COMPUTER VISION TOOLS FOR THE AUTOMATIC EVALUATION OF COLLAGEN VI DEFICIENCIES

A master's Thesis
Submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by
ALEX RAMÍREZ MÁRQUEZ

In partial fulfilment
of the requirements for the Master's degree in
ADVANCED TELECOMMUNICATIONS TECHNOLOGIES

Advisors: JOSEP MARIA PORTA PLEITE
VERONICA VILAPLANA BESLER

Barcelona, January 2022

Abstract

In recent years, there has been a growing trend in the use of Artificial Intelligence (AI) for processing medical images, more particularly in the classification of such images. Thus, using Deep Learning (DL) models that are able to classify medical images, one can build systems that can greatly help doctors to get a diagnosis.

The development of such systems for rare diseases is challenging, mainly because of the lack of a huge amount of data to study them. One of such diseases is collagen VI-related muscular dystrophy. The work presented below, combines machine learning models and data extracted from patients to implement an automatic diagnosis of collagen VI-related muscular dystrophies.

The Computer-Aided Diagnosis (CAD) system works on images of fibroblast cultures obtained from a confocal microscope and relies on a DL model based on a Convolutional Neural Network (CNN) to classify patches of such images in two classes: the negative class is denoted 'control' which corresponds to healthy subjects and the positive class is called 'patient' from persons affected by a collagen VI-related muscular illness. Once this classification is done, it is used to generate an overall diagnosis on a query image using a majority voting method.

It is difficult to find words to express my gratitude
to my family, girlfriend and friends for always supporting me.

Acknowledgements

I would express my deepest gratitude to everyone that has made this thesis possible.

In the first instance, in the academic environment, I would show my gratitude to my advisors, Josep Maria Porta and Veronica Vilaplana, for helping me in the development of the project. For giving me tips to solve the issues that have appeared over the course and to look for exits when I was stranded. Also, I would thank them to tracking my steps and caring about my evolution during the whole project.

In a more personal environment, I would like to thank my parents for the education they have given to me and the support I received from them in bad situations and always encouraging me to work hard to reach my own goals. Also for paying all my costs, specifically the studies that let me be an engineer.

To my brother Xavier, with whom we are always joking and laughing to let me forget studies problems. Also for supporting me and for being a reference for him.

To my girlfriend Mireia, for helping me every day in the studies and in life. Without her help, I would not be where I am now.

To all my friends, both from university life and old friends to giving me support on the decisions I take in my life.

Revision History and Approval Record

Revision	Date	Purpose
0	20/12/2021	Document creation
1	11/01/2022	Theoretical part revision 1
2	14/01/2022	Document revision 1
3	17/01/2022	Document delivery

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Alex Ramírez Márquez	alex.ramirez@estudiantat.upc.edu
Josep Maria Porta Pleite	porta@iri.upc.edu
Vernoica Vilaplana Besler	veronica.vilaplana@upc.edu

Written by:		Reviewed and approved by:	
Date	10/01/2022	Date	14/01/2022
Name	ALEX RAMÍREZ MÁRQUEZ	Names	JOSEP MARIA PORTA PLEITE VERONICA VILAPLANA BESLER
Position	Project Author	Position	Project Supervisors

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence. 1

AUC Area Under the Curve. 15

CAD Computer-Aided Diagnosis. 1, 7, 10–14, 19, 39, 40, 45, 49

CNN Convolutional Neural Network. 1, 5, 8, 10, 11, 13, 15–17, 20–31, 49

CV Computer Vision. 10

DL Deep Learning. 1, 11, 12, 17, 30, 49

GPI Image Processing Group. 12

IRI Institut de Robòtica i Informàtica de Barcelona. 10, 12

kNN k-Nearest Neighbours. 15

LIF Leica Image Format. 20, 21, 24, 50

NN Neural Network. 18

TIFF Tagged Image File Format. 20–22, 25, 32, 50

UPC Universitat Politècnica de Catalunya. 12

1 Introduction

1.1 Motivation

Deficiencies in the structure of collagen VI are a common cause of neuromuscular diseases with manifestations ranging from the Bethlem myopathy to the severe Ullrich congenital muscular dystrophy. Their symptoms include proximal and axial muscle weakness, distal hyperlaxity, joint contractures, and critical respiratory insufficiency, requiring assisted ventilation and resulting in a reduced life expectancy. Moreover, the skin and other connective tissues where collagen VI is abundant are also affected [1][2]. As described in the Online Mendelian Inheritance in Man (OMIM)[3] database entries 254090 and 158810, the collagen VI structural defects are related to mutations of three main genes, namely genes COL6A1, COL6A2, and COL6A3. As in all diseases caused by dominant mutations, where there is not a complete absence of a main protein, and where the effect of a genetic variant on the protein structure is not evident, collagen VI defects are difficult to diagnose. Thus, before any genetic analysis, the standard technique for the diagnosis of collagen VI-related dystrophies is the analysis of images of fibroblast cultures [4] (see Fig.1.1). Several aspects of the images, such as the orientation of the collagen fibers, the distribution of the collagen network, and the arrangement of cells in such network, are taken into account by the specialists to identify potential patients. However, this evaluation is only qualitative, and the regulatory agencies would not approve any treatment (such as, for instance, gene editing via the CRISPR technology) without an objective methodology to evaluate its effectiveness [5]. Thus, there is an imperative need for accurate methodologies to quantitatively monitor the effects of any possible new therapy. The work presented is intended to meet the need for precise techniques to quantitatively evaluate the impacts of any new treatment.

In this context, the availability of Computer Vision (CV) technologies offers the possibility to implement systems to support the diagnosis process. In a previous project, in 2017 researchers from the Institut de Robòtica i Informàtica de Barcelona (IRI) and from the Hospital Sant Joan de Déu developed the first Computer-Aided Diagnosis (CAD) system [6] for dystrophies caused by defects in the structure of collagen VI from images of fibroblast cultures obtained with a confocal microscopy. This CAD software implemented classical CV tools to identify high level features in the input images. Such features were then used to identify images similar to the input ones on a database with healthy subjects labeled as "controls" and subjects affected by collagen VI muscular dystrophy labeled as "patients". The images in this database were generated by doctors specialized in collagen VI defects. This database was also used to train a Convolutional Neural Network (CNN) to complement the results of the classical approach.

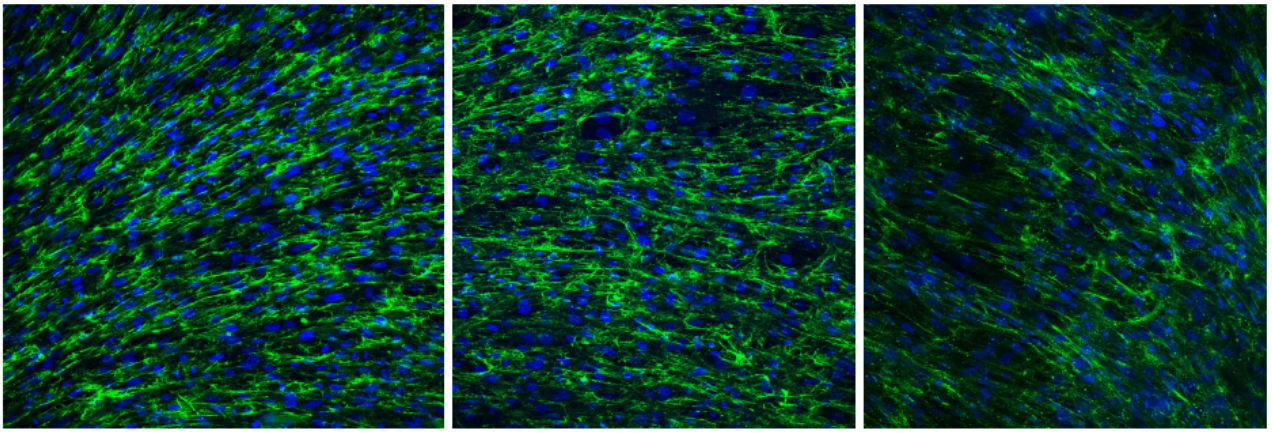

Figure 1.1: Confocal microscopy images of fibroblast cultures. Left: Image of a control subject. Center: Image from a patient with the Bethlem myopathy. Right: Image from a patient of the Ullrich muscular dystrophy. In the three images, the network of collagen is shown in green and the fibrolast nuclei in blue.

Despite the Hospital Sant Joan de Déu is the national center of reference for dystrophies related with collagen VI defects, the availability of the confocal microscope and its use to capture images of fibroblast cultures is relatively recent. Thus, the main limitation of the previous CAD software was the small size of the database with correctly classified patients and controls. However, from 2017 until now, the database has been significantly extended and this offers the possibility of notably improving the performance of such system.

1.2 Objective

The objective of this project is to develop a new CAD system to automatically classify images of fibroblast cultures obtained with a confocal microscope. It must be:

- **Accurate:** The system must generate a low number of missclassification, i.e., false positives, or false negatives.
- **Robust:** The accuracy must be maintained over the whole database available nowadays, not only on the reduced database used to generate the first CAD system.

Moreover, the developed solution must be integrated in the software currently used in the hospital to support the diagnosis of new cases of collagen VI deficiencies and to monitor the evolution of the patients. This monitoring is performed by obtaining scores for the different features extracted. In addition, a measure called 'Goodness' provided by the CNN is used to quantitatively evaluate the subject under study.

The objective of this project is challenging since the variety of the cases recently included in the database is significant. To address this challenge, we will rely on Deep Learning technologies. From all these technologies, a pretrained CNN such as Efficientnet will be used. Deep Learning is providing exciting solutions for the identification, classification, and quantification of patterns in medical images. The key feature of Deep Learning is the exploitation of hierarchical feature

representations learned solely from data, instead of handcrafted features mostly designed based on domain-specific knowledge.

The project has been carried out at the Universitat Politècnica de Catalunya (UPC) with the Image Processing Group (GPI) at the Department of Signal Theory and Communications (TSC), jointly with the Institut de Robòtica i Informàtica de Barcelona (IRI) and in collaboration with the Hospital Sant Joan de Déu.

This document is structured as follows. Section 2 describes the previous CAD system for collagen VI muscular dystrophies, which is the starting point of this project. Then, Section 3 describes the DL approach proposed in this project and Section 4 includes the results obtained with this approach. Section 5 details the project budget and, finally, Chapter 6 summarizes the conclusions and proposes future implementations to improve the system.

2 Previous Work

As mentioned in the introduction, this project departs from the first CAD system for dystrophies caused by defects in the structure of collagen VI. Therefore, before focusing on the work done in this project, we will describe the previous work. This existing system analyses images of fibroblast cultures obtained with a confocal image. Thus, we will first describe the process used to obtain such images. Then, we will describe the two CAD approaches already developed. While the first one is based on classical computer vision tools, the second one relies on the use of a CNN.

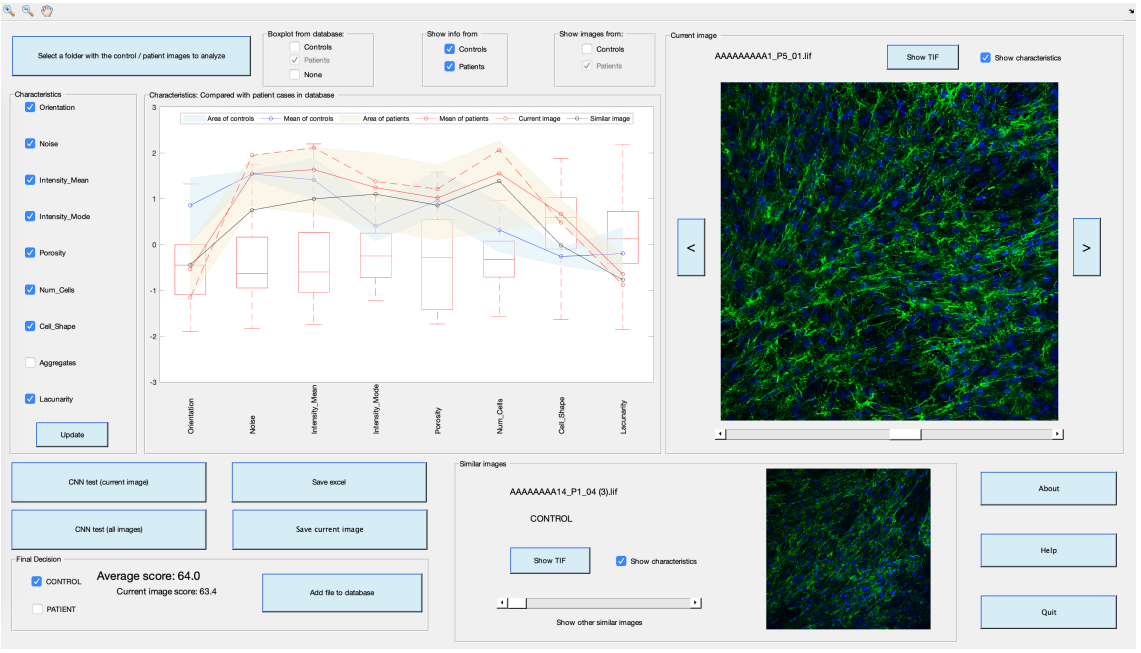

Figure 2.1: View of the interface of the proposed automatic diagnostic software.

