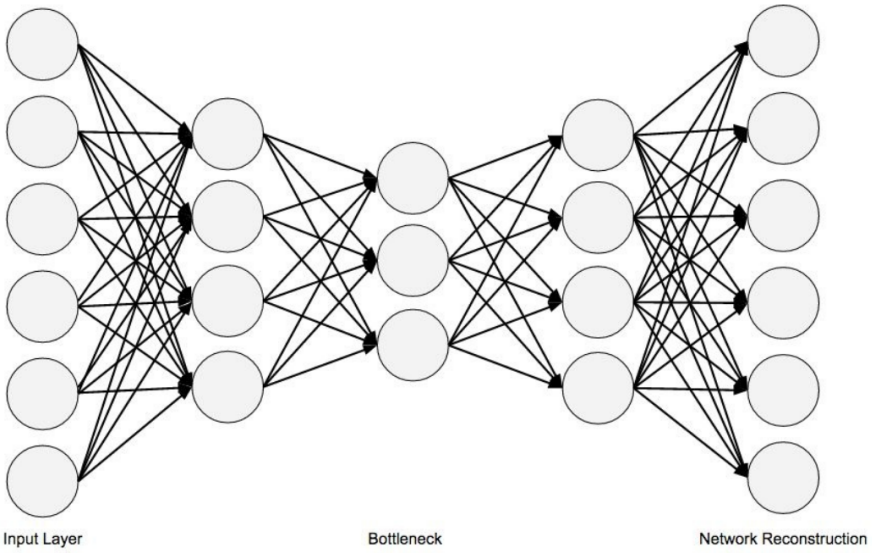



Figure 2.9: Example of Autoencoder [21]

#### Deep Belief Networks

Deep Belief Networks (DBNs) are composed of layer of Restricted Boltzmann Machines (RBMs).
A RBM is a neuronal network which can be useful for different kind of problems like dimensionality reduction, classification, regression, collaborative filtering, feature learning and topic modeling. The RBM is composed by two layers (visible layer and hidden layer). The nodes in the layer are not interconnected and each node starts making stochastic decisions about whether to transmit that input or not, this is the main

difference between the autoencoder and the RBM.

In unsupervised learning we use RBMs to extract high-level features from the input data. We discovered that if we let to RBMs to extract this high-level features it progressively can combine non-linear functions to extract more complex data, to understand better how it works we can see an example with MNIST digits [21].

### 2.2.3 Generative Adversarial Networks

The Generative Adversarial Networks is an example of unsupervised learning architecture where we train two models in parallel. Nowadays GANs architecture become really famous to create new images based on other images as you can see in the figure §2.10 [21].

Figure 2.10: Example of a GAN with The Mona Lisa painting [22]

GAN is composed by two players, one player (the generator) generates samples that are similar to the training data without having access to that data and the second player (the discriminator) examine that samples and determine that data is real or fake [22].

### 2.2.4 Convolutional Neuronal Network (CNN)

If we want to use a ANN for images the size of the network will be very large because of the number of neurons analysis will results an overfitting. Moreover in the

case that we want to use big resolution images the size of the network will be huge [5].

The convolutional neuronal network (CNN) is an advance ANN, which allows the network to extract local as well as global features from the data, enhancing the decision-making procedure of the network[6]. As you can see in the figure §2.11 a CNN typically has convolutional layers interspersed with pooling (or sub-sampling) layers and then followed by fully connected layers as in a standard multi-layer neural network [7].

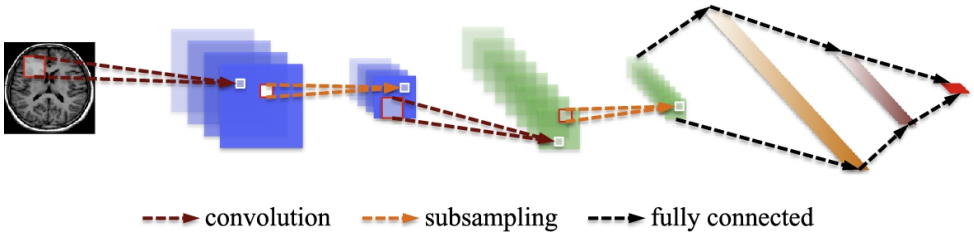



Figure 2.11: An architecture of a convolutional neural network

**Convolution**

A convolution is defined as a mathematical operation describing a rule for how to merge two sets of information [8].

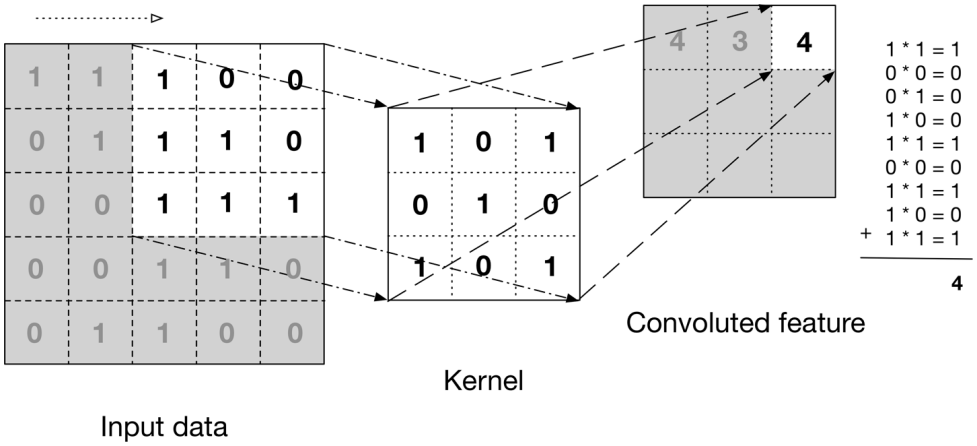



Figure 2.12: The convolution operation [5]

The figure §2.12 illustrate how the kernel is slid across the input data to produce the convoluted feature (output) data.

### Pooling

Usually after the convolution layer we will include a pooling layer to reduce the spatial size [8] (height and width) for the next layer. The most common pooling operation is max-pooling that reduces the size by a factor of $n^2$ [7] but exists other pooling operations like average-pooling, winner-takes-all pooling [10] and stochastic pooling [11]. The figure §2.13 represents the max-pooling and the average-pooling operation for a better understanding.

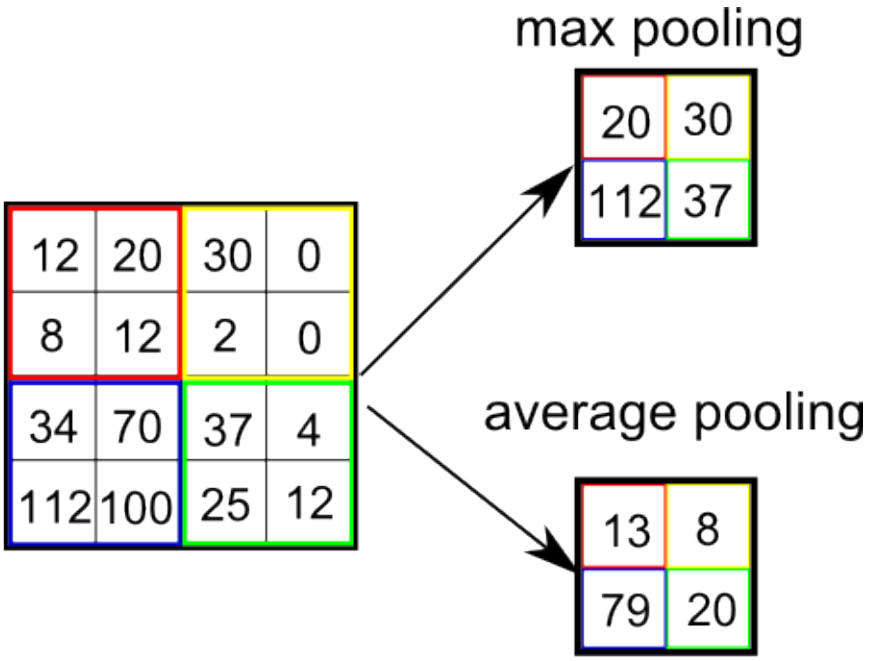



Figure 2.13: The max-pooling and average-pooling operation [26]

### Fully Connected Layers

We use this layer to compute class scores that we will use as output of the network. The dimension of the final output will be [1 x 1 x N] where N is the number of classes [8]. For example in our case the number of classes will be 2 (malignant or benign).

## 2.3 Software

There are many frameworks to resolve this problem. We are going to explain some for Python and compare them in order to choose the best framework to resolve our problem.

### 2.3.1 Tensorflow

Tensorflow was created by the IA Google Team. They followed the same structure that Theano library, the engine was written in C/C++ to increase the speed [13] It support the CPU and GPU operations and you can run with multiple GPU to optimize your model [15]. Furthermore this library include a visualization of your model called Tensorboard so you are able to see how is training your model for debugging and optimization. Tensorflow is the most famous library for Machine Learning (ML) as you can see in the next figure §2.14 works but, it's hard to learn and understand how it works in the beginning if we don't have any other experience in ML.

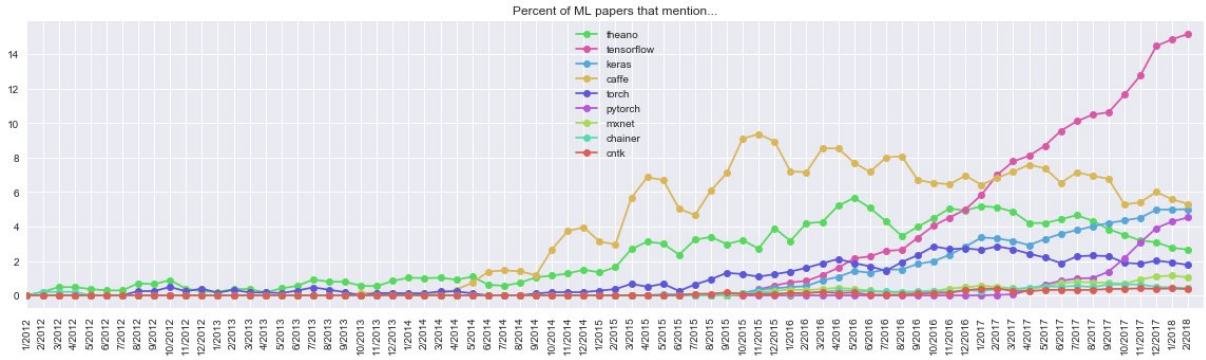



Figure 2.14: Percent of mentions in ML papers [12]

### 2.3.2 Theano

Maintained by Montréal University group [14] they were the pioneered to use a computational graph that will be use in Tensorflow project [13]. In case for large models Theano will take long time to compute it, also it doesn't support multiple GPU. The error messages can be unhelpful so it will be hard for debugging tasks.

### 2.3.3 Keras

Keras is an easy-to-use Python library [14] that sits atop Tensorflow and Theano so it took the advantages of both. It has an intuitive and simple API so you can write a model with a few lines of code. Keras is not really flexible and has some problems to use the multi-gpu.

### 2.3.4 Pytorch

The Python version of Torch called Pytorch is an open source project of Facebook created in January 2017. PyTorch has quickly become the favourite among machine learning researchers, because it allows certain complex architectures to be built easily [13]. Furthermore the modularity of Pytorch because it's easy to pull someone's code and use it [14].

### 2.3.5 H2O

It's an open source software platform which core is coded in Java [19] H20 is feature rich, ease of use and it's known for it's R and Spark integration but it provides support to other languages [20]. H20 counts with other plugins to create an easy app after train your model (Shinny App).

### 2.3.6 DL4J

DL4J is a JVM-based, industry-focused, commercially supported, distributed deep learning framework [13]. Python has many scientific environments to work with Deep Learning like Numpy or Theano but DL4J with Java and Scala has several advantages, Java and Scala are inherently faster than Python. Anything written in Python by itself, disregarding its reliance on Cython, will be slower. Furthermore the license under DL4J is Apache 2.0 License so anyone is free to make and patents derivative works based on Apache 2.0-licensed code.

## 2.4 Evaluate the model

There are many metrics and ways to evaluate the model and check the accuracy. Depends of the scenario we have to use a specific metric or other, there is not exist a metric for all the possibles scenarios. The objective of this section is to explain most of the famous metrics for a classification problem.

### 2.4.1 Confusion Matrix

This metric is the simplest because it's very easy to use and understand. Each column of the matrix represents the number of predictions for each class, while each row represents the instances in the actual class. One of the benefits of confusion matrices is

that they make it easier to see if the system is confusing two classes.

The associated values of a confusion matrix are:

1. **True Positives (TP):** This happens when we give a positive diagnosis and the disease is present. In our case, the patient has melanoma lesion and we diagnose melanoma.

2. **True Negative (TN):** This happens when we give a negative diagnosis and the disease is not present. In our case, the patient does not have melanoma and we diagnose not melanoma.

3. **False Positive (FP):** This happens when we give a positive diagnosis and the disease is not present. In our case, the patient does not have melanoma and we diagnose melanoma.

4. **False Negative (FP):** This happens when we give a negative diagnosis and the disease is present. In our case, the patient has melanoma and we diagnose not melanoma.

Let's see an example of the confusion matrix to understand how it works. The scenario is a dog vs cat classification and the results are the following:

| | | Actual Class | |
|---|---|---|---|
| | | Cat | Dog |
| Predicted Class | Cat | 5 | 2 |
| | Dog | 3 | 3 |

**Sensitive**

Measures the ability of a test to detect the condition when the condition is present.

$$Sensitive = TP/(TP + FN)$$

In our cat classificator the sensitive is:

$$Sensitive = 5/(5 + 3) = 0.625$$

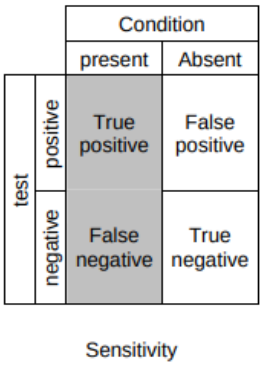



Figure 2.15: Calculate the Sensitive

**Specificity**

Measures the ability of a test to correctly exclude the condition (not detect the condition) when the condition is absent.

$$Specificity = TN/(TN + FP)$$

In our cat classificator the specificity is:

$$Specificity = 3/(3 + 2) = 0.6$$

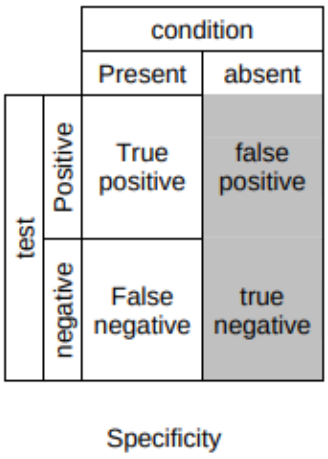



Figure 2.16: Calculate the Specificity

**Predictive value positive**

Measures the proportion of positives that correspond to the presence of the condition.

$$PVP = TP/(TP + FP)$$

In our cat classificator the predictive value positive is:

$$PVP = 5/(5+2) = 0.714$$

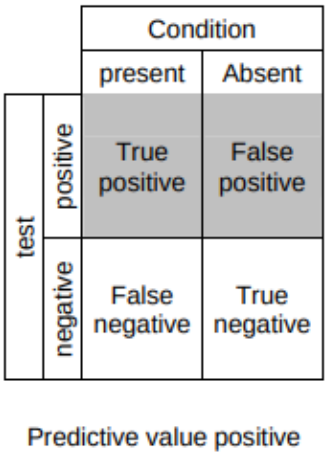



Figure 2.17: Calculate the Predictive Value Positive

**Negative predictive value**

Measures is the proportion of negatives that correspond to the absence of the condition.

$$NPV = TN/(TN+FN)$$

In our cat classificator the predictive value negative is:

$$NPV = 3/(3+3) = 0.5$$

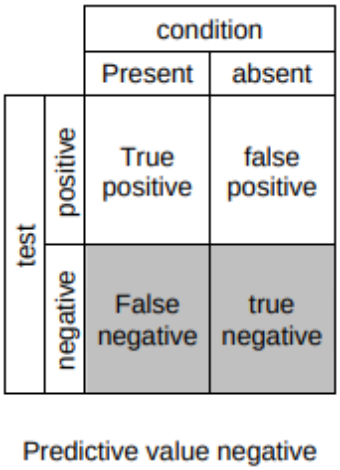



Figure 2.18: Calculate the Predictive Value Negative

### 2.4.2 The Area Under an ROC Curve

The area under the ROC curve ( AUC ) is a measure of how well a parameter can distinguish between two diagnostic groups (diseased/normal).

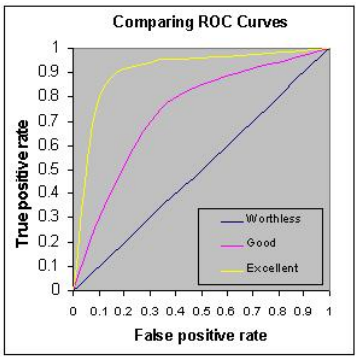



Figure 2.19: Examples of differents ROC curves [17]

It's a famous metric in the health scenario, the accuracy is measured by the area under the ROC curve. An area of 1 represents a perfect test; an area of .5 represents a worthless test [17]. A rough guide for classifying the accuracy of a diagnostic test is the traditional academic point system:

1. **.90 - 1** Excellent (A)
2. **.80 - .90** Good (B)
3. **.70 - .80** Fair (C)
4. **.60 - .70** Poor (D)
5. **.50 - .60** Fail (F)

## 2.5 Architectures

Neural network can be compared with lego blocks, where you can build almost any simplex to complex structures. Actually, most of the famous architectures have been discovered during the Large Scale Visual Recognition Challenge (ILSVRC), this challenge evaluates algorithms for object detection and image classification at large scale [18].

Depends of the task that we want to solve we have to use a type of architecture or other. The main types of tasks that computer vision can be categorised following the next figure §2.20.

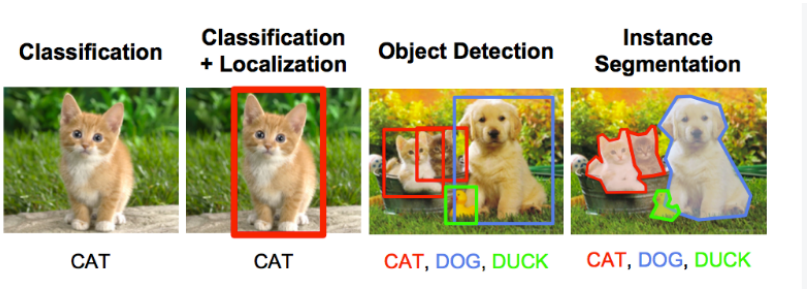



Figure 2.20: Examples of the main types of task in computer vision [18]

1. **Object Recognition / Classification.** In object recognition, the goal is to identify the class givin a raw image.
2. **Classification + Localisation.** - In this one we have to identify the class and find the location of that object in the raw image.
3. **Object Detection.** The task is to identify where in the image does the objects lies in.
4. **Image Segmentation.** It's a bit sophisticated task, where the objective is to map each pixel to its rightful class.

### 2.5.1 AlexNet

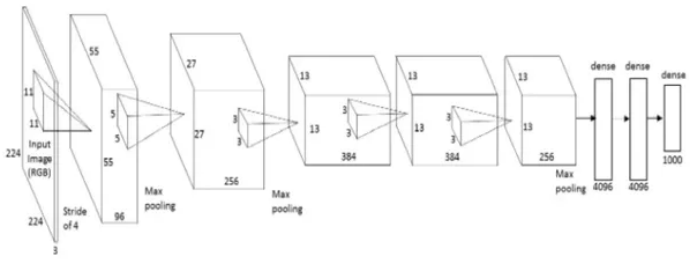



Figure 2.21: AlexNet Architecture [27]

It's the first deep architecture created by Geoffrey Hinton and his colleagues. When broken down, AlexNet seems like a simple architecture with convolutional and pooling layers one on top of the other, followed by fully connected layers at the top. This is

a very simple architecture, as you can see in the figure §2.21, which was conceptualised way back in 1980s [27].

### 2.5.2 VGG Net

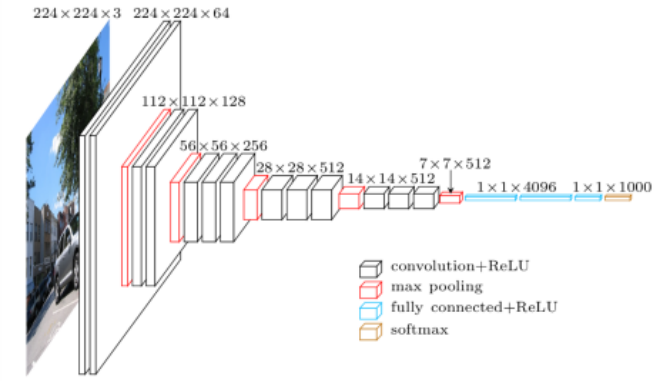



Figure 2.22: VGG Net Architecture [27]

The figure §2.22 represents an architecture which was introduced by the researchers at Visual Graphics Group at Oxford (hence the name VGG). The architecture is characterized by it's pyramidal shape, where the bottom layers which are closer to the image are wide, whereas the top layers are deep [27].