The figure shows the CAD system interface. To run it, the images to be analyzed must be selected. The current image can be seen on the right side in large size. The left part is dedicated to show bloxpots of the features extracted by the classical approach. In the lower left part, there is a button that allows to run the inference with the Convolutional Neural Network (CNN). When executed, it will open a window where the prediction on each of the image patches will be displayed. In the lower right part, cases with similar features to the current image are shown.

2.1 Image Acquisition

To obtain the training images, samples from the forearm were obtained from patients, as well as from age-matched controls. Primary fibroblasts cultures were established using standard procedures [4]. Patient and control samples were treated in parallel, with 25 μg/mL of L-ascorbic acid phosphate magnesium (Wako Chemicals GmbH, Neuss, Germany) for 24 hours. After that time, cells were fixed with 4% paraformaldehyde in phosphate-buffered saline solution. Collagen VI was detected by indirect immunofluorescence using a monoclonal antibody (MAB1944, Merck,

Germany) and fibroblast nuclei were stained using 4,6-diamidino-2-phenylindole (Sigma Chemical, St. Louis, USA).

Images of the samples with 1024x1024 pixels were obtained with a Leica TCS SP8 X White Light Laser confocal microscope with hybrid spectral detectors (Leica Microsystems, Wetzlar, Germany), an HCX PL APO 20x/0.75 dry objective, and the confocal pinhole set to 1 Airy unit. Collagen VI was excited with an argon laser (488 nm) and detected in the 500-560 nm range. Nuclei were excited with a blue diode laser (405 nm) and detected in the 420–460 nm range. Appropriate negative controls were used to adjust confocal settings to avoid non-specific fluorescence artifacts. The detector gain and offset values were adjusted to use the entire dynamic rate of the detector (12 bits), but avoiding oversaturated voxels. Sequential acquisition settings were used to avoid inter-channel cross-talk. Ten sections of each sample were acquired every 1.5 μm along the focal axis (Z-stack) and combined into an intensity projection to form a single image.

Finally, the training images were labelled either as control for healthy subjects or patient for individuals affected by collagen VI muscular dystrophy by a team of expert doctors. For this reason, it can be safely assumed that there is no noise or incorrectness in the labels of the data used for training the proposed system.

2.2 Classical Approach

The fist CAD system developed to identify images with defects in the collagen VI relied on classical computer vision tools. A total of nine high-level features have been defined relying on the experience of the doctors. These features are:

- The orientation of the collagen fibers.
- The noise in the images.
- The mean intensity.
- The intensity mode.
- The porosity, defined as the size of the wholes in the collagen network.
- The number of cell nuclei in the image.
- The shape of the cell nuclei.
- The number and size of the collagen aggregates (i.e., spots in the image with high concentrations of collagen).
- The lacunarity, which is a fractal measure of how patterns fill space.

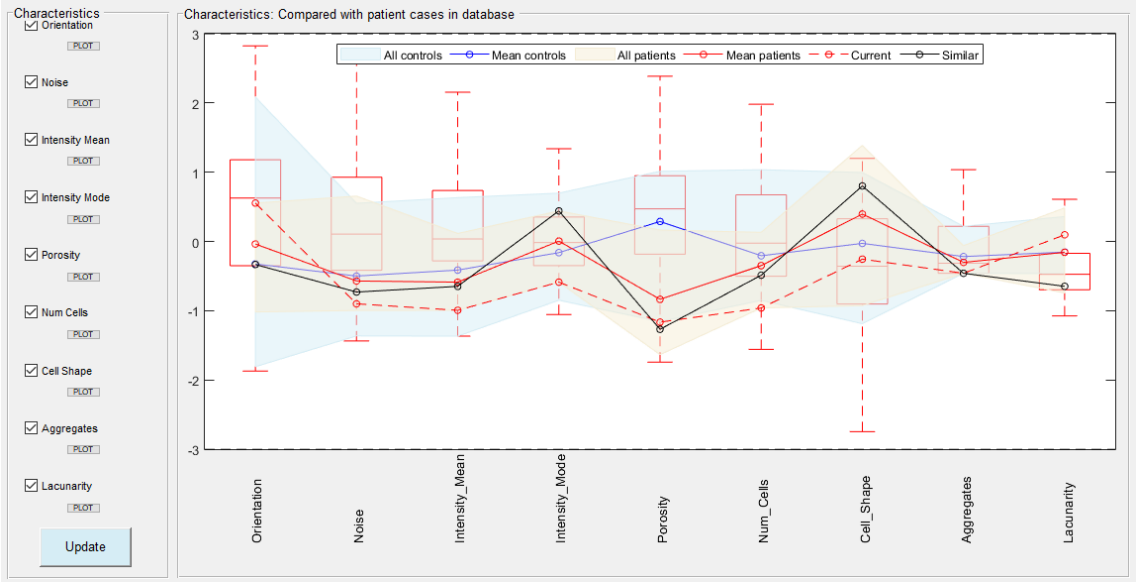

Figure 2.2: Overview of the proposed automatic diagnosis system using the classical approach.

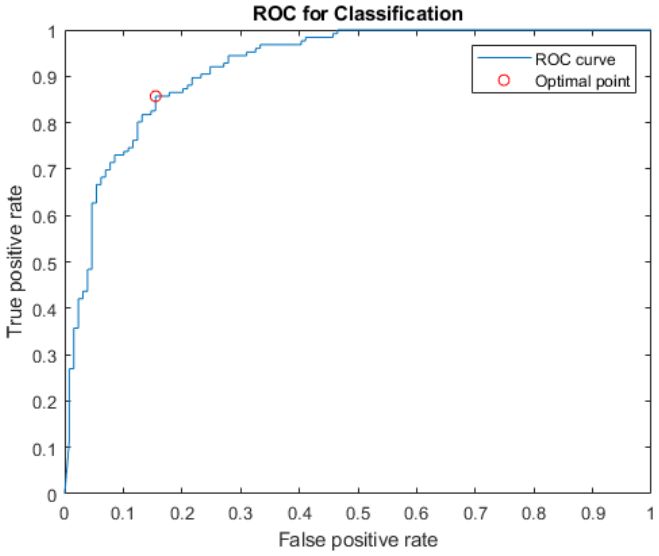

Figure 2.3: ROC of classical approach.

The characteristics are used in a k-Nearest Neighbours (kNN) search to identify similar cases in the available database. The proposed diagnostic (see figure 2.2) is based in the number of cases/patients retrieved in such query. The classification accuracy of this approach was about 88% with and Area Under the Curve (AUC) of 90% (see figure 2.3).

2.3 Deep Learning Approach

The previous approach was complemented with a classification approach relying on a CNN. Since this approach is the starting point of this project, we will describe it in detail, including the data augmentation procedure, the architecture used to classify patches of the input images, the majority voting system used to classify images, and, finally, the results obtained with this approach.

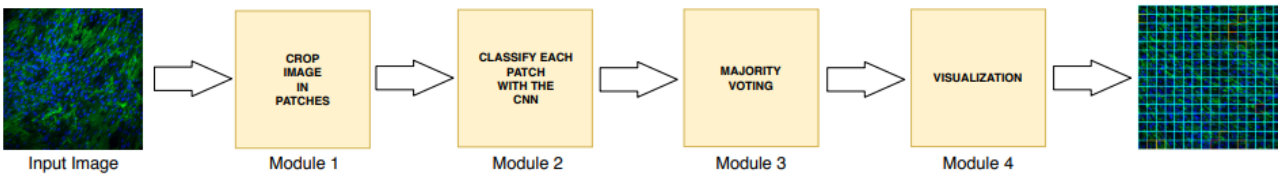

Figure 2.4: Overview of the proposed automatic diagnosis system using a majority voting on the individual patches decisions of the CNN model. the system also provides a detailed visualization of the diagnosis.

2.3.1 Data Preprocessing

Since the image acquisition procedure is complex and time-consuming, and we are dealing with a rare disease, the available samples are limited. For this reason, to generate enough data to train the CNN, the data augmentation scheme described next was used.

The data was augmented by splitting the available full-size images into small, non-overlapping patches. Each patch was used as an independent input to train the CNN model. The patch size was of 64x64 pixels, which was enough to capture the main features of the input images, i.e., a significant portion of the collagen network and several nuclei or parts of them. Moreover, the use of such small patches allowed to generate enough data for training and testing.

Since input images were of 1024x1024 pixels, 256 patches were generated from each image. Each patch was further transformed to get even more variations of the data. The patches were rotated clockwise by 90, 180, 270 and 360 degrees and every rotated patch was flipped horizontally. Thus, eight different variations of each original patch were generated and, consequently, each input image produced 2048 training patches for the CNN. Since initially we had 276 images, with the proposed data augmentation process, the training and testing sets included, respectively, 56320 and 14336 patches, without taking into account the rotated and mirrored patches. The patches were normalized so that they had zero mean and unit variance. Normalization reduces the effect of possible noise in the images and improves the learning process avoiding its premature convergence.

2.3.2 CNN

A CNN architecture suitable for the addressed problem was identified. To this end, two concepts were considered: the size of the network (with a small/large number of convolution kernels) and the abstraction level (with increasing/decreasing sizes of the subsequent layers of the CNN). Experimentally, small CNNs with a decreasing number of features provided the best results. The proposed architecture is detailed in Table 2.1 and is similar to the one in [7], which has already been proven to be particularly adequate for image classification. However, the proposed network is smaller, since the classification task addressed here is simpler than the one addressed in [7].

Layer	Type	Number of neurons (output layer)	Kernel size	Stride
1	Convolution	64x64x128	3x3	1
2	Max Pooling	32x32x128	2x2	2
3	Convolution	32x32x64	3x3	1
4	Max Pooling	16x16x64	2x2	2
5	Convolution	16x16x32	3x3	1
6	Max Pooling	8x8x32	2x2	2
7	Fully Connected	150	-	-
8	Fully Connected	2	-	-

Table 2.1: Details of the CNN architecture proposed to classify the 64x64 pixels image patches.

Table 2.1 provides a schematic representation of the CNN architecture finally used to classify the patches of the fibroblast images. In this architecture, each image patch is passed through three convolutional layers (layers 1, 3, and 5), which include the application of ReLU activation functions after the kernel computation. The first convolution layer defines 128 feature maps of size 64x64, the second one defines 64 feature maps of size 32x32, and the last one defines 32 feature maps of size 16x16. The reduction in the size of the feature maps is obtained with max-pooling layers (layers 2, 4, and 6), with a stride of 2 following the convolution layers. After the feature generation layers, the classification is implemented with two fully-connected layers (layers 7 and 8). The first one has 150 neurons followed by a ReLU activation function and trained with a dropout mechanism with probability 0.5. The second fully connected layer has 2 neurons, whose output is truncated into a single binary output by a sigmoid activation function to provide the final classification.

2.3.3 Majority voting

The DL approach provides a score for each patch in the test image. The goodness of the overall image is obtained with majority voting, i.e., it is the percentage of patches classified as CONTROL class. This final score, thus indicates quantitatively how much the image resembles that of a healthy person, being 100% a healthy person and 0% a completely sick person.

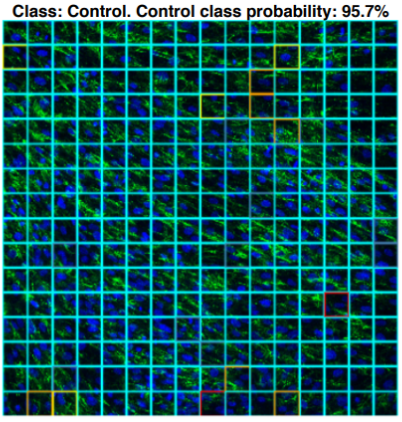

Figure 2.5: Visualization of the diagnosis of a given fibroblast culture image. Each patch of the image is colored according to its probability of belonging to the control class. The system also gives an overall score computed as the percentage of patches classified as control in the image.