### 2.5.3 ResNet

Residual Networks (ResNet in short) is an architecture composed by residual blocks that can preserve good results through the addition of layers. They solved a problem of accuracy saturation when you add many layers to the neuronal network, this problem is called vanish gradient.

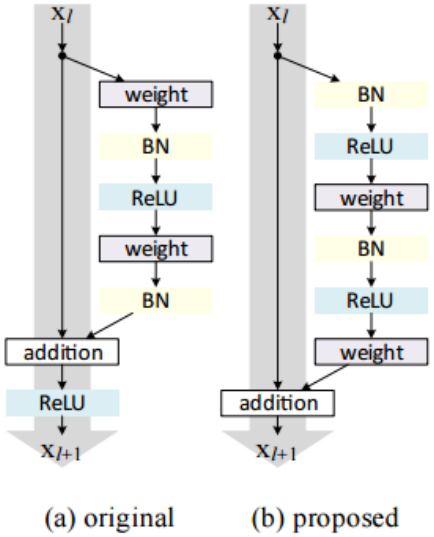



Figure 2.23: ResNet Architecture [27]

As you can see in the previous image §2.23 the main advantage of ResNet is that hundreds, even thousands of these residual layers can be used to create a network and then trained. This is a bit different from usual sequential networks, where you see that there is reduced performance upgrades as you increase the number of layers [27].

# Part II

# Project

# Dataset

*The dataset is one of the most important things in an artificial intelligence project, we are going to define the dataset, how we obtained this data and the preprocessing methodology in order to increase the number of the data.*

## 3.1 DATASET

### 3.1.1 Description

In order to set the feasibility of the project we have to look for a good dataset which contains a great deal of different data. In our case it's not a hard task to acquire the data, the ISIC institute exposed in 2018 more than 20.000 skin lesion images during a challenge. https://challenge2018.isic-archive.com/

Furthermore exist other famous datasets like *PH2Dataset* from the Oporto University which the number of images is much lower but we can use some Data Augmentation techniques to improve the results.

### 3.1.2 Acquisition of data

The ISIC Archive contains over 23k images of skin lesions, labelled as 'benign' or 'malignant'. The first option to download the entire ISIC archive is via the direct download button on their website, as you can see in the figure §3.1 which is the easiest and most comfortable way and doesn't always finish successfully for some reason. We suspect this is happening due to the large file size.

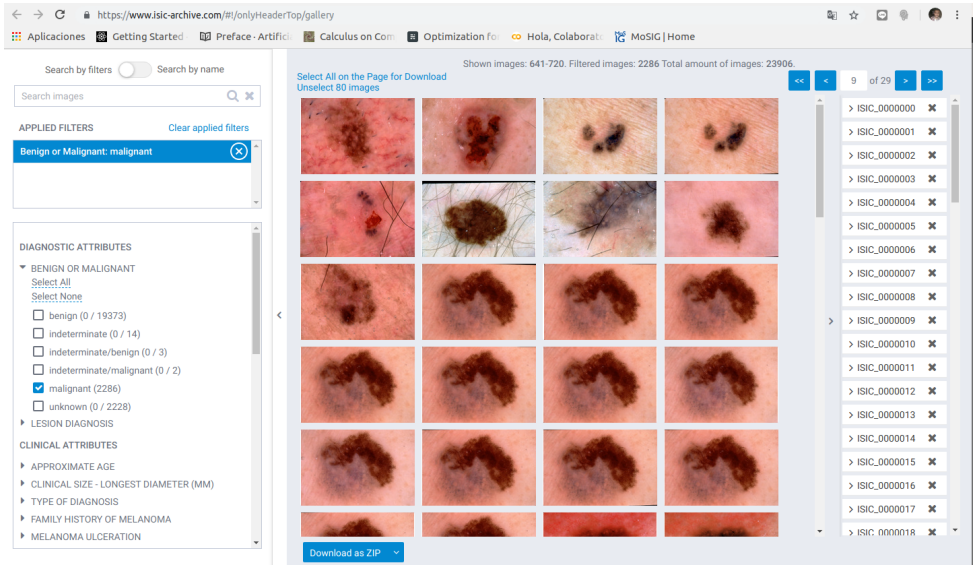



Figure 3.1: Download the data from the ISIC website

The second option is to download the small partitions of the archive, called 'datasets' one by one. Seems rather good if you plan to download the archive only a few times.

The third option is to download the archive using the Grider API provided in the website but seems infeasible.

Thankfully, Oren Talmor and Gal Avineri have created a script which we can use to download all the ISIC archive easily [23]. Furthermore we can filter and limit the number of images that we want to download.

After download the ISIC archive we divided in two separated folders:

- Training: The neuronal network will use the images in this folder to get training in every epochs.
- Test: The neuronal network will use the images in this folder to validate the training process and calculate the accuracy of the model in every epochs.

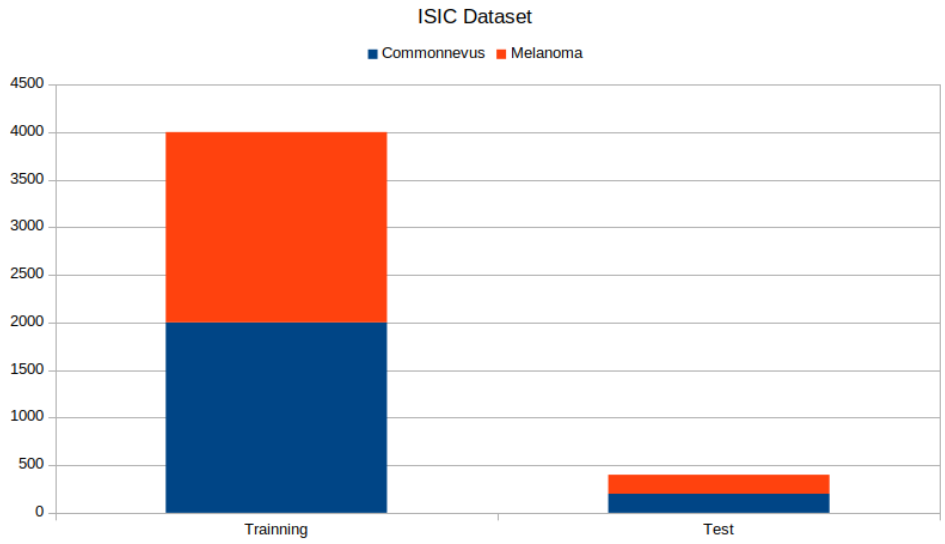



Figure 3.2: Data divided in training and test groups

You can see in the figure that we divided the dataset in the two groups, 4000 images will be used for training and 400 images for test purposes. Furthermore we have implemented some preprocessing techniques to increase the number of data but we will explain these techniques in the next section.

### 3.1.3 Image preprocessing

The image preprocessing is critical to subsequent steps and plays an important role in achieving a good performance. The source dermoscopy images provided by the

competition are obtained from real clinic environment, which means that there are a great number of man-made interferences. The objective of preprocessing is to improve the quality of the source images,and make the following parts run easily. In the next figure §3.3 you can see some techniques of image preprocessing (crop, rotation, scaling, add noise, change the colours and so on).

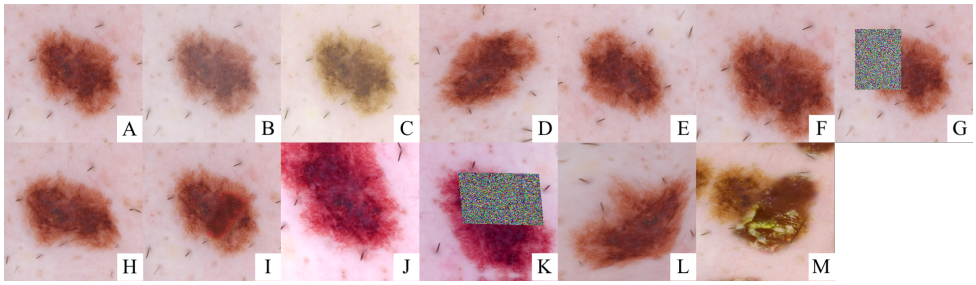



Figure 3.3: Examples of data augmentation. [24]

We are going to use some techniques for data augmentation. Data augmentation goal is to add new data points to the input space by modifying training images while preserving semantic information and target labels [24]. Thus, it is used to reduce overfitting. As we read from this paper, the results will increase if we use this techniques.

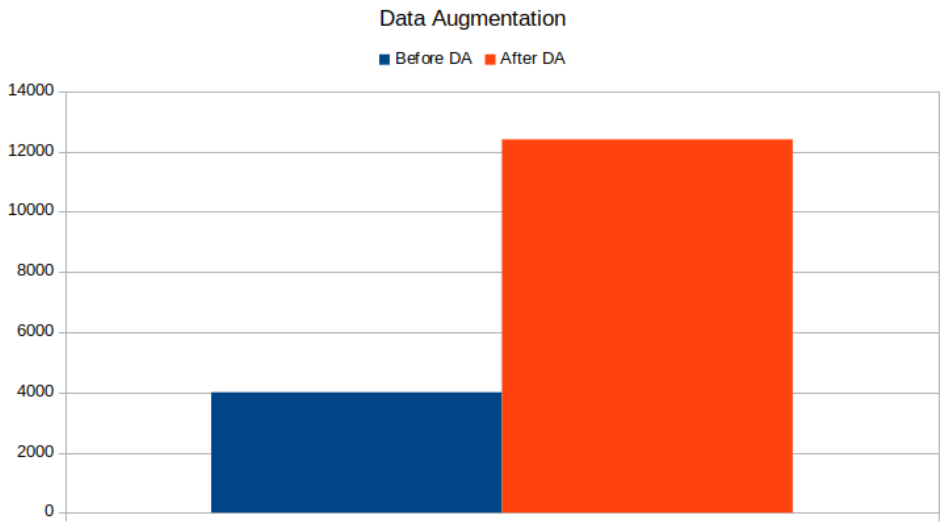



Figure 3.4: Number of data after apply data augmentation techniques

# Data analysis

*What is the best architecture? Which parameters should we change? How do we define a good model? In this chapter we are going to explain all these questions, we are going to expose the 3 different models that we are going to test and validate their results.*

## 4.1 Transfer Learning

In practice, very few people train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pretrain a ConvNet on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet either as an initialization or a fixed feature extractor for the task of interest [28].

This technique, initialize the neuronal network with a pretrained model (from the ImageNet challenge) then the weights of all the network will be frozen except of the two last fully connected layers that corresponds with the output of our problem (malignant or benign). This is the only layer which is trained during all the process.

In the next tables we are going to define the models that we are going to train and test for our project.

| 1º ResNet Model | |
|---|---|
| **Architecture** | ResNet |
| **Number of Layers** | 18 |
| **Optimizer** | Stochastic Gradient Descent |
| **Learning rate** | 0.001 |
| **Epochs** | 11 |

Table 4.1: Definition of the 1º model

| 2º ResNet Model | |
|---|---|
| **Architecture** | ResNet |
| **Number of Layers** | 50 |
| **Optimizer** | Stochastic Gradient Descent |
| **Learning rate** | 0.001 |
| **Epochs** | 21 |

Table 4.2: Definition of the 2º model

| 3º DenseNet Model | |
|---|---|
| **Architecture** | Densenet |
| **Number of Layers** | 101 |
| **Optimizer** | Stochastic Gradient Descent |
| **Learning rate** | 0.001 |
| **Epochs** | 21 |

Table 4.3: Definition of the 3º model

# Software analysis

*In this chapter we are going to define the requirements of our software solution. Furthermore, the risks and the initial costs of the project will be detailed.*

## 5.1 Requirements of the software

The main goal of this project is research about Deep Learning and how can we use this techniques to analyse skin cancer images, so we are not in the classical software lifecycle. However, we are going to provide an API that we will consume from an App to get the results of the predictions, this is just a prototype to show the power of this technology and how can we use in a real world.

| **REQ-001: Obtain the dataset and divide it** | |
|---|---|
| **Version** | 1.0 |
| **Author** | Álvaro González Jiménez |
| **Description** | The system must have all the images available and divided in two categories (trainning and tests). |
| **Priority** | High |
| **Status** | Done |
| **Comments** | - |

Table 5.1: REQ-001 Obtain the dataset and divide it

| **REQ-002: Preprocessing dataset images** | |
|---|---|
| **Version** | 1.0 |
| **Author** | Álvaro González Jiménez |
| **Description** | The system must to provide a set of tools to increase the number of images in the dataset in order to get better results. |
| **Priority** | High |
| **Status** | Done |
| **Comments** | - |

Table 5.2: REQ-002 Preprocessing dataset images

| REQ-003: Hyperparameter optimization or tuning | |
|---|---|
| **Version** | 1.0 |
| **Author** | Álvaro González Jiménez |
| **Description** | The system must to provide a set of variables called hyperparameters to select and change in order to optimize the models. |
| **Priority** | High |
| **Status** | Done |
| **Comments** | - |

Table 5.3: REQ-003 Hyperparameter optimization or tuning

| REQ-004: Data visualization | |
|---|---|
| **Version** | 1.0 |
| **Author** | Álvaro González Jiménez |
| **Description** | The system must to provide figures or tables in order to see how the models get trained and monitor their performance. |
| **Priority** | High |
| **Status** | Done |
| **Comments** | - |

Table 5.4: REQ-004 Data visualization

| REQ-005: Provide an API | |
|---|---|
| **Version** | 1.0 |
| **Author** | Álvaro González Jiménez |
| **Description** | The system must to provide an API that will receive an image as an input and will return the prediction of the image (common nevus or melanoma). |
| **Priority** | High |
| **Status** | Done |
| **Comments** | - |

Table 5.5: REQ-005 Provide an API

| REQ-006: Mobile client | |
|---|---|
| **Version** | 1.0 |
| **Author** | Álvaro González Jiménez |
| **Description** | Provide a mobile client (Android and IOs) that will take a picture, send it trought internet to the API and will return the prediction to the user. |
| **Priority** | High |
| **Status** | Done |
| **Comments** | - |

Table 5.6: REQ-006 Mobile client

## 5.2 Design and architecture

In this section we are going to define the architecture of our software, we will present the class diagram, the sequences diagrams, the definition of the API and the mocks up of the App.

### 5.2.1 Component UML

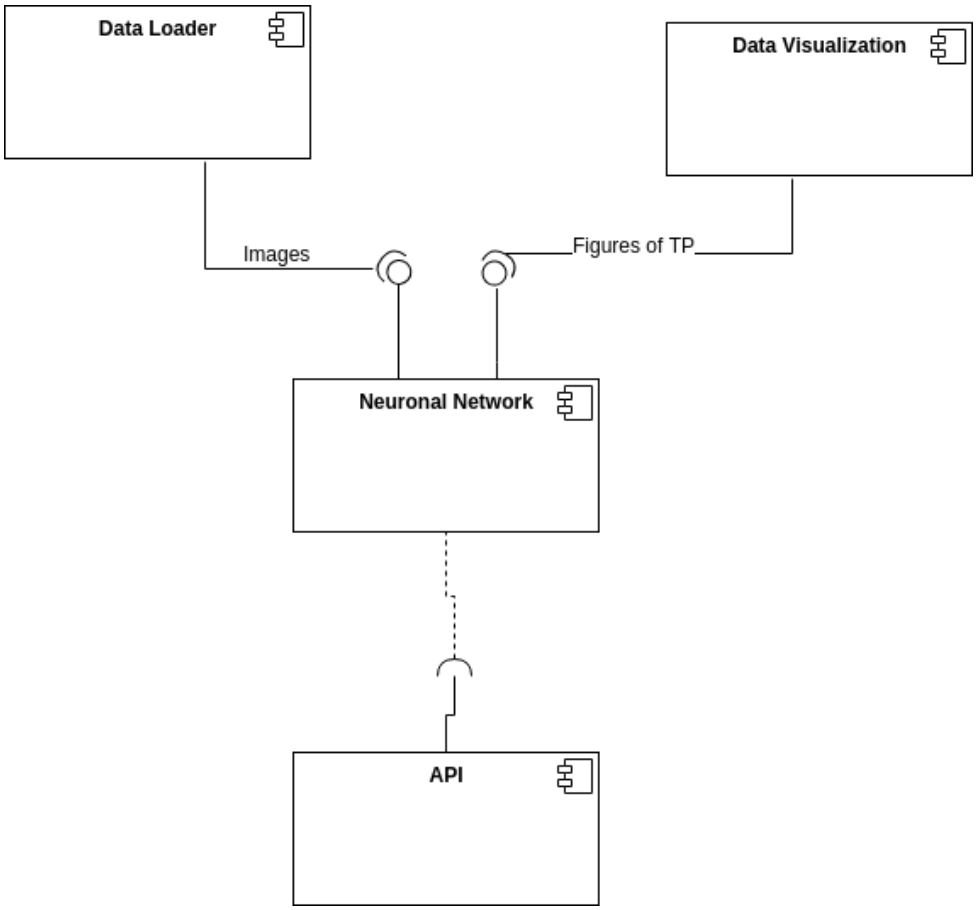



Figure 5.1: Component UML of the system.

The system is based on 3 main components, the Data Loader whose main objective is to load the images from the training and tests folders in an efficient way. The Data Visualization is the responsible of give the plots and figures of the training process to evaluate and monitor it. The Neuronal Network will train the model to predict the skin cancer images that we will use in the API component.

### 5.2.2 Use Cases UML

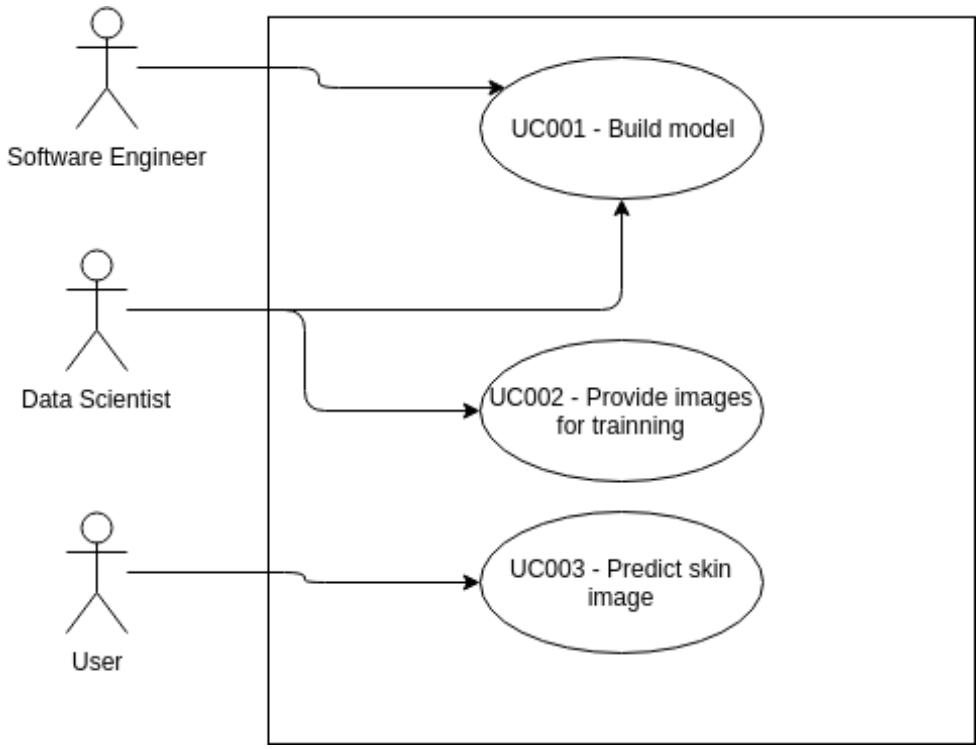



Figure 5.2: Use Cases UML.

**Definition of the use cases**

| UC001: Build model | |
|---|---|
| **Version** | 1.0 |
| **Author** | Álvaro González Jiménez |
| **Description** | The system must behave like in the following secuence diagram |
| **Priority** | High |
| **Status** | Done |
| **Comments** | - |

Table 5.7: UC001: Build model

| UC002: Provide images for trainning | |
|---|---|
| **Version** | 1.0 |
| **Author** | Álvaro González Jiménez |
| **Description** | The system must provide an easy process to store the images for trainning and provide them to the neuronal network throught the Data Loader. |
| **Priority** | High |
| **Status** | Done |
| **Comments** | - |

Table 5.8: UC002: Provide images for trainning

| UC003: Predict skin image | |
|---|---|
| **Version** | 1.0 |
| **Author** | Álvaro González Jiménez |
| **Description** | The system must behave like in the secuence diagram to predict a new image. |
| **Priority** | High |
| **Status** | Done |
| **Comments** | - |

Table 5.9: UC003: Predict skin image

### 5.2.3 API definition

In this subsection we are going to define the API that we will consume to predict the new images for the APP.

The API only has a endpoint which is the responsible to receive the image from the APP, evaluate it with the trained model and return the result through a JSON.