2.3.4 Results

The following is a summary of the results obtained in this first version, which will serve as a reference for the current work being carried out.

First of all, the configuration of the hyperparameters with which the best results were obtained will be explained. The batch size was set to 32. Overall, every epoch required around 2 minutes to finish and under 10 epochs are necessary to converge. The function loss used is the categorical cross entropy which is used in multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories, and the model must decide which one. For training the Neural Network (NN) Adam optimizer was used, with a learning rate of $\alpha = 0.01$, an exponential decay rate for the first moment estimates of $\beta_1 = 0.9$, an exponential decay rate for the second moment estimates of $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ to prevent any division by zero. In order to evaluate the different models, a 5-fold cross-validation was carried out.

In this confusion matrix (Figure 2.6), the quantity of patches accurately and mistakenly classified corresponding to the control class is in the first row, i.e. 7585 patches correctly classified as control, and 530 false positives (fp), i.e. the inputs incorrectly classified as patient. The classification of patients is given in the second row, where 102 are false negatives (fn), i.e., inputs incorrectly classified as control, and 6119 are true positives (tp), i.e. inputs correctly classified as patient. In the majority voting system, the accuracy is perfect and, thus, the confusion matrix is trivial and not given here.

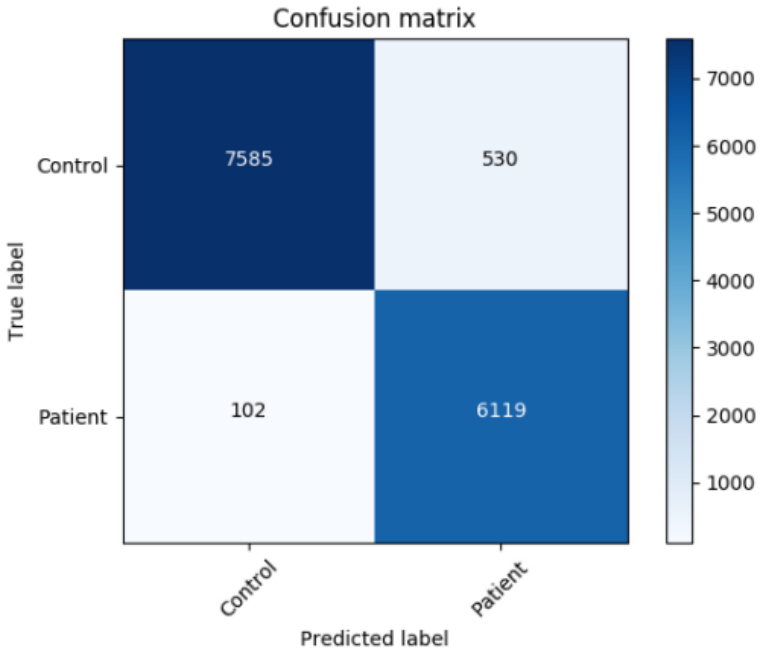

Figure 2.6: Confusion matrix of the test set for the model trained with the 64x64 image patches

The approach proposed in the first CAD generates features that capture the latent patterns in the images and, consequently, it outperforms the other alternative approaches. It achieves an accuracy of 95%, a precision of 92%, a sensitivity of 98%, a specificity of 93%, and an F1 score of 95%. This outstanding performance at the level of patches is the basis of the perfect classification obtained with the majority voting scheme.

3 A CNN to Detect Collagen Deficiencies

This project arises from the necessity of considering the extended dataset of fibroblast cultures that has been compiled at the Juan de Déu Hospital from 2017 till now. Thus, we first describe this dataset and then present the limitations of the previously developed CNN to correctly classify this dataset. We also describe the attempts we made to extend this CNN. Finally, we describe the CNN architecture proposed in this project, which is an adaptation of the smaller member of the EfficientNet family of CNN models.

3.1 The New Dataset

The available dataset includes fibroblast culture images. In particular:

- 28 cases labeled as control
- 28 cases labeled as patient
- 11 cases labeled adult control
- 12 cases labeled adult patient

Adult cases tend to be milder since patients with severe cases rarely survive the adolescence.

Usually, samples from a potential patient and for a control are obtained on the same day. The samples are analyzed together. In this way, each patient has a control associated with him/her, which is contrasted to check that the sample processing has been correct and also serves the specialists to compare them with each other. Thus, for each case, there is the date at which the samples were obtained and the LIF files obtained with the confocal microscope. The LIF is a multi-image (i.e., each file can contain more than one image), multi-plane (i.e., each image can include several layer) file format. In our case, for each case there are, on average, about 10 LIF files each containing a single image and where each image includes 10 planes obtained from the same sample, but at different distances from the objective (i.e., in the Z direction).

The first step was to convert each of these LIF files to a format where the image can be displayed. On one hand, this will help to understand how the images are and to draw a good preprocessing strategy for the network to be able to generalize correctly. On the other hand, this will facilitate the final integration of the software, i.e., to show the subdivision of the image into patches and to know their prediction through a color code.

The conversion format of choice is Tagged Image File Format (TIFF), although it is also known as TIF. This format works very well for images such as those processed in this work, since it is a lossless raster format, which allows us to store very high quality images. On the other hand, the generated files have a larger size than if they were saved in another format such as JPEG or PNG.

For the development of this project there was no inconvenience in having so many large files stored, since there was enough memory available. Besides, what was really needed was to have high quality images to feed the network.

The conversion between the two formats was carried out using MATLAB software. As mentioned in the image acquisition section, 10 sections every 1.5 µm along the focal axis (Z-stack) were performed for each sample. These sections are stored in the LIF file and projected against the XY plane to generate a single image that combines the information from all 10 planes. The dynamic range of each of these planes is 12 bits, i.e. from 0 to 4095, but when saved in TIFF format, this range is 8 bits, i.e. from 0 to 255. A noteworthy feature, as mentioned in section 2, is that despite being RGB images, there are only two channels different than 0. The red channel is not used, but has not been deleted either, since the results are just as good with it as without it. The green channel shows the collagen while the blue channel shows the cells of the sample.

During the creation of the TIFF images, there were 3 cases that could not be completely processed. These cases corresponded to the oldest extractions, dated 2017, where the dynamic range of the planes was 8 bits instead of 12 as was the case for the rest of the images. For the first version, these cases were discarded because it was thought that they could alter the trained model. After an analysis of MATLAB documentation it was found that the dynamic range of the image did not matter as it was always stored in 8 bits. It was decided to incorporate the images of these 3 cases, but it turned out that there was also a difference in the size of the planes, while all the others were 1024x1024, these three were 800x800.

In addition, it was decided to find out why these cases contained a different number of pixels if the size of the samples extracted from people were similar. The pixel resolution of both 800x800 and 1024x1024 images was then checked and, it was found that while resolution in the former was 0.97µm , resolution in the latter was 0.7579µm:

$$\frac{0.97\mu m}{0.7576\mu m} \cdot 800\ pixels \approx 1024\ pixels \tag{1}$$

It can then be concluded that the sample size is the same, but a different number of pixels are used to cover it. The three cases could be rescaled, but it was decided not to do so since it only affected 3 cases and furthermore, it was assumed that making such modifications could negatively affect the learning of the network.

The following will explain how the adult cases contained in the dataset will be treated. Unfortunately, people suffering from this disease do not have a very long life expectancy. When there is a reference to adult cases, the patients are most often teenagers or adults at a young age. Usually these people present low symptomatology.

Initially, adult cases were not treated and only pediatric cases were processed. How the adult cases were incorporated and what results were obtained will be explained later.

Discarding the cases mentioned above, those with different image resolution and adults, the dataset contained 227 TIFF images labeled as control and 257 images labeled as patients.

A process common to all the experiments detailed below was data augmentation. More specifically, how patches were generated from each of these images in order to increase the number of data so that the network would be able to train without overfitting. In addition to the creation of patches, other on-the-fly processing was also applied to generate new patches.

To detect potential problems in the dataset, it was proposed to use the model that had been used so far and which had a 95% accuracy. In this way, it would be known at the patch level which ones were not being classified correctly, at the image level those that presented a greater error and finally those cases that were not classified according to the annotated class.

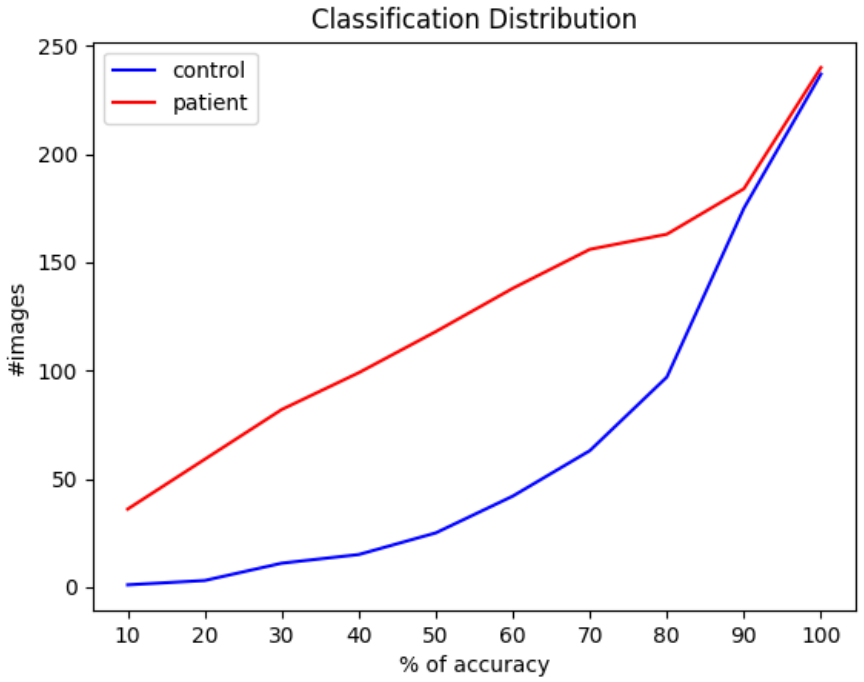

Figure 3.1: Accuracy of new dataset at image level

The image 3.1 shows for each of the classes, in blue the control class and in red the patient class, and at the image level, the accuracy obtained. The accuracy per image is no more than the percentage of correctly classified patches. The boundary between the 2 classes is 50%, i.e. if an image has 50% or more correctly classified patches it implies that the prediction and the actual label are equal. It can be observed that while the control class does not have a large number of misclassified images, about 25 images, the patient class does have a large number of misclassified images, about 120 images.

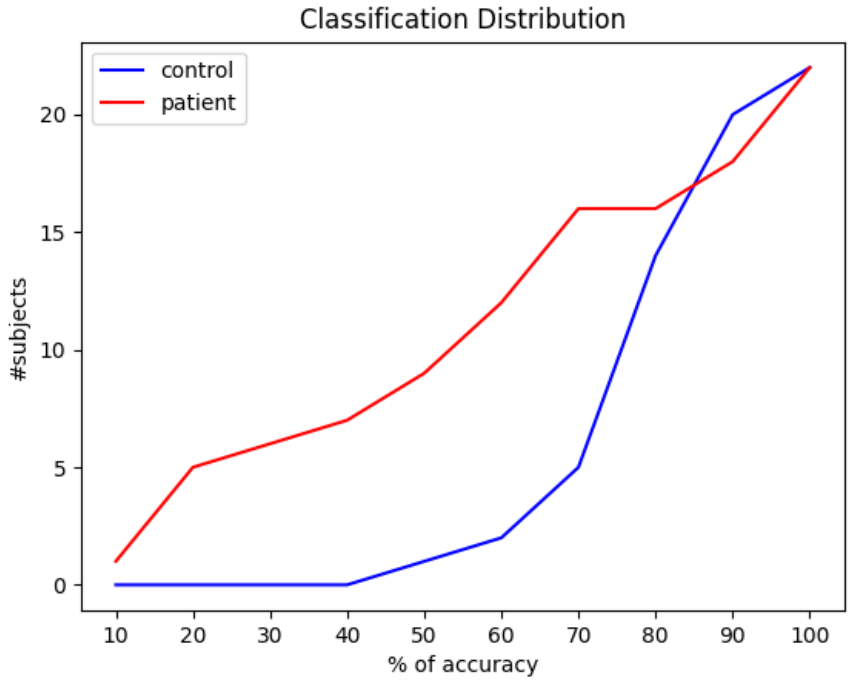

Figure 3.2: Accuracy of new dataset at subject level

The figure 3.2 shows for each of the classes, in blue the control class and in red the patient class, and, at the subject level, the accuracy obtained. Each person has a set of images extracted from his samples, generally about 10. The accuracy per image is nothing more than the percentage of correctly classified images. The boundary between the 2 classes is 50%, i.e. if a person has 50% or more of correctly classified images it implies that the prediction and the actual label are equal. It can be observed that while the control class has only 1 misclassified subject, the patient class has up to 8 misclassified subjects.