We can think that the endpoint should use the GET attribute because we want to get some information about an image. However, it is not possible to send an image with the GET attribute so we must use the POST attribute for that.

| Endpoint 1 - Predict an skin image | |
|---|---|
| **URI** | 1.0 |
| **HTTP Attribute** | Álvaro González Jiménez |
| **Content-Type** | Application/x-www-form-urlencoded |
| **Input key** | "image" |
| **Input key** | The image which we want to get the information about. |
| **Type of answer** | JSON |
| **Example of anwer** | "class": "melanoma", "probability": 0.823556 |

Table 5.10: Endpoint 1 - Predict an skin image

### 5.2.4 APP Prototype

As we have described in the previous section we are going to build an APP that consumes the services of the API, this is just a prototype for research purposes. However, we can include more features to this APP and sell it as a product or SAAS.

The APP must work in the most important platforms, Android and IOs. Furthermore, we have to be able to take a picture from the camera or get it from the gallery. We need a simple APP so we will implement one screen with the input data and show the results on the same. You can see in the next figures §5.3 §5.4 the mobile application that we are going to build.

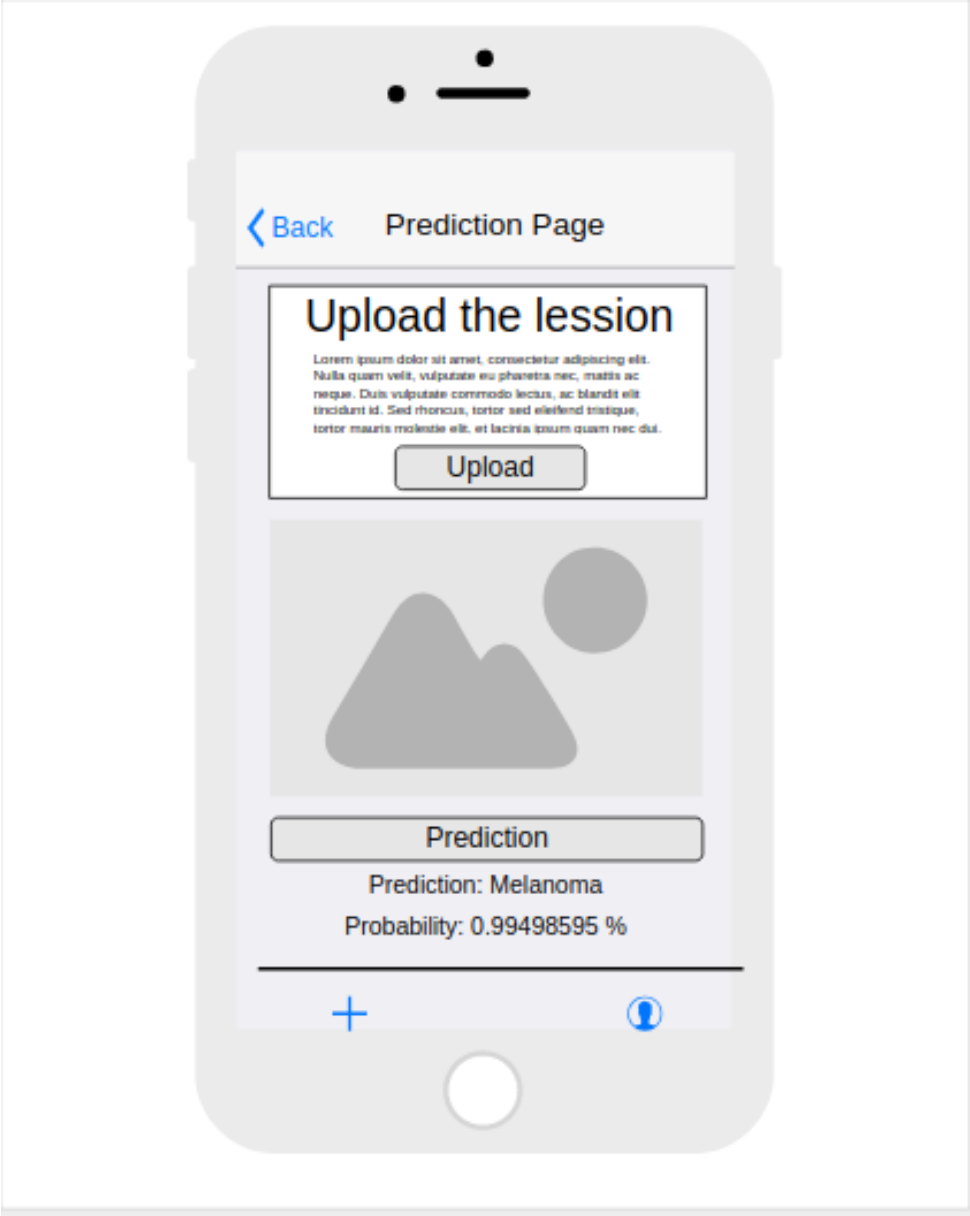



Figure 5.3: Mockup of the APP.

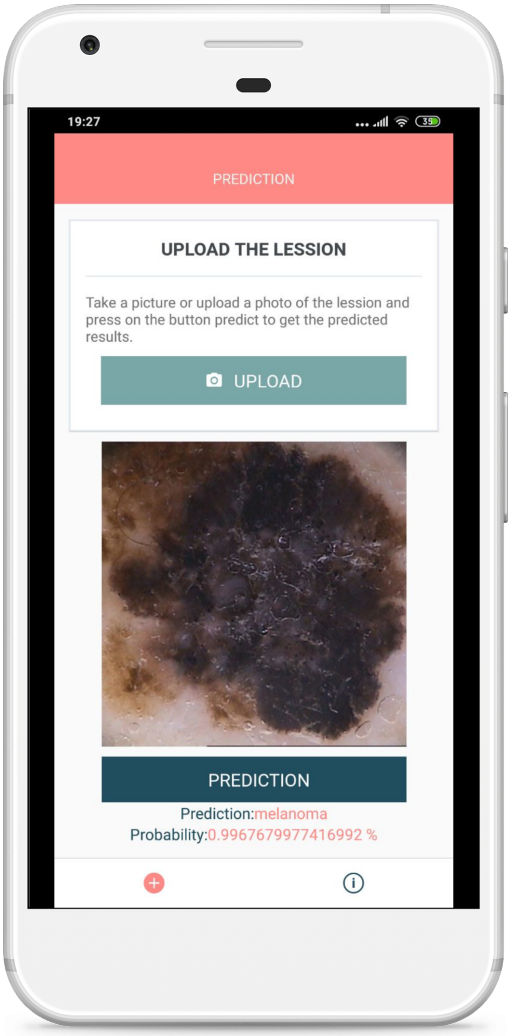



Figure 5.4: Final version of the APP.

## 5.3 PLANNIFICATION

We are going to follow the structure of a classical AI projects wich contains the following steps:

1. Obtain the data

2. Cleaning and preprocessing the data
3. Build a model
4. Evaluate it and tuning it to improve the results
5. Deploy the model or expose it in an API

We will define the structure of the project in the EDT (next section).

### 5.3.1 Resources available

**Hardware and Software**

The hardware is one of the most important thing in an AI project, it will determine the time to get the results of the model. Nowadays GPUs provide a great deal of speed in calculus and mathematics operations. In our case we haven't got the machines with GPUs, but we can use one of the famous GPUs clouds (AWS, Google Cloud, Paperspace, Floydhub, etc).

There are plenty of GPUs clouds services if we compare them the best between hardware and price is Paperspace [25].

| Machine 1 - Laptop | |
|---|---|
| **Model** | Lenovo G710 |
| **CPU** | Intel Core i7-4702MQ 2.2 GHz, |
| **GPU** | NVIDIA GeForce 820m |
| **RAM Memory** | 8GB |
| **OS** | Ubuntu 18.04 |

Table 5.11: Machine 1 - Laptop

| Machine 2 - Google Collab | |
|---|---|
| **Model** | P5000 |
| **CPU** | Intel Xeon E5-2623 v4 |
| **GPU** | NVIDIA Quadro P5000 with 2560 CUDA cores. |
| **RAM Memory** | 30GB |
| **OS** | Ubuntu 18.04 |
| **Price** | 0 $/hour |

Table 5.12: Machine 2 - Google Collab

The machine 1 will be used to read the articles, write the documentation and train the base model because it hasn't got the enough resources and power to train the entire dataset, it will take more than 1 week to train the dataset.

The machine 2 will be used to train the models, modify them and get the best accuracy. The training time in this machine will take few hours.

The software that we will use in the machine 1 are:

1. Microsoft Office 360
2. Draw.io
3. TexMaker (Latex)
4. Python 3.5
5. Jupyter Notebook
6. Pytorch

The software that we will use in the machine 2 are:

1. Python 3.5
2. Jupyter Notebook
3. Pytorch

**People**

The only available person to make this project is myself cause this is a final degree project. My supervisor will guide me, give me advises to build a good project but she is not going to implement it.

| Person 1 - Ph.D. Isabel Nepomuceno Chamorro | |
|---|---|
| **Rol** | Ph.D. |
| **Organization** | Dpto. de Lenguaje y Sistemas Informaticos (LSI) |
| **Description** | Member of the LSI department, she will guide the student to build good project and review the documentation. |

Table 5.13: People 1 - Ph.D. Isabel Nepomuceno Chamorro

| Person 2 - Álvaro González Jiménez | |
|---|---|
| **Rol** | Software Engineer |
| **Organization** | Dpto. de Lenguaje y Sistemas Informaticos (LSI) |
| **Description** | Software engineer student, he has the reponsability to implement the software, train the models, and write the documentation of the project. |

Table 5.14: People 2 - Álvaro González Jiménez

### 5.3.2 Work Breakdown Structure

To achieve the goal of our projectl we must plan each of the elements of the system and interconnect them as if they were boxes, this technique we call mind map to see the information we have (in our case what tasks we have) you can see the result of our mind map in the figure §5.5, later in the following figure §5.6 we can observe a work breakdown structure (WBS) that organizes the team's work into manageable sections, besides this technique is effective to measure and monitor the time that are needed in each of the sections.