Looking at the graph 3.2, it was decided to discard those cases at the subject level that were worst classified although it was not convenient to discard all misclassified cases as this way the network would not be able to learn about the new cases presented in this dataset.

Therefore, on the previous graph, a threshold was established where those who had an accuracy above it were considered as correct cases to be trained and those who were below it were discarded.

The hospital doctors were contacted to consult the 5 worst cases (threshold $< 20\%$). Their response was to discard these cases as they were rare patients.

The worst classified case corresponds to a patient that presents a secondary phenotype. The following worst classified case corresponds to a young adult and as mentioned above, adult cases are not included in the dataset. The third one, is the same patient of the first case (secondary phenotype). The fourth and fifth cases are the same pediatric patient which has a rare or new mutation.

All the cases mentioned above are rare cases which are best not considered in the training (at least the 3 most misclassified). The other two are from the same person who is an atypical malignant (possibly a non-standard mutation). Although it could be interesting to consider them because the model would see more examples, finally it was decided not to include them.

Not all cases correspond to a different person, although at the beginning it was considered that each sample in the dataset was from a unique subject. In the case of the cases labeled as controls, many of them came from the same repeated person. In the case of the cases labeled as patients, this also occurred but to a lesser extent. This is because the tests were repeated to see how the disease progressed and to be able to analyze its evolution.

Despite the fact that there are repeated individuals, the hospital specialists assured that the processing of the samples is repeated each time they are analyzed with the confocal microscope. Thus, even if the samples are the same or come from the same person, the LIF files are generated anew each time.

After observing that some people were repeating themselves, it was decided to give a unique identifier to each person to take into account how the dataset was actually distributed. In this way, better conclusions could be drawn from the results. Especially when it came to detecting misclassified cases.

After unique identification, the dataset contains 5 different pediatric people labeled as controls and 20 pediatric people labeled as patients. Regarding the adult cases, all the cases labeled as control correspond to the same person while in the patients, there are 9 different people. Furthermore, the persons in the borderline cases also have other cases whose person-level accuracy is above the stated threshold separating borderline and non-bankable cases. For this reason 2 cases also had to be discarded, making a total of 7 cases discarded, 4 of them were from the same person where 3 were already considered doubtful cases, 2 where 1 was considered a rare case and 1 was already considered doubtful.

All these cases are considered as doubtful, but they should be considered part of the training at some point when a model with good accuracy was obtained. It would be good to add them to see if the classification becomes poor or if the network is able to capture small details and get a good accuracy on them. This project cannot be limited to re-learn (a bit better and with a bit more generality) what the previous network already knew, the idea here is to try to learn more things (to have a good accuracy even in doubtful cases). Once a clear network structure that works well was obtained, the next step is to work with the doubtful ones (or at least with some of them).

While this identification was being carried out, some cases that were mislabeled were also modified. Specifically, a control case and a patient corresponding to the same date were not

pediatric but were adult cases. In addition, on another date, a case labeled as a control was not a control but was a patient.

Once doubtful cases have been ruled out, the cleaned dataset was distributed as follows:

- 28 cases labeled as CONTROL
- 21 cases labeled as PATIENT
- 11 cases labeled ADULT CONTROL
- 12 cases labeled ADULT PATIENT

This distribution is represented by 221 images corresponding to control individuals and 199 images labeled as patients. The total amount of images from patients that have been discarded is 57. As it can be seen, the dataset is more or less balanced so there is no need to apply any technique to balance it

3.2 Limitations of the Previous Network

The next step was to re-train the CNN, but now using the new dataset provided by clinicians at the hospital, and whose distribution was explained in the first part of this section 3.1.

For the development of this experiment, the entire dataset was used. It consisted of 477 TIFF images and contained both classes in a more or less equal proportion. It was divided in a proportion of 80% to train the architecture and 20% for testing purposes. After running this experiment, it was observed that the metrics obtained with the test set were not acceptable (see section 4.1).

This is where this training task, which in principle should not present a great difficulty, was complicated. The first step proposed to detect the source of errors was to use only those images that were present in both the old dataset and this new dataset. Since the names of the files were not the same in both sets, it was not possible to know exactly which images belonged to both sets, although it was known approximately between which years they were included. So from the new dataset only images up to 2018 were taken.

By training this subset, the network was able to generalize correctly since the metrics obtained resembled those obtained in the previous section 2.3.4 where the old set was used for retraining.

It was then considered the possibility that the data was not completely well labeled and, in addition, there were some cases that were abnormal. All the comprehensive analysis that was performed has already been explained in the previous section 3.1 in more detail.

In the experiments performed until then, a single training stage was performed and the metrics were obtained on a specific set of test images. In order to avoid relying on a fixed training-test partition, k-Fold cross-validation [8] was used to compute the model performance in a more robust way. Cross-validation is a resampling method used to evaluate AI models on a restricted information set. The procedure has a single parameter called k that alludes to the quantity of groups or folds that a given information set is to be split into. This technique is basically used to estimate the skill of a machine learning model on unseen data. That is, to use a limited example to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model. It is a popular strategy since it is easy to comprehend and in light of the fact that it produces by and large outcomes in a less biased or less hopeful estimate of the model skill than other techniques, like a straightforward train/test split.

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups.
3. For each unique group:
   (a) Take the group as a hold out or test dataset.
   (b) Take the remaining groups as a training dataset.
   (c) Fit a model on the training set and evaluate it on the test set.
   (d) Retain the evaluation score and discard the model.
4. Summarize the skill of the model using the sample of model evaluation scores.

It is vital to see that every observation in the data test is assigned to a singular group and stays in that group for the duration of the procedure. This implies that each example is offered the chance to be used in the hold out set 1 time and used to train the model k-multiple times.

Figure 3.3: kfold cross-validation method

3.3 Using Larger Patches

At this point we had a model that, in spite of having decreased the accuracy with respect to the model we had until then, was much more robust since we had managed to see many more examples. The objective of obtaining a much more robust model than the previous one had been achieved, but the purpose now was to try to improve the metrics based on a good optimization of the parameters, both the hyperparameters of the network and the processing of the input data, as well as to offer a good visualization of the training to know where to focus the efforts.

So far, size 64 patches have been used mainly for two reasons. The first is that the full image size of 1024 is divisible by the patch size 64, and this makes it easier to create patches without overlapping and occupying the entire image. Secondly, it allows to create enough data for the network to be able to train correctly without generating too much overfitting. But now that the number of images has grown to more than 100 for each of the two labels, one could think of using larger patch sizes. For example, they could be increased to 128x128 so that the network would not learn such local details and would be able to extract somewhat more global features. Furthermore, it was demonstrated that the classical approach, which works with the entire image size, worked better than the CNN in the new dataset. Note that the ideal patch size for training the network if there were enough images would be the size of the image itself.

For this experiment we have used the same number of images as in the previous section, but since the patch size has doubled, the number of pixels per patch is quadrupled, so there are now fewer patches to train. Still, it is possible to train the network to generalize correctly without generating overfitting. In the same way as in the previous experiments, in this one the Kfold technique has also been used to see if the results obtained were more or less constant regardless of the test set used.

So far there are two models obtained with the old network, the one where the input size was 64x64 and the one with the input size of 128x128 and also the model using the Efficientnet (see section 3.4) network. So with these 3 models it is not possible to compare which of the two networks performs better since, as seen with the first two models, only increasing the input size already improves the metrics. Therefore, for comparison, the experiment of training the old network with 224x224 patches is performed. In this way, both networks can be compared equally as long as the hyperparameters used are the same.

3.4 A New Architecture

The old network offers reasonable results, but not as good as the ones obtained with the small dataset available when it was developed. To improve these results, one should extend the network (adding more layers), optimize the hyperparameters, or use another network. This is where Efficientnet[9] appears. EfficientNet is family of networks which work very well for classification tasks.

3.4.1 The EfficientNet Family

Convolutional Neural Network are commonly developed at a fixed resource budget, and then scaled up to obtain better accuracy if there are more resources available. This can be done by adding more layers to the baseline model. In [9], the authors systematically study the model scaling and identify that carefully balancing network depth, width, and resolution can lead to better performance. This process of expansion is not easy to understand and there are many ways to do it. The most common is to raise their depth or width. Another less common, is to scale up models by image resolution. Based on this observation, what they propose is a new scaling method that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective compound coefficient. They demonstrate the effectiveness of this method on scaling up MobileNets and ResNet.

In most cases, CNN architectures are either too large, too deep, or have an extremely high resolution. Increasing the dimensions initially improves the model, but it soon saturates, and the model created simply has more parameters and is thus inefficient. Nevertheless, scaling up depth, width and resolution in a more systematic manner can improve the performance without saturating. Efficientnet applies that scaling method to a baseline network consisting of MBConv blocks, the basic building block that also makes up MobileNetV2. These blocks are a type of residual block used for image models that use an inverted structure for efficiency reasons.

Architectures such as EfficientNet are particularly useful for using deep learning on the edge, as it reduces computational cost, battery usage, and also training and inference speeds. This kind of model efficiency ultimately enables the use of deep learning on mobile and other edge devices. The family of models is efficient and produces better outcomes with a significantly lower number of parameters. The models of this architecture go from 0 to 7 and the main differences is the number of sub-blocks that contain each model and the input resolution.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figure 3.4: EfficientNet-B0 baseline network – Each row describes a stage i with L^i layers, with input resolution hH^i, W^i and output channels C^i

To go significantly further, they utilize neural engineering search to design a new baseline network and scale it up to get a family of models, called EfficientNets, which accomplish much

better accuracy and efficiency over past CNN. Specifically, their greater model, EfficientNet-B7, accomplishes state-of-the-art 84.3% top-1 accuracy on ImageNet, while being 8.4x smaller and 6.1x quicker on inference than the best existing CNN. Their EfficientNets likewise transfer well and achieve state-of-the-art accuracy on CIFAR-100 (91.7%), Flowers (98.8%), and 3 other transfer learning datasets, with a significant degree less parameters.

Because training EfficientNet on ImageNet dataset takes a huge amount of resources and several techniques that are not a part of the model architecture itself. Hence the Keras implementation, which is the one that has been used for this project, by default loads pre-trained weights obtained via training with AutoAugment[10].

3.4.2 EfficientNet on Collagen VI Images

The idea was to use Efficientnet to see what possibilities it offered, not only in accuracy but also in training/inference times and resources consumption. In the event that this network did not offer good results, other pre-trained network options would be explored to see if they perform better than the current network. For example, the possibility of using VGG16[11] or its larger version VGG19, InceptionV3[12] or ResNet50[13], among others, has been considered.

All models of the Efficientnets family are accessible for both Tensorflow and Pytorch. For this situation it has been decided to use Tensorflow since the old version is implemented in this library and it was consistent to continue using it.

For the development of this work, the smallest model (EfficientNetB0) has been chosen since it is also the one with the smallest size and therefore the training times are shorter. This model takes input pictures of shape (224, 224, 3), and the input data should range between 0 and 255. The first layers are liable for normalizing the input data. Consequently, the patches to be made should have this size. If not, the first layers of this architecture consist of a preprocessing of the data that, among different changes, plays out a rescaling so the input size is 224x224x3.

As mentioned, this architecture has already been pretrained on a dataset called ImageNet. For that reason these networks can be retrained in 3 distinct ways. Assuming you have a huge set, you can train from scratch, that is, with an arbitrary initialization of weights. The remaining two options are more interesting for the case under study, since they start from weights that have already been trained on another set of images as discussed above. On the one hand there is a technique called fine tuning[14] where all the layers of the network are trained and the weights of all of them are updated as the training progresses. On the other, there is transfer learning[15] where the Effcientnet layers are locked so that the assigned weights are not updated in the training. Only the output layers are adjusted. In this way, faster training is achieved.

In this work the last two options have been tested, since training from scratch does not take advantage of the learning obtained from the pretraining.