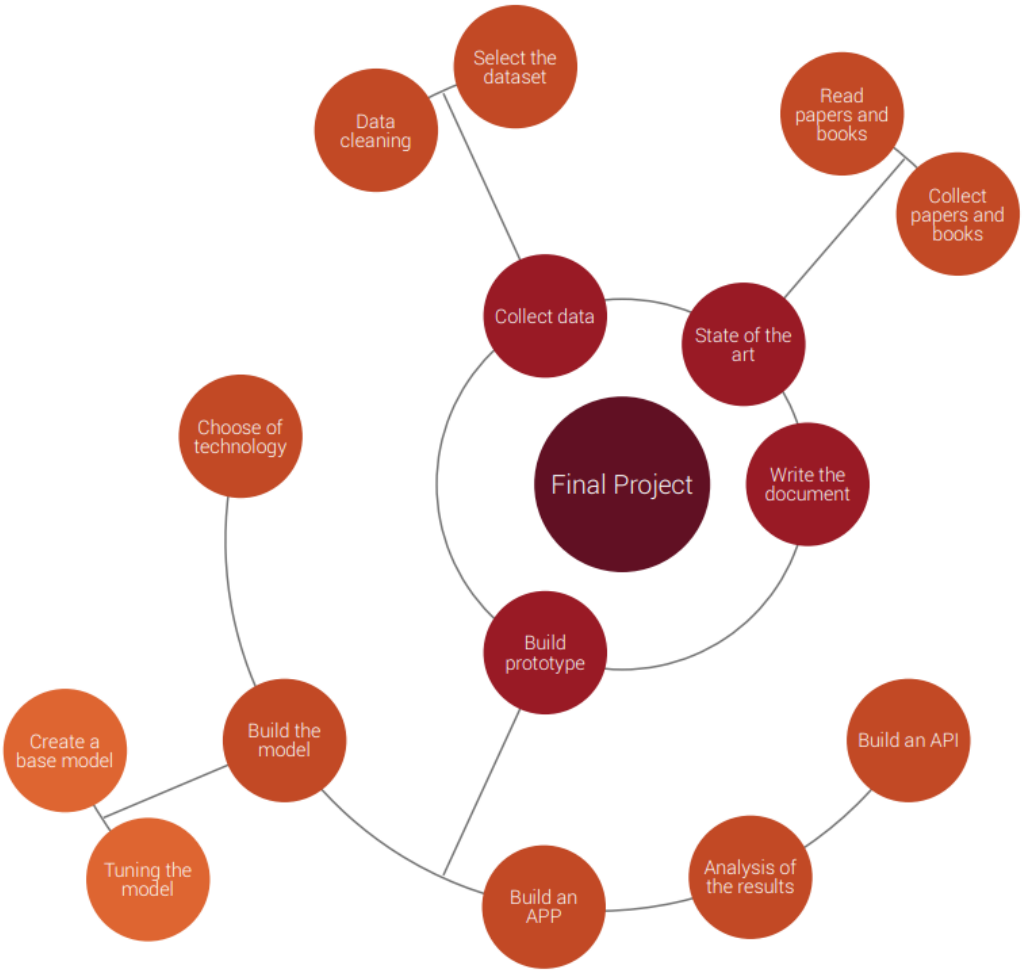



Figure 5.5: Mind map of the project

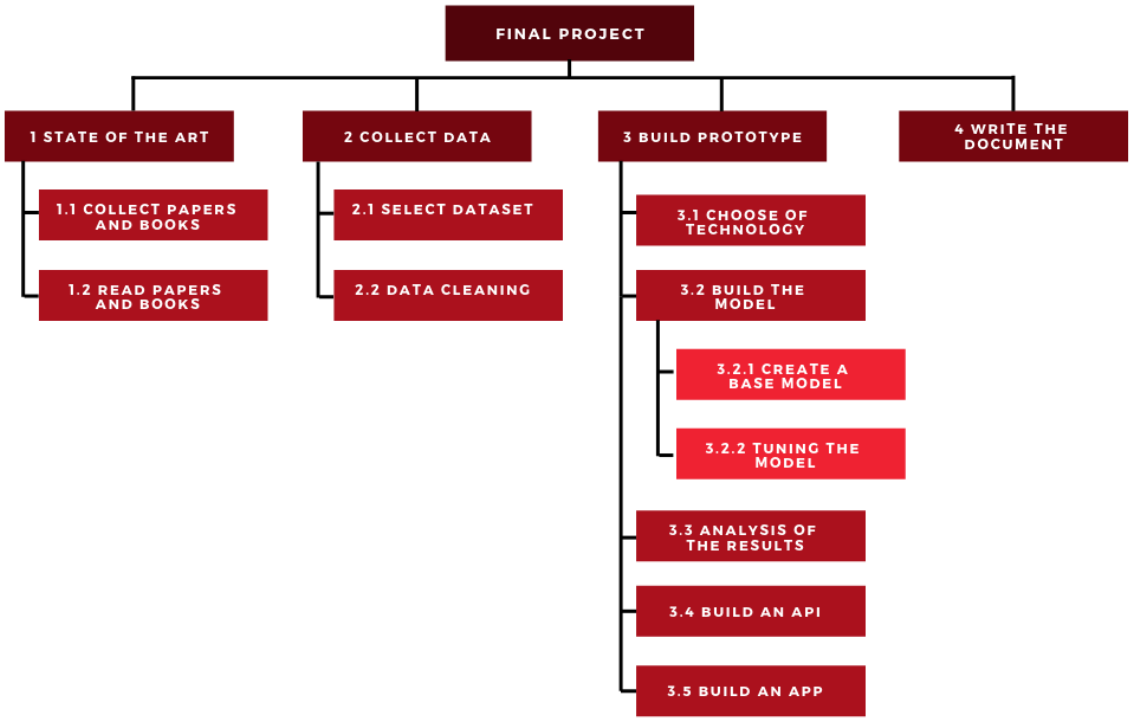



Figure 5.6: Work Breakdown Structure

| State of the art | |
|---|---|
| **ID** | 1 |
| **Description** | Research the current state of the art for DL |
| **Duration** | 60 hours |
| **People** | Álvaro González Jiménez |
| **Resources** | Machine 1 |

Table 5.15: WBS: 1. State of the art

| Collect papers and books | |
|---|---|
| **ID** | 1.1 |
| **Description** | Get available documentation for the state of the art |
| **Duration** | 20 hours |
| **People** | Álvaro González Jiménez |
| **Resources** | Machine 1 |

Table 5.16: WBS: 1.1 Collect papers and books

| Read papers and books | |
|---|---|
| **ID** | 1.2 |
| **Description** | Read, understand and learn the concepts of the documentation |
| **Duration** | 40 hours |
| **People** | Álvaro González Jiménez |
| **Resources** | Machine 1 |

Table 5.17: WBS: 1.2 Read papers and books

| Collect data | |
|---|---|
| **ID** | 2 |
| **Description** | Obtain the needed data to build the models |
| **Duration** | 10 hours |
| **People** | Álvaro González Jiménez |
| **Resources** | Machine 1 and Machine 2 |

Table 5.18: WBS: 2 Collect data

| Select the dataset | |
|---|---|
| **ID** | 2.1 |
| **Description** | Analyze and evaluate the obtained dataset in order to choose another one or improve it. |
| **Duration** | 2 hours |
| **People** | Álvaro González Jiménez |
| **Resources** | Machine 1 and Machine 2 |

Table 5.19: WBS: 2.1 Select the dataset

| Data cleaning | |
|---|---|
| **ID** | 2.2 |
| **Description** | Preprocesing the images and data augmentation |
| **Duration** | 8 hours |
| **People** | Álvaro González Jiménez |
| **Resources** | Machine 1 and Machine 2 |

Table 5.20: WBS: 2.1 Data cleaning

| Build the prototype | |
|---|---|
| **ID** | 3 |
| **Description** | Create the prototype to test the initial hypothesis |
| **Duration** | 250 hours |
| **People** | Álvaro González Jiménez |
| **Resources** | Machine 1 and Machine 2 |

Table 5.21: WBS: 3 Build the prototype

| Choice of technology | |
|---|---|
| **ID** | 3.1 |
| **Description** | Analysis of the current technologies, choose one and learning it to build the prototype. |
| **Duration** | 20 hours |
| **People** | Álvaro González Jiménez |
| **Resources** | Machine 1 |

Table 5.22: WBS: 3 Choice of technology

| Build the model | |
|---|---|
| **ID** | 3.2 |
| **Description** | Make the model which will classify the images |
| **Duration** | 100 hours |
| **People** | Álvaro González Jiménez |
| **Resources** | Machine 1 and Machine 2 |

Table 5.23: WBS: Build the model

| Create a base model | |
|---|---|
| **ID** | 3.2.1 |
| **Description** | Make an initial model that we will improve it to get better results. |
| **Duration** | 20 hours |
| **People** | Álvaro González Jiménez |
| **Resources** | Machine 1 and Machine 2 |

Table 5.24: WBS: Create a base model

| Tuning the model | |
|---|---|
| **ID** | 3.2.2 |
| **Description** | Try different inputs and train differents models to get the best model. |
| **Duration** | 80 hours |
| **People** | Álvaro González Jiménez |
| **Resources** | Machine 2 |

Table 5.25: WBS: Tuning the model

| Analysis of the results | |
|---|---|
| **ID** | 3.3 |
| **Description** | Get the results of the differents models, analyze base on the metrics that we decided and choose the best one. |
| **Duration** | 30 hours |
| **People** | Álvaro González Jiménez and Dr. Isabel Nepomuceno Chamorro |
| **Resources** | Machine 1 |

Table 5.26: WBS: Analysis of the results

| Build an API | |
|---|---|
| **ID** | 3.4 |
| **Description** | Make an API to get skin images and predict the label. |
| **Duration** | 50 hours |
| **People** | Álvaro González Jiménez |
| **Resources** | Machine 1 |

Table 5.27: WBS: Build an API

| Build an APP | |
|---|---|
| **ID** | 3.5 |
| **Description** | Make a small APP wich will take pictures or send pictures of the skin lession to the API in order to get the information about the lession. |
| **Duration** | 50 hours |
| **People** | Álvaro González Jiménez |
| **Resources** | Machine 1 |

Table 5.28: WBS: Build an APP

| Write the document | |
|---|---|
| **ID** | 4 |
| **Description** | Write all the documentation of the project in English and review it. |
| **Duration** | 50 hours |
| **People** | Álvaro González Jiménez and Dr. Isabel Nepomuceno Chamorro |
| **Resources** | Machine 1 |

Table 5.29: WBS: Write the document

## 5.4 Costs

In the people table we can see there are two types of participants in the project. Data scientist and project supervisors in order to estimate the cost of the project we will count the hour of each type employee and multiply it by the cost per hour associated to each type of employee.

It is worth mentioning that this is a final degree project, even if the author takes on the role of data scientist we are talking about a person without experience, knowledge and non graduated so the salary will be affected.