Since the case at hand is a binary case, the output of our network must be 2. But as previously mentioned when the Efficientnet architecture was defined, it has been trained with 1000 classes, which implies that the last layer contains 1000 outputs. That is why an adaptation is needed to be able to have a network that works for this work. As figure 3.4 shows the last layers must be removed and replaced by others that are suitable for this case. When the model is intended for fine tuning or transfer learning, the Keras implementation provides an option to remove the top layers. As this network is implemented using Tensorflow library, the lower layers cannot be eliminated one by one (corresponding to stage 9 from figure 3.4). Instead, all 3 layers must be completely delated and then added individually by convenience. Replacing the top layer with custom layers allows using EfficientNet as a feature extractor in a transfer learning workflow. In this project they have been replaced by a Global Average Pooling (GAP) layer followed by a dense layer that contains 2 outputs, one for each class (see table 3.1).

Stage	Type	Output Shape	Param
1	Input Layer	[(None,224, 224, 3)]	0
2	efficientnetb0	(None, None, None, 1280)	4049571
3	Global Average Pooling	(None, 1280)	0
4	Dense	(None, 2)	2562

Table 3.1: Details of the Efficientnet CNN architecture proposed to classify image patches.

As in the previous experiments, the Kfold technique was also used in this one to see if the results obtained were more or less constant regardless of the test set used. Once similar results are obtained for the different test sets, a single training can be performed to try to obtain the best metrics and the final model to be integrated with the software.

3.5 Integration

This section describes how the CNN is merged with the software that brings together both the classical approach and the DL approach. The way the CNN approach works is to take the image that is passed to it, generate patches from it and make a prediction on each one of them. Each patch of the image is colored according to its probability of belonging to the control class. In this regard, cyan is used to frame patches with more than 90% probability of belonging to the control class, steel blue is used for patches with probability between 70% and 90%, yellow for patches with probability between 50% and 70%, orange for patches with probability between 30% and 50%, and finally red for patches with less than 30% of probability of belonging to the control class. The percentage of these that are classified as control class is called goodness and is a measure of how quantitatively an image resembles a control image.par

Although the integration has not been very costly since the main script was already done, some adjustments had to be made to make the new version compatible. On the one hand, the patch generation had to be adapted since before they were made without overlap and with a size of

64x64 and now, patches of 224x224 with overlap must be generated. Furthermore, the number of patches has to be a reasonable value so that a failure in one of them does not affect the goodness too much. Moreover, as these patches are displayed in the software next to the original image, a value has to be chosen that allows to visualize the patches in a clear way and to understand which areas of the image may be more difficult to classify.

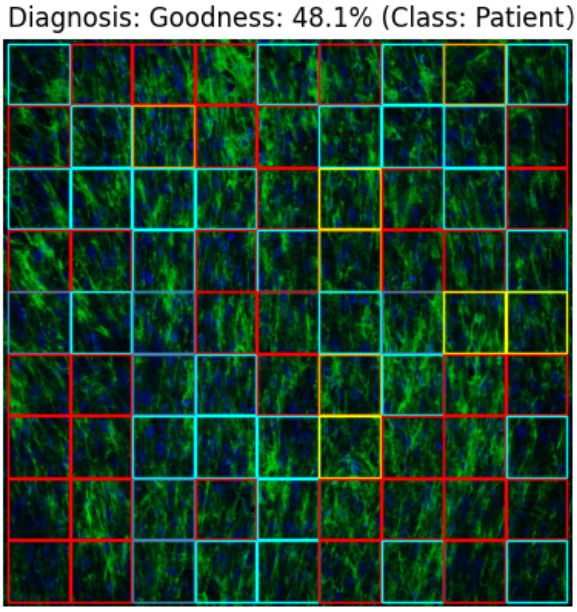

Figure 3.5: Visualization of the diagnosis of a given fibroblast culture image.The system also gives an overall score computed as the percentage of patches classified as control in the image.

4 Results

To analyze the methods explained in section 3 a set of experiments has been performed. It should be noted that not all the results of the implementations mentioned in the previous section will be shown. This is because there have been procedures that despite being useful for the progress of the project, produced poor results. Thus, no results will be given for the method proposed to replicate results of the previous network (although it must be said that the results obtained were very similar) nor for the old network trained with the whole new training set.

The dataset used to develop the following experiments contains 221 images in TIFF format labeled as control and 199 images labeled as patients. Data augmentation is applied to each of these images in order to have much more data available. All techniques have been explained in section 3.

Before starting to explain the experiments that have been carried out, some hyperparameters that have been used in all of them will be explained.

$$Log \quad loss = \frac{1}{N} \cdot \sum_{i=1}^{N} -(y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i)) \tag{2}$$

First of all, the loss function used is the binary cross entropy (equation 2). The loss function indicates the quality of the model predictions. Binary cross entropy looks at each of the predicted probabilities to the actual class performance which can be 0 or 1. That implies how close or far it is from the actual value. This loss function is appropriate when working with binary tasks.

It is also known that too little training for too few epochs will mean that the model will underfit the training, validation and test datasets. Too much training for too many epochs will imply that the model will overfit the training data set and perform poorly on the validation and test sets. A trade off is to train on the training dataset yet to quit training right when execution on a validation dataset begins to degrade. This simple, effective, and generally used way to deal with training neural networks is called early stopping.

During training, the model is evaluated on a validation dataset after every epoch. If the performance of the model on the validation dataset begins to degrade (for instance, loss begins to increase or accuracy begins to decrease), the training process is stopped. Also a trigger for stopping the training process must be chosen. The trigger will use a monitored performance metric to decide when to stop training. More elaborate triggers might be needed in practice. This is because the training of a neural network is stochastic and can be noisy. Plotted on a graph, the performance of a model on a validation dataset may go up and down many times. This means that the first sign of overfitting may not be a good place to stop training. The trigger chosen for this work is to stop after a decrease in performance observed over a given number of epochs, specifically after 10 epochs.

4.1 Training patches of size 64x64

The first experiment was to train the old network with an input size of 64x64x3. By having an image size of 1024x1024x3, it was not necessary to overlap the generated patches because there were enough patches to train and evaluate correctly. Thus, 256 patches were generated for each image in the new dataset resulting in a total of 122112 patches.

The optimizer used was the Adam. The Adam optimizer is a stochastic gradient descent method based on adaptive estimation of first and second order moments. The learning rate is equal to 0.001. The exponential decay rate for the first moment was set to 0.9. While the exponential decay rate for the second moment was set to 0.999. The last parameter for this optimizer is epsilon which is used to avoid any division by zero and the value is 10-8.

Another important hyperparameter to choose is the batch size. Batch size is a term used in deep learning and refers to the number of training examples used in an iteration. Usually the size is smaller than the total number of examples contained in the training set. The two main advantages are that less memory is needed during training and that networks are trained faster with mini-batches. Conversely, the smaller the batch the less accurate the estimate of the gradient will be. For this experiment a batch size equal to 256 has been used.

As mentioned in the section, the kfold method was used for training and testing. In this case, it was decided to use k=5.

Although in the paper presented for the previous work it was stated that less than 10 epochs were needed for the model to converge. For this project it has been decided to train for many epochs in order to have the best possible results. The total number of epochs was 30 and each epoch takes approximately 2 minutes 13 seconds to run.

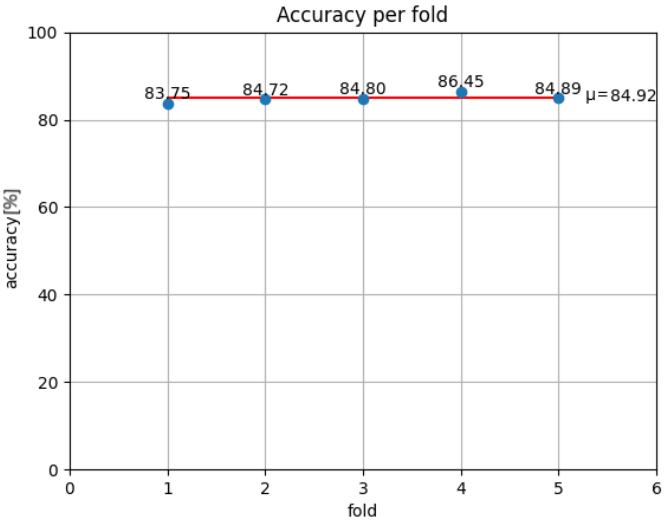

Figure 4.1: Accuracy with input size 64x64

As can be seen in the graph 4.1, the average accuracy obtained in the 5 folds is 85%. This represents a drop of 10% with respect to the same architecture, but with the dataset until 2018. Furthermore, as mentioned above, we have been able to obtain a much more robust model as we have trained with many more examples.

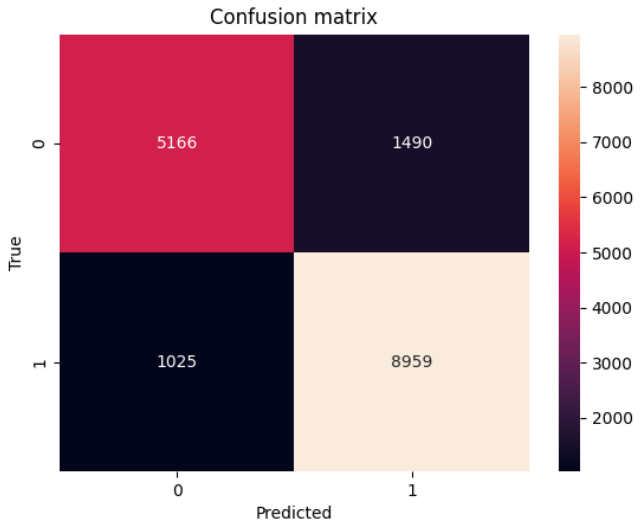

Figure 4.2: Confusion matrix of the test set for the model trained with the 64x64 image patches.

Figure 4.2 gives the confusion matrix of the system diagnosis performance on the 64x64 patches from the test set. In this confusion matrix, the number of patches correctly and incorrectly classified as belonging to the control class is in the first row. From a total of 6656 control patches in the test set, 5166 are true negatives (tn), i.e., the patches correctly classified as control, and 1490 are false positives (fp), i.e., the patches incorrectly classified as patient. The classification of patients is given in the second row, where 1025 are false negatives (fn), i.e., the patches incorrectly classified as control, and 8959 are true positives (tp), i.e., the patches correctly classified as patient. On this confusion matrix, three metrics such as accuracy, precision and recall can be calculated, which help to better understand the system. For this experiment and on this confusion matrix an accuracy of 84.9% is obtained. The recall gives a value of 89.7%, while the precision is 85.7%.

ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. The higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. In other words, the higher the AUC, the better the model is at distinguishing between patients with the disease and no disease.

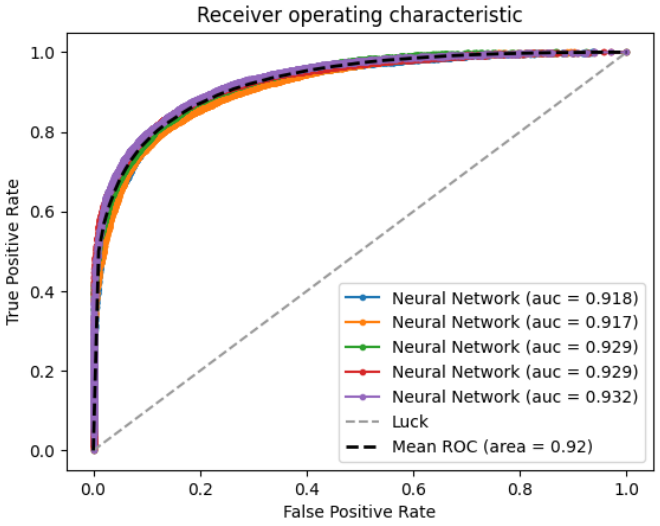

Figure 4.3: ROC with input size 64x64

The graph 4.3 shows the ROC curves obtained for each of the folds. Moreover, the mean ROC curve of the 5 folds has also been computed, which has a mean AUC of 0.92. Despite having a lower accuracy than its previous model, the AUC value has been reduced by 0.03 points.

4.2 Increasing patch size

For this second experiment, the choice of hyperparameters is the same except for the input size. The purpose of this experiment is to check if a larger patch size provides better metrics than in the previous experiment. That is why the patch sizes are now 128x128x3.

If the same method were used to generate patches as in the previous experiment, 64 patches would be created for each image, for a total of 26880 patches. This value is not sufficient if one wants to train a network that is able to generalize correctly. Therefore, many more patches have to be generated. This is done by applying an overlap when generating them. It can be thought of as a window that slides over the image and extracts 128x128 patches. The overlap that has been applied is 50%, which generates a total of 70650 patches.

The optimizer used is Adam with a learning rate of 0.001, batch size is equal to 256. The kfold technique was also used in this experiment to obtain more robust metrics. The total number of epochs used for this experiment is 30 but with larger input sizes, the duration per epoch has been increased to 4 minutes 47 seconds on average.

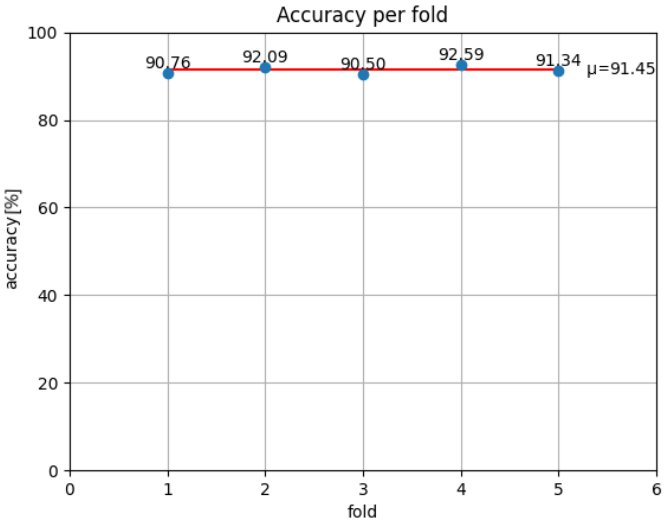

Figure 4.4: Accuracy with input size 128x128

As it can be seen in the figure 4.4 an average accuracy of 91.45% has been obtained in the 5 folds. This increase of approximately 6% compared to the previous experiment confirms that using a larger input size helps the network to generalize better. It can be assured that by using larger patches, the network looks at more global details and not as local as when experimenting with 64x64 size patches.

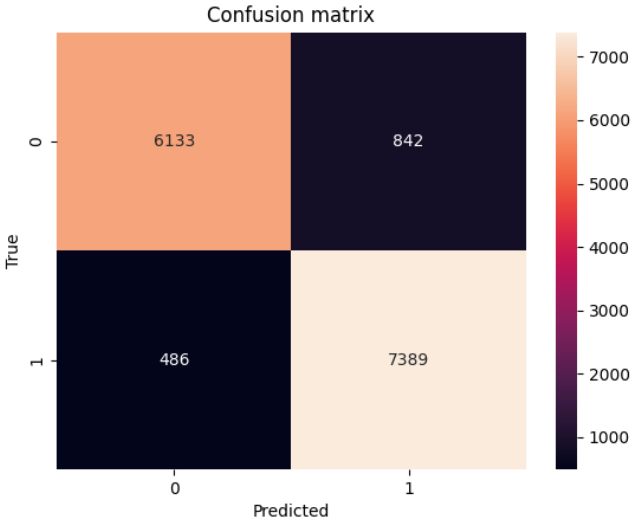

Figure 4.5: Confusion matrix of the test set for the model trained with the 128x128 image patches.

In the confusion matrix observed above Figure 2.3, the quantity of patches accurately and mistakenly named having a place with the control class is in the first row.i.e. the patches correctly classified as control, and 842 are false positives (fp), i.e. the patches incorrectly classified as patient. The classification of the class patients is given in the second row, where 486 are false negatives (fn), i.e., the patches incorrectly classified as class control, and 7389 are true positives (tp), i.e. the patches correctly classified as class patient.

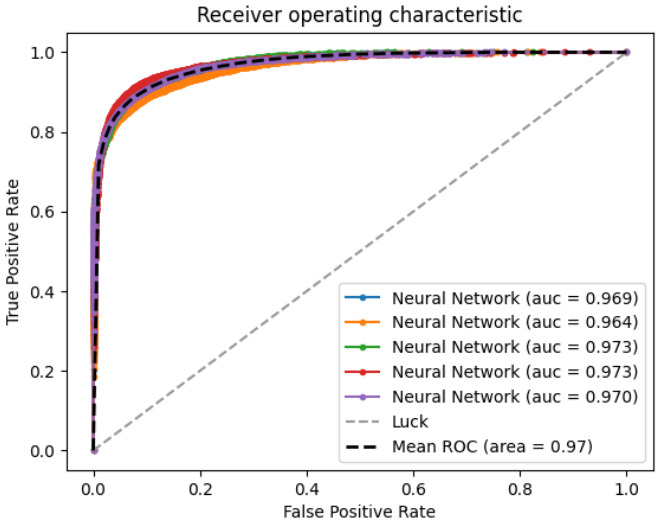

Figure 4.6: ROC with input size 128x128

Observing the ROC figure 4.6, it can be seen that in this case the curve tends to go more towards the point (0, 1) than in the previous experiment. This implies an increase in the AUC which, as shown in the legend, has an average AUC of 0.97.

4.3 Training Efficientnet

The next proposed experiment is to change the architecture used to a pre-trained CNN. The proposed network is the Efficientnet in its smallest model since this is sufficient to be able to train satisfactorily. The input size of this model is 224x224x3. Since the size of 224 pixels is not divisible by that of the complete image (1024 pixels), that is why when extracting patches it is necessary to do it with overlap. If not, those patches that are located to the right of the image and in the lower part would contain black pixels since it would be accessing areas outside the image. In addition, if this overlap were not applied, only 16 patches would be extracted for each image making the total dataset small to be able to train. In the case of this work, an overlap of 174 pixels (approximately 77%) has been applied so that in such a way it can have enough samples to train and test with large enough sets. Exactly 400 patches are being sampled for each image, making the dataset resulting in a total of 168000 patches.

The first experiment performed with this configuration was using the Adam optimizer and with a learning rate of 0.001. The constant learning rate is the default learning rate setting in the Adam optimizer in Keras. However, it is often useful to reduce the learning rate as training progresses when training deep neural networks. This reduces the step size at each iteration while moving toward a minimum of a loss function as epochs are passed. This allows one to train much faster at the beginning and at the end take smaller steps to approach the minimum of the loss function. The way to modify the learning rate during training is by using a scheduler. For the development of this experiment the time-based decay has been used. This implies that the value

of the learning rate is modified after each epoch according to the following expression:

$$\varepsilon_{n+1} = \varepsilon_n \cdot (\frac{n}{n + \varepsilon_n}) \tag{3}$$

, with $\varepsilon_n = 0.01$ and where n is the total number of epochs.

Two experiments were carried out using the Efficientnet. In the first one, the whole network has been trained without freezing the already trained layers. The batch size chosen for this experiment is 128.

In order to show robust results, it is not enough just to perform a single training and show the metrics obtained from it. What is proposed is to perform 10 independent trainings where the test set is different in each of them and then show the mean over these 10 trainings.

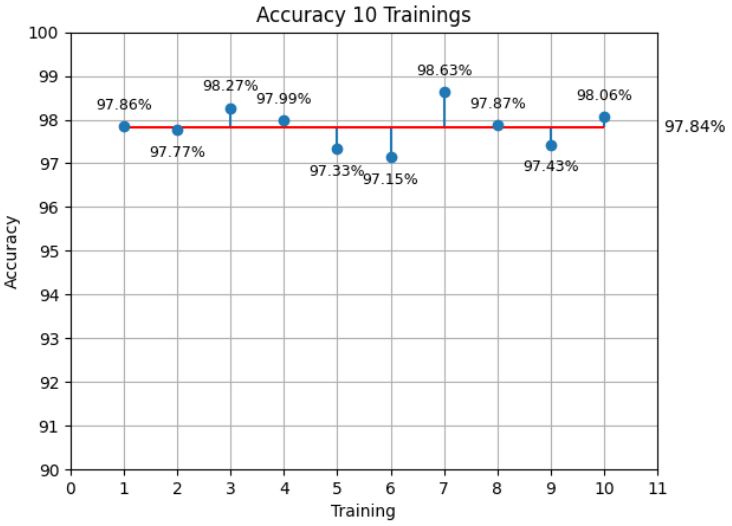

Figure 4.7: Accuracy with EfficientnetB0

The graph 4.7 shows that the system is able to achieve an average accuracy in the 10 training sessions of 97.8%. It can be concluded that using for this project a network that has been previously pre-trained has a better performance than training the old network from scratch. It is also observed that increasing the input size causes the accuracy to increase, this trend already happened in the experiment using 128x128 patches.

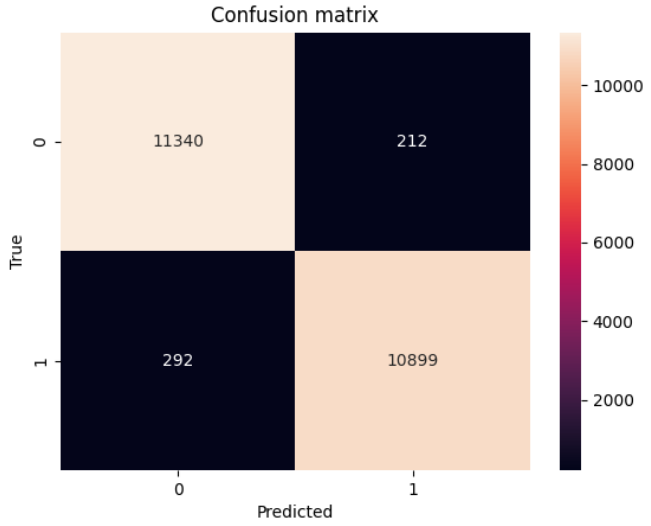

Figure 4.8: Confusion matrix of the test set for the model trained with EfficientnetB0.

In Figure 4.8, the confusion matrix is calculated with test set for the model trained with the EfficientnetB0. when the accuracy is calculated, the result is a 97.78%, which indicates that the network does perform very well in classifying correctly the majority of patches. The precision obtained is 98.1%, while the recall is 97.4%. It can be concluded that it is the model that performs best with a low number of missclassification, i.e., false positives, or false negatives.

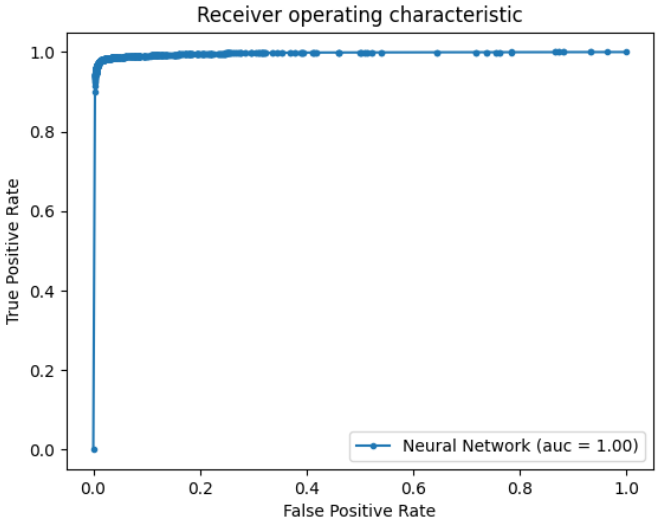

Figure 4.9: ROC using EfficientnetB0

In the ROC curve obtained in this experiment, it can be observed that the curve is practically the ideal curve, which approaches the vertex (0, 1). It can be affirmed, observing results different from those already mentioned, that this CAD system offers a very good classification. It can also be observed that the AUC for this experiment is practically 1.

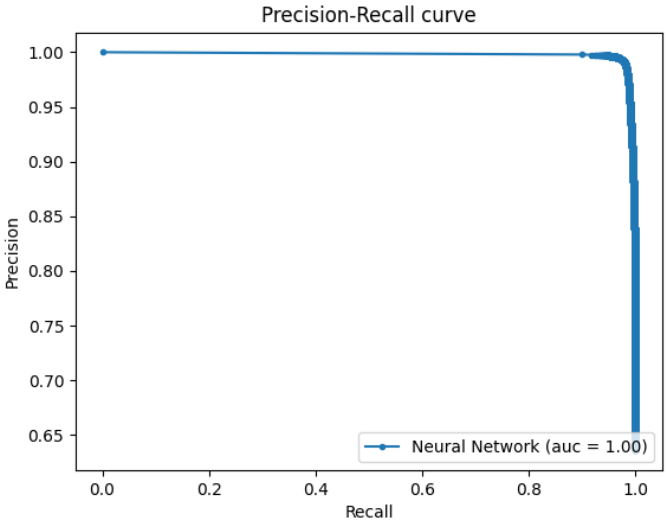

Figure 4.10: Precision-Recall curve using Efficientnet on the test set.