In case of the supervisor of the project, due to the fact that she is a PhD teacher with a great deal of experience her salary will be higher.

| People salary | |
|---|---|
| **Álvaro González** | 18.000 €/per year |
| **Isabel Nepomuceno** | 30.000 €/per year |

Table 5.30: People salary

Assuming a total of 1722 hours per year we will get the nexts results.

$$18000/1722 = 10,45 eur/hour$$

$$30000/1722 = 17,42 eur/hour$$

| Person | Estimate hours | Cost |
|---|---|---|
| Alvaro Gonzalez | 370 | 3866,5€ |
| Isabel Nepomuceno | 20 | 348,4€ |
| Total | 390 | 4214,9€ |

Secondly, we will calculate the amortization of hardware equipment in 4 years, which will give us an annual amortization quota. This annual amortization cost will be multiplied by the time of use of these equipment during the project, which will result in the cost associated with the project of each of the hardware equipment.

In the case of the second machine (server) hasn't got any amortization because we are not the owners of this hardware, we are going to pay as long as we use it.

The cost of the first machine is 900€ (C). We know that the porcentage that we can use for the amortization is 10% for a maximum of 4 years.

$$M = 900 eur * 0.1 * 4 = 360 eur$$

$$B = C - M = 900 eur - 360 eur = 540 eur$$

$$Annualrepayment = 540 eur/4 years = 135/year$$

The project will be done in 9 months so the cost associated is:

$$Annualrepayment = 135 eur * 0.75 year = 101,25 eur$$

In this section we will make the sum of all the costs associated with the project, which are mainly personnel costs and hardware costs. Knowing that the personnel costs amount to 4214,9€ and the hardware costs, to 101,25€. We have that the total cost of the project is 4214,9€ + 101,25€ = 4.113,65€.

| Summary of the cost of the project | |
|---|---|
| **People salary** | 4214,9 € |
| **Machine 1** | 101,25 € |
| **Total** | 4.113,65 € |

Table 5.31: Summary of the cost of the project

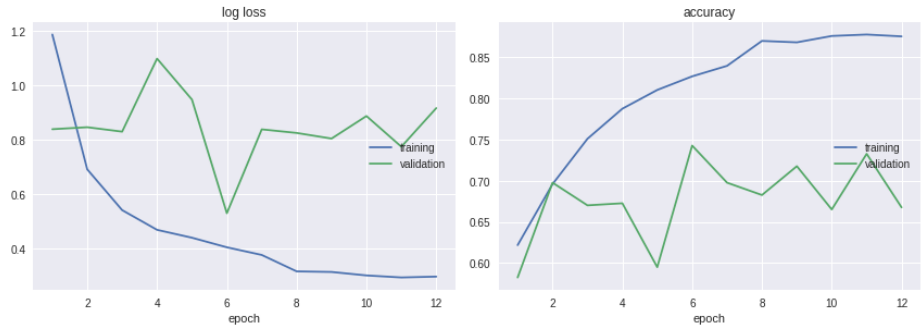
# Evaluation of the Models

*In this chapter we are going to detail the results that we have obtained in the different models comparing it with the metrics that we had defined in the evaluation section of the model.*

## 6.1 RESULTS

At the beginning, in the section on transfer of learning, the models that we were going to use were defined and their subsequent result was measured. The models are distinguished by the complexity of the architecture, the hyperparameters that have been used and the learning period.

The first model is the simplest of the three, using a resnet architecture with a total of 18 layers and 11 training periods. Thanks to these parameters the time needed to train the model has been approximately 4 hours. The accuracy of this model is 75%, you can see in the following image as the model has evolved over the years.




```
log loss:
training   (min:    0.294, max:    1.186, cur:    0.297)
validation (min:    0.530, max:    1.098, cur:    0.916)

accuracy:
training   (min:    0.622, max:    0.877, cur:    0.875)
validation (min:    0.583, max:    0.743, cur:    0.667)
Training complete in 237m 39s
Best val Acc: 0.742500
```

Figure 6.1: Training of ResNet-18.

The second model is an evolution of the previous one, we used the same architecture but with a higher number of layers (50) and a total of 21 epochs instead of 11. The good thing about this architecture is that you can increase the number of layers as we want and we will see a greater precision. However, there is also an over-training called overfitting in the training dataset. Also due to the complexity of the same and the number of epochs that have been established has needed a total of 7 hours to complete the training. The accuracy of this model is 88% being the highest of the three.


```
log loss:
training   (min:    0.117, max:    0.489, cur:    0.117)
validation (min:    0.297, max:    0.486, cur:    0.404)

accuracy:
training   (min:    0.797, max:    0.956, cur:    0.952)
validation (min:    0.780, max:    0.880, cur:    0.845)
Training complete in 445m 28s
Best val Acc: 0.880000
```

Figure 6.2: Training of ResNet-50.

The last model the architecture used is a ResNet alteration called DenseNet in which overtraining in small data sets is reduced and your training is more efficient than in the ResNet architecture. Although in our case has not been so, we have needed the same number of hours 7 hours and observe a greater overfitting using this architecture and parameters. The accuracy of this model has been 85% higher than ResNet-18 but lower than ResNet-50.


```
log loss:
training   (min:    0.131, max:    0.508, cur:    0.133)
validation (min:    0.333, max:    0.737, cur:    0.357)

accuracy:
training   (min:    0.803, max:    0.949, cur:    0.945)
validation (min:    0.672, max:    0.858, cur:    0.853)
Training complete in 432m 57s
Best val Acc: 0.857500
```

Figure 6.3: Training of Densenet-121.

| Architecture | Epochs | Optimizer | Time | Acc. Training | Acc. Test |
|---|---|---|---|---|---|
| ResNet - 18 | 18 | Adam | 4 Hours | 0.87 | 0.74 |
| ResNet - 50 | 21 | SGD | 7 Hours 30 Min | 0.95 | 0.88 |
| DenseNet | 21 | SGD | 7 Hours 45 Min | 0.94 | 0.85 |

In conclusion, the model we have selected has been the second (ResNet-50) due to the precision it gives us. In the following section we are going to calculate the sensitivity, the specificity and the area under the ROC curve.

### 6.1.1 Obtaining the Confusion Matrix

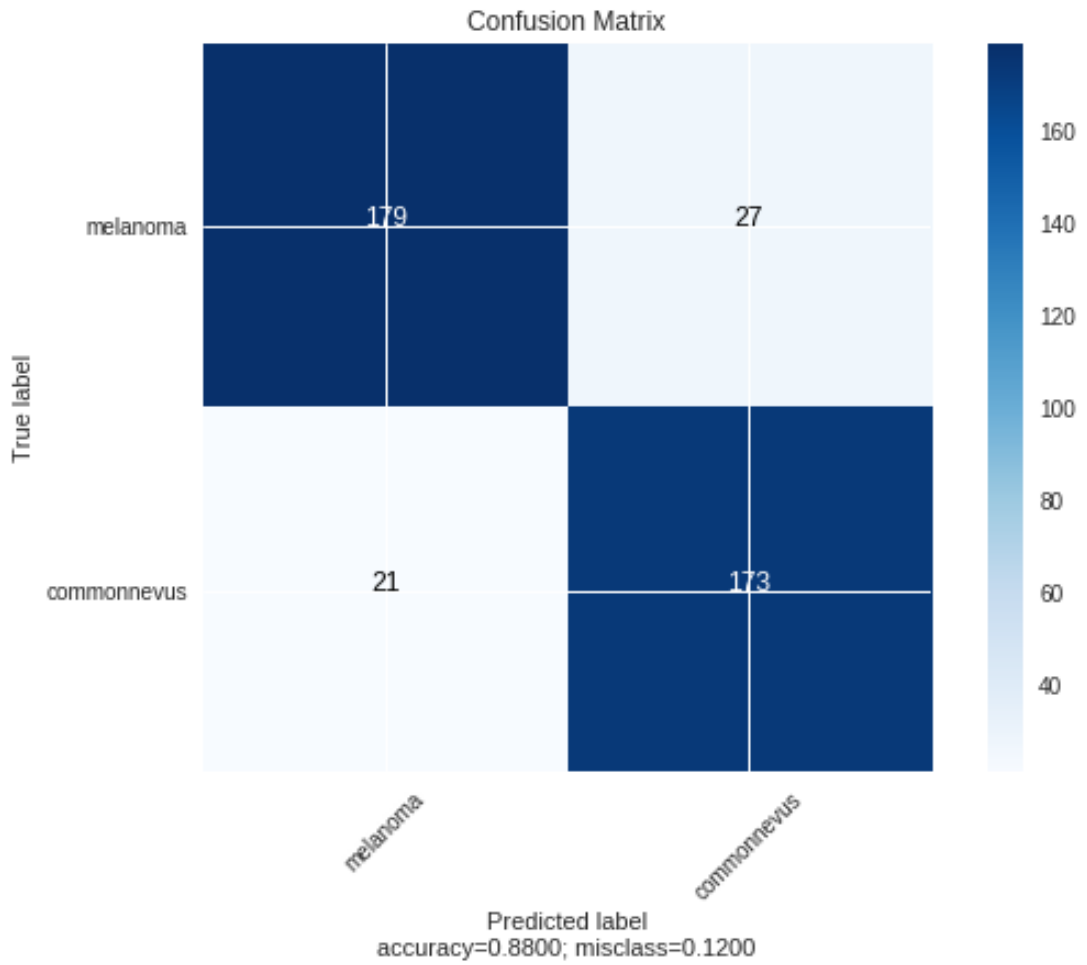



Figure 6.4: Confusion matrix of the model.