In the Precision-Recall curve (see figure 4.10) obtained in this experiment, it can be observed that the curve is practically the ideal curve, which approaches the vertex (1, 1). It can be affirmed, observing results different from those already mentioned, that this CAD system offers a very good classification because it maintain both a high precision and high recall across the graph. It can also be observed that the AUC for this experiment is practically 1.

The second experiment that has been carried out with the Efficientnet has frozen the already trained layers and only the layers that were added at the end are trained to adapt it to our case. The total number of epochs was 30 and each epoch takes approximately 212 seconds to run. Since in this experiment the layers of the EfficientnetB0 have been frozen, the weights associated with its neurons are not updated and this makes the execution time up to 6 times shorter compared to the experiment in which the whole network was trained.

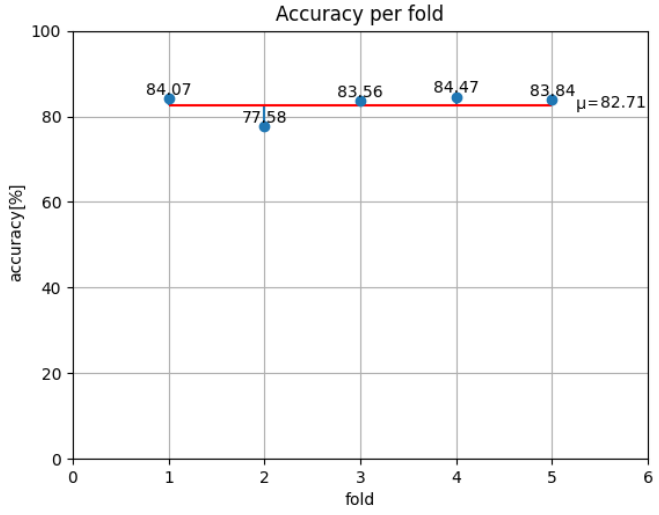

Figure 4.11: Accuracy with freezing the EfficientnetB0 layers.

It can be seen graph 4.11 that the average accuracy obtained in the kfold is 82.7%. Therefore, it is concluded that applying the transfer learning technique for this task does not offer optimal results. This is understandable since the dataset used to train the Efficientnet and the one used in this project are very different so it is also necessary to train the pre-trained network to learn features from the images of our dataset.

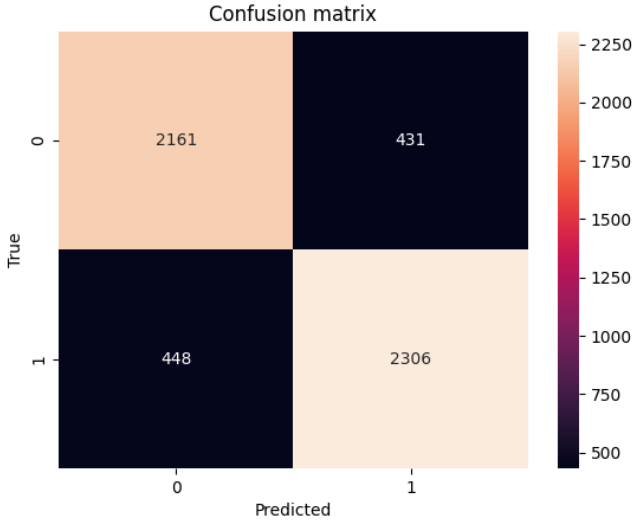

Figure 4.12: Confusion matrix of the test set for the model trained with EfficientnetB0 freezing its layers.

Figure 4.12 gives the confusion matrix of the system diagnosis trained on the Efficientnet and freezing its layers from the test set. In this confusion matrix, the number of patches correctly and incorrectly classified as belonging to the control class is in the first row. From a total of 2592 control patches in the test set, 2161 are true negatives (tn) and 431 are false positives (fp). In the second row, 448 patches are false negatives (fn) and 2306 are true positives (tp).

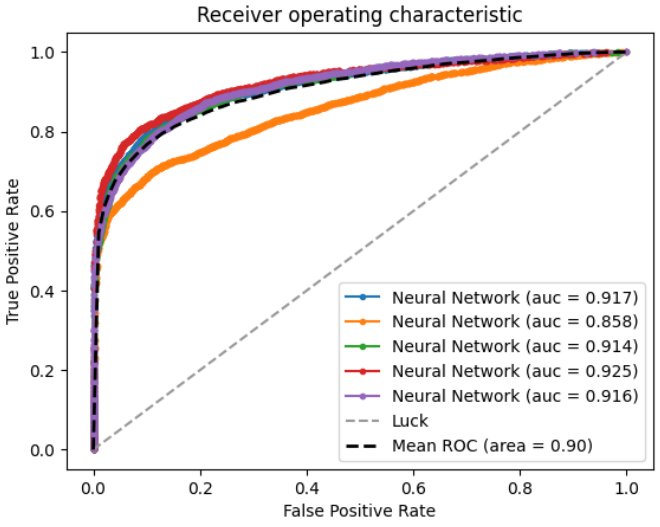

Figure 4.13: ROC with freezing the EfficientnetB0 layers.

The graph 4.13 shows the ROC curves obtained for each of the folds. Moreover, the mean ROC curve of the 5 folds has also been computed, which has a mean AUC of 0.90.

4.4 Comparison between networks

In order to compare the old network with the Efficientnet, an experiment has been performed where the hyperparameters are the same. The only difference, as mentioned above, is the architecture used. Although it is difficult to surpass the results obtained with the Efficientnet, it is interesting to know how much difference there is between the models in terms of accuracy and execution time.

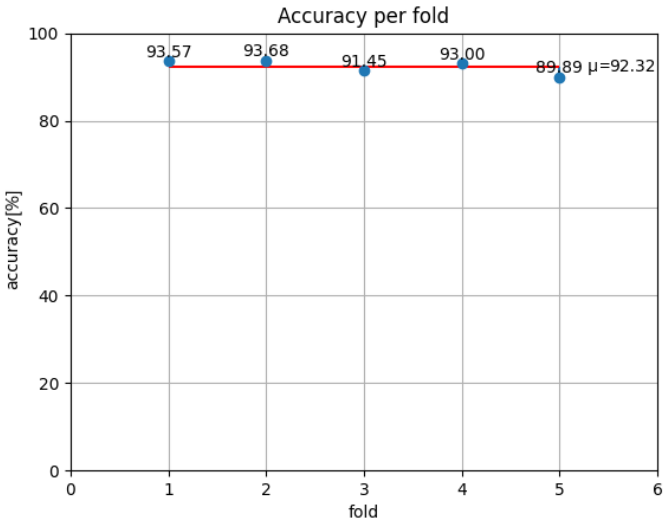

Figure 4.14: Accuracy with input size 224x224 with old network.

Observing the graph it can be said that the average accuracy obtained in the 5 folds is 92.3%. This value is closer to the result obtained in the experiment where this same network was trained with 128x128 patches than to the result obtained with the EfficientnetB0 with fine tuning. It can be concluded that for the task we are dealing with in this project, the use of a pre-trained network such as EfficientnetB0 has a better performance than the previously used network.

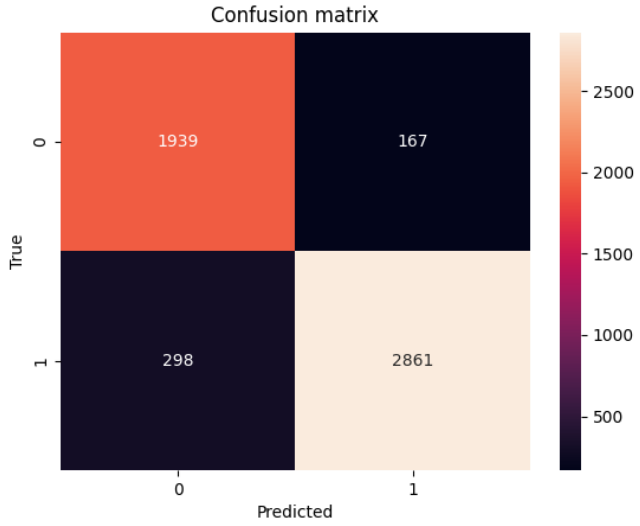

Figure 4.15: Confusion matrix using old network and input patches of 224x224.

As can be seen in the confusion matrix 4.15, the number of true negatives is 1939, while the number of false positives is 167. In the bottom row, it can be seen that there are 298 patches classified as negative class and 2861 patches correctly classified as positive class.

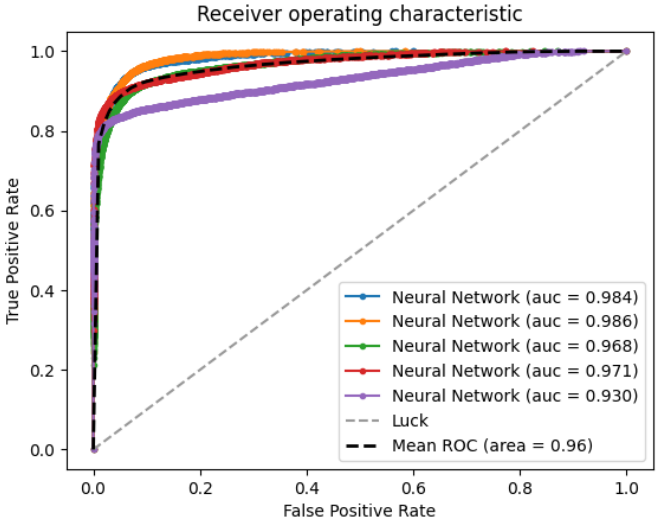

Figure 4.16: ROC with input size 224x224 with old network.

The graph 4.16 shows the ROC curves obtained for each of the folds. Moreover, the mean ROC curve of the 5 folds has also been computed, which has a mean AUC of 0.96.

4.5 Training with the whole Dataset

The last experiment performed was to train Efficientnet with the entire dataset, i.e., taking the pediatric patients used in the previous experiments and adding those that were discarded. In total there were 477 images of which 221 belonged to controls and 256 to patients.

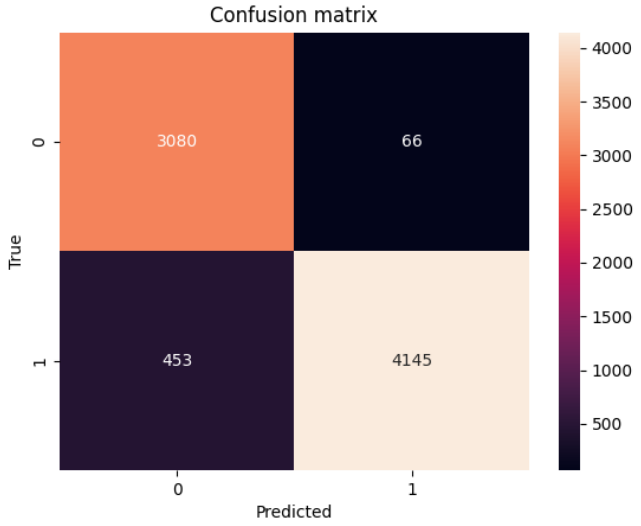

Figure 4.17: Confusion matrix using Efficientnet on the test set.

Figure 4.17 gives the confusion matrix of the system diagnosis performance on the test set when training the Efficientnet on the whole dataset, including the doubtful subjects. From a total of 3146 control patches in the test set, 3080 are true negatives (tn) and 66 are false positives (fp). The classification of patients is given in the second row, where 453 are false negatives (fn) and 4145 are true positives (tp). On this confusion matrix, three metrics such as accuracy, precision and recall can be calculated, which help to better understand the system. For this experiment and on this confusion matrix an accuracy of 93.3% is obtained. The recall gives a value of 98.4%, while the precision is 90.1%.

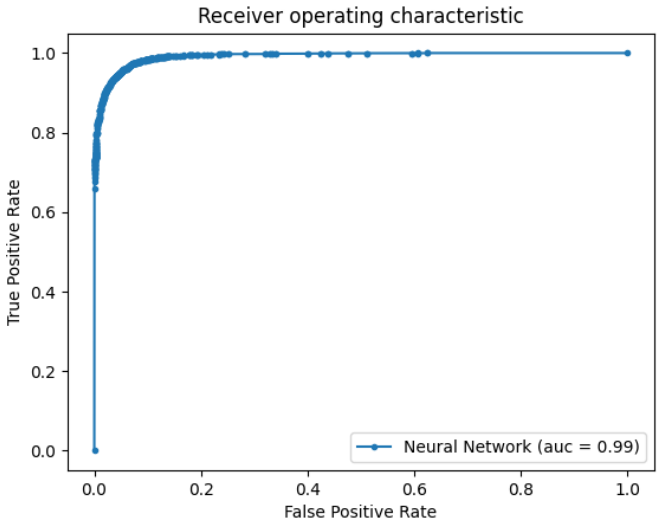

Figure 4.18: ROC using Efficientnet on the test set.