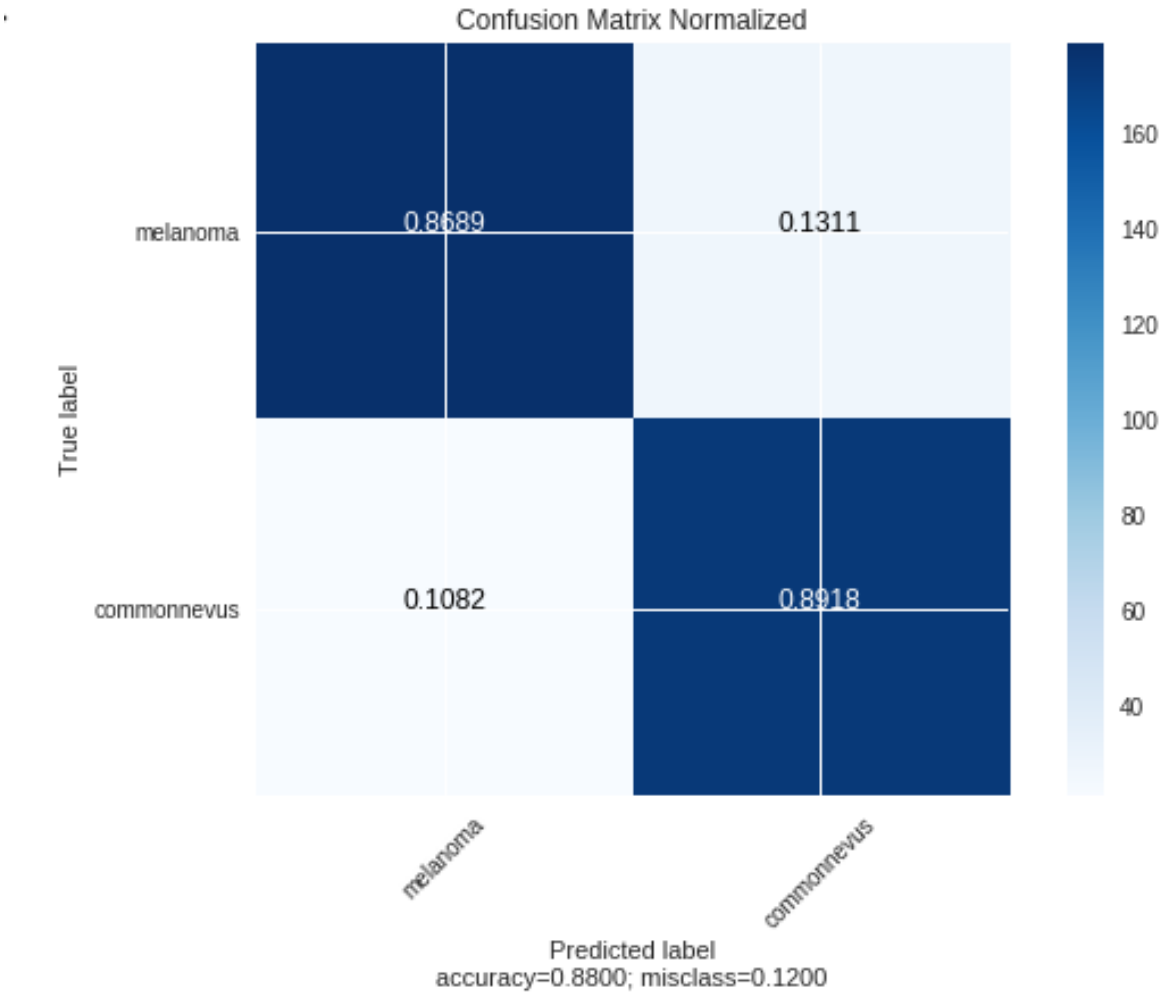



Figure 6.5: Confusion matrix normalized of the model.

The confusion matrix has been obtained on the test data set because if we use the training data set the result obtained and the metrics would be erroneous. From the confusion matrix we will compute the other metrics

1. TP = 179
2. FP = 27
3. TN = 173
4. FN = 21

Sensitivity will tell us which of the patients who really have skin cancer have been diagnosed with skin cancer

$$Sensitive = TP/(TP + FN) = 179/(179 + 21) = 0,895$$

Specificity measures the number of patients who do not have skin cancer have been categorized as having no skin cancer.

$$Specificity = TN/(TN + FP) = 173/(173 + 27) = 0,865$$

PVP will measure the proportions of positive and negative results in statistics and diagnostic tests that are true positive.The ideal value of the PPV, with a perfect test, is 1 (100%)

$$PVP = TP/(TP + FP) = 179/(179 + 27) = 0,868$$

NPV will measure the proportions of positive and negative results in statistics and diagnostic tests that are true negative. The ideal value of the NPV, with a perfect test, is 1 (100%)

$$NPV = TN/(TN + FN) = 173/(173 + 21) = 0,892$$

In the metrics it can be seen that the classifier is quite homogeneous, and things don't happen that are difficult to interpret. The classifier, as can be seen in the confusion matrix, has a fairly stable level of success in both classes, which indicates that it has correctly generalized the detection of cancer in the images. In short, the classifier has behaved very well compared to this set of images with respect to other classifiers and techniques.

Even knowing that medical images are complicated to deal with, the classifier has been able to generalize the details that separate an image of a patient who suffers from skin cancer of another patient who does not suffer from it, thus achieving predict accurately in most cases.

# Part III

# Conclusions

# Final conclusions

*In this final chapter we are going to present our conclusions of the project, recapitulating from the beginning what we have done and analyzing whether this has been a successful project or not. In addition, some ideas will be given to improve the project for the future.*

## 7.1 Personal opinion

To conclude this project, we are going to review everything developed in this work and whether it corresponds to the initial idea we had. The objective of this project was to apply the techniques and algorithms of deep learning for the prediction and classification of moles suspected to be possibly carcinogenic. In addition, a small mobile application was to be developed as a practical example of the use of this classifying model.

In the first place, a state of the art has been realized through the systematic analysis of the tools, architectures and techniques used by the companies and groups of research companies, which has led us to have a theoretical basis. We have also analysed a set of articles that provided us with some information on deep learning in general, and some of them also showed us the behaviour of other techniques in our field of study as well as a formation of the technologies to carry out this work in a successful way.

Next, we developed a set of tools to support our analysis, although due to its generic nature, we adapted it for use in any automatic learning project. We have obtained the set of images from ISIC, we have used pre-processing techniques and data augmentation to make our results as optimal and high as possible. We also divided the data into a training set and a test set.

We have carried out different models comparing them among them obtaining a model whose result surpasses what was expected at the beginning of the project. In this way, the effectiveness of the application of deep learning techniques for the analysis of medical images was verified.

The model was deployed on a server and finally we made a mobile application with some of the latest technologies to demonstrate that the use of these technologies are not only focused on the academic world but also in the real world, creating products that can be useful for people every day.

As a personal opinion it is worth mentioning that this project has been fascinating for me, in spite of the difficulties that it presented since I did not have the previous knowledge to use the technologies and to apply them to a field like the health which I am not specialized. I consider that I have faced the difficulties of the project, besides writing the documentation in English that is not my native language to obtain a project as professional as possible.

## 7.2 FUTURE WORKS

At the end of this project, different possible ideas emerge that will allow us to increase the results and accuracy of our model.

One of them contemplates, the architecture of a more complex software system to save the images, to treat them so that they had more quality. These images could come from the same application or from specialized centers. It would also be interesting to store meta information such as age, sex, birthplace, etc.. So we could create predictive models to know who and where are the people most likely to have skin cancer and from there discover the reason.

With respect to the mobile application section, it can be improved considerably as applications that currently exist in which the patient and the mole are monitored to see how size and colour change and to see if the risk of skin cancer increases or not.

# Bibliography


[1] Instituto Nacional Del Cáncer *https://www.cancer.gov/espanol/tipos/piel/pro* , June 2017 (pages v, 6, and 7).

[2] Pippa Biddle,Diamond Inc *https://bit.ly/2rkhRUY* , June 2017 (pages v and 10).

[3] Fernando Sancho Caparrini *http://www.cs.us.es/ fsancho/?e=72* 23 of April 2017. (page 15).

[4] Sagar Sharma *https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53* 9 of September 2017. (page 15).

[5] Rajalingappaa Shanmugamani *Deep learning for computer vision: expert techniques to train advanced neural networks using TensorFlow and Keras* 2018. (pages v, 15, 16, and 19).

[6] Abdullah-Al Nahid, Mohamad Ali Mehrabi, and Yinan Kong*Histopathological Breast Cancer Image Classification by Deep Neural Network Techniques Guided by Local Clustering* 7 March 2018.School of Engineering, Macquarie University, Sydney, Australia. (page 19).

[7] S. Kevin Zhou, Hayit Greenspan, Dinggang Shen *Deep learning for medical image analysis* 2017. (pages 19 and 20).

[8] Josh Patterson and Adam Gibson *Getting started with deep learning* 2018. (pages 19 and 20).

[9] Maxim Lapan *Deep Reinforcement Learning Hands-On/*, June 2018 (pages v and 14).

[10] R.K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, J. Schmidhuber *Advances in Neural Information Processing Systems.* 2013. (page 20).

[11] M.D. Zeiler, R. Fergus*Stochastic pooling for regularization of deep convolutional neural networks.* 2013. (page 20).

[12] Andrej Karpathy *https://twitter.com/karpathy/status/972295865187512320?lang=es* March 2018 (pages v and 21).

[13] *https://skymind.ai/wiki/comparison -frameworks-dl4j-tensorflow-pytorch* (pages 21 and 22).

[14] Vicky Kalogeiton Stéphane Lathuilière, Pauline Luc Thomas, Lucas Konstantin Shmelkov *https://project.inria.fr/deeplearning/files/2016/05/DLFrameworks.pdf* 25 of January 2017 (pages 21 and 22).

[15] Tensorflow API *https://www.tensorflow.org* 25 of January 2017 (page 21).

[16] Andrej Karpathy *https://twitter.com/karpathy/status/972295865187512320*

[17] The Area Under an ROC Curve definition *http://gim.unmc.edu/dxtests/roc3.htm* (pages vi and 26).

[18] Advanced Deep Learning Architectures Data *https://www.analyticsvidhya.com/blog/2017/08/10-advanced-deep-learning-architectures-data-scientists/* (pages vi, 26, and 27).

[19] Srijan Agarwal *https://opensourceforu.com/2017/01/introduction-h2o-relation-deep-learning/*, 21 of January 2018 (page 22).

[20] Oksana Kutkina, Stefan Feuerriegel *http://www.rblog.uni-freiburg.de/2017/02/07/deep-learning-in-r/*, 7 of March 2016 (page 22).

[21] Josh Patterson, Adam Gibson *Deep Learning: A practitioner's approach/*, August 2017 (pages v, 17, and 18).

[22] Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny, Marian Mazzone *CAN: Creative Adversarial Networks Generating "Art" by Learning About Styles and Deviating from Style Norms/*, August 2017 (pages v and 18).

[23] Oren Talmor, Gal Avineri *https://github.com/GalAvineri/ISIC-Archive-Downloader* (page 35).

[24] Fábio Perez, Cristina Vasconcelos, Sandra Avila, Eduardo Valle *Data Augmentation for Skin Lesion Analysis* (pages vi and 36).

[25] Rupak Kr. Thakur *https://medium.com/@rupak.thakur/aws-vs-paperspace-vs-floydhub-choosing-your-cloud-gpu-partner-350150606b39*, October 24 2017 (page 51).

[26] Dimpol, *https://stackoverflow.com/questions/44287965/trying-to-confirm-average-pooling-is-equal-to-dropping-high-frequency-fourier-co*, May 31 2017 (pages v and 20).

[27] Faizan Shaikh, *https://www.analyticsvidhya.com/blog/2017/08/10-advanced-deep-learning-architectures-data-scientists/* , August 9, 2017 (pages vi, 27, 28, and 29).

[28] *http://www.deeplearningessentials.science/transferLearning/* , August 9, 2017 (page 38).

[29] *https://vas3k.com/blog/machine_learning/Machine*, August 9, 2017 (pages v, 12, and 13).