In the ROC curve obtained in this experiment, it can be observed that the curve is practically the ideal curve, which approaches the vertex (0, 1). It can be affirmed, observing results different

from those already mentioned, that this CAD system offers a very good classification. It can also be observed that the AUC for this experiment is practically 1.

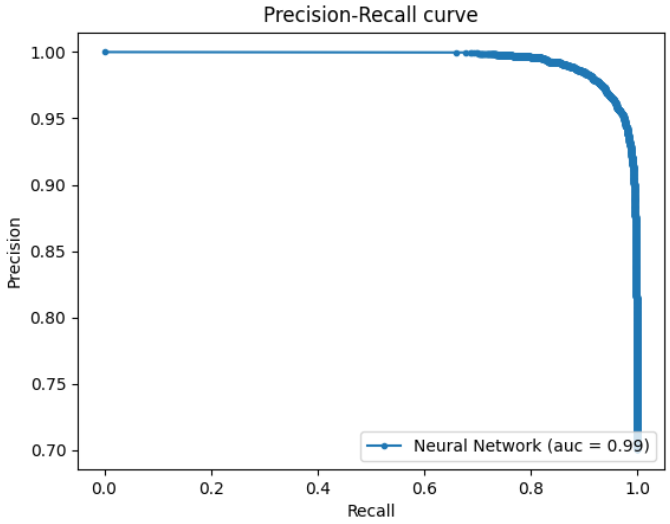

Figure 4.19: Precision-Recall curve using Efficientnet on the test set.

Figure 4.19 shows the Precision-Recall curve on the test set obtained when Efficientnet is trained with the whole dataset. A good classifier like this one maintain both a high precision and high recall across the graph, and will "hug" the upper right corner in the figure below. The AUC for this curve is 0.99, slightly less than the previous P-R curve (see figure 4.10)

As it can be seen, the accuracy when training with the whole dataset has decreased by 2% with respect to the model where doubtful cases were discarded. Even so, a model with similar metrics to the model of the previous version is obtained.It is proposed to use the best model obtained with the EfficientnetB0 to predict on the whole dataset and to extract a confusion matrix at the individual level. Each person has associated TIFF images, in general 10 images, then a prediction is made on each of these images and the majority voting method is applied to determine the final class of the person. Once the class is obtained for each person, the confusion matrices are extracted.

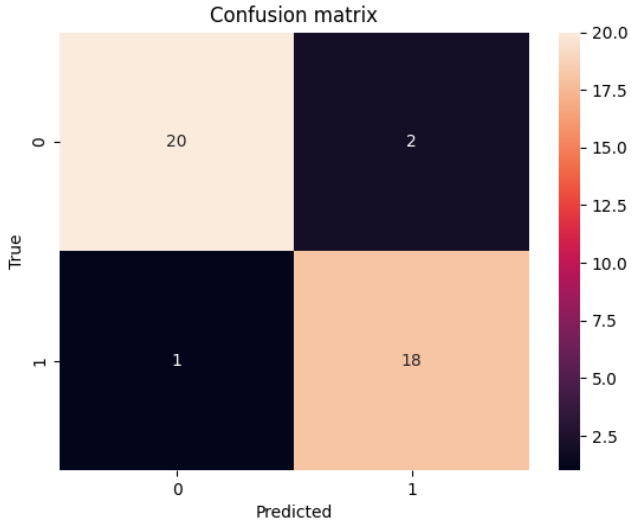

Figure 4.20: Confusion Matrix at person level without counting doubtful cases.

In figure 4.20, the confusion matrix is given by the dataset that does not contain the borderline cases. If the accuracy is calculated, 92.9% is obtained. It can be seen that there are 2 subjects who are predicted to be ill when labeled as controls. The precision obtained is 90%. Despite being an error, labeling a healthy person is not a serious problem. The serious problem is in the person who has been predicted as belonging to the control class when in fact it is a subject labeled as sick. This error causes a recall of 94.7%.

The 7 doubtful cases that had been ruled out for training are added below.

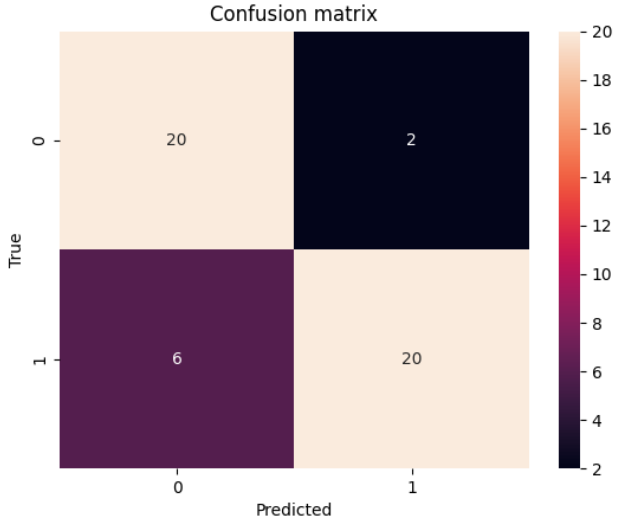

Figure 4.21: Confusion Matrix at person level counting doubtful cases.

In figure 4.21, the confusion matrix is calculated with the data set containing the borderline cases. If the accuracy is calculated, 83.3% is obtained, which indicates that the network does not perform very well in detecting these rare cases. It can be observed that there are still 2 subjects who are predicted to be ill when labeled as controls. The precision obtained is 90.9%. It is observed that the people predicted as controls when they are sick subjects has grown. This means that most of the doubtful cases are being classified as healthy persons. This classification results in a recall of 76.9% which represents a very serious error since the system does not allow to correctly classify rare cases of rare mutations or secondary phenotypes.

Finally, we wanted to check what happened if a prediction was made on the set containing images of adults. These images, being different from those of the pediatric cases, cannot be mixed to train a single model that makes predictions about the two types of subjects. If this were done, it would result in a model that would not be able to correctly classify either the pediatric or adult cases. That is why we propose to compute the confusion matrix on this set of adults and observe what the network trained with pediatric cases predicts.

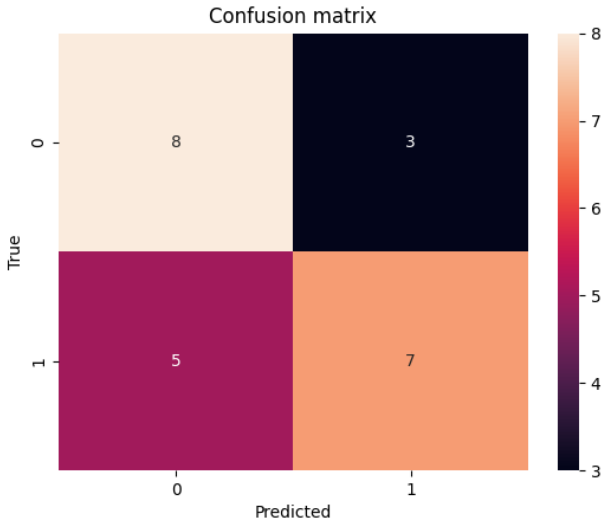

Figure 4.22: Confusion Matrix at person level of adult cases.

As can be seen in figure 4.22, the model is able to correctly predict 15 cases, 7 cases for the positive class and 8 for the negative class, making the accuracy at the individual level of 65.2%. These results are very poor for this type of task. In terms of precision, 70% is obtained. The problem relies in the recall of 58.3% which causes sick people to be predicted as healthy subjects; this can be caused because patient subjects can be mild sick cases.

5 Budget

This project consists of an improvement of a system capable of making an automatic evaluation of collagen IV deficiencies, so no prototypes are designed or built. When computing the total costs only the software and material used will be considered.

The budget includes, not only the income for the manpower devoted to the project but also the exceptional expenses performed for the fulfilment of the project as well as the proportional part of the annual expenses of the business based on the time the development has taken.

The entire project has been developed in Python, the text editor used is Pycharm which is a free software to interpret that language and Matlab which has a license fee. So to carry out the work, it has been needed the material shown in table 5.1:

Concept	**Estimated cost**
Computer	1200€
GPI Server	0.35€/h[1]
Matlab Student License	35€

Table 5.1: Material costs of the project.

As it has been developed in the author's house due to the existence of COVID-19, no office or space has been rent and its costs have been null. The manpower cost is in table 5.2:

Concept	**Cost/hour**	**Hours dedicated**	**Total cost**
Junior Engineer	13€/hour	550 hours	7150.00€
Senior Engineer	60 €/hour	52 hours	3120.00€

Table 5.2: Human costs of the project.

Adding the manpower cost to the material costs, both shown in tables 5.1 and 5.2 respectively, we have the total budget needed to make this project. So, the total cost has a value of **15325€** approximately.

Due to the fact of working on a simulations-based project, the computed total cost is not that high. This means that, although making an investment is needed to cover the required payments and proceed with the study of the project, it is economic viable. When comparing the costs produced by the project with the technology advance that it can provide, its relation is proportional. So as the impact of this study is that important, the cost estimated it coherent, and therefore feasible.

[1]The cost has been estimated through the Google calculator.
Total number of hours = 2000

6 Conclusions and Future Development

The first version of the software used a CNN created by the author himself[6], which was developed with a total of 276 images corresponding to the two classes. As there were few images, it was decided to divide them into patches so that more data could be generated and used to train the network. The size of these patches was 64x64. After analysis and optimization of hyperparameters, a model was obtained that was able to correctly classify 95% of the patches, which means that practically all images, if not all, were correctly classified.

This system works well for images that resemble those with which the network was trained, but if inference is made on weird cases it may not get the class right and therefore may not perform the automatic diagnosis well. That is why it was necessary to retrain this model with new cases so that it is able to correctly classify many new cases.

Therefore, in this work it has been decided to retrain the network with many more cases that have been collected in the last few years. In addition, an exhaustive analysis has been developed to try to improve the existing network and to obtain better metrics. To improve the previous CAD system, we relied on Deep Learning technologies.

In this project we have proposed and experimented with pre-trained networks to see if they offer better results than those obtained in the previous version. For this task we have used the Efficientnet, a state of the art network that offers better results than other pre-trained networks. The best model obtained has been using the smallest model of the Efficientnets family, the EfficientnetB0, which has provided an average accuracy of 97.84% in 10 training iterations.

With this model, the objectives set at the beginning of the project have been met. On the one hand, a more robust system has been obtained since it has been trained with about 200 more images. On the other hand, we have achieved a more accurate model than the previous version, which results in a low number of missclassification, i.e., false positives, or false negatives.

Some conclusions can be drawn from the work carried out. The two architectures used have been built for different purposes. While the old network was built specifically for this project and adapted to it, the Efficientnet aims at a more general classification since it was trained with 1000 different classes. That is why while the former started from randomly initialized weights, the latter already had weights associated with it. In this way, it has been proved that, for this work, the pre-trained network has achieved higher metrics than those obtained with the old network. Moreover, although the set of images has been expanded with respect to that used in the first version of the software, it is still insufficient to train the network with complete images. That is why a very interesting strategy to generate new data is to generate patches from them. Also the EfficientnetB0 is constrained to work with 224x224 images. In this way, a sufficient amount of data is obtained to avoid overfitting and underfitting and to obtain a satisfactory performance

without having to rescale the original images.

Although several methods and techniques have been developed in order to have a model capable of correctly predicting this disease, there are some studies that are still pending and that it would be interesting to develop in order to draw more conclusions when diagnosing this disease.

As mentioned above, each LIF image obtained by the confocal microscope contains several image planes of the sample. In our development, a projection of these planes is applied to obtain a single TIFF image. It would be interesting, instead of projecting all the planes in a single image, to be able to work with each of these planes separately to obtain multichannel images. For example, if this image contains 10 planes, an image with 20 channels would be obtained and fed to the network for training.

On the other hand, in the data set provided by the hospital, there was also a group of images with higher resolution. These images differed from the ones used in this project in the number of planes extracted per sample in addition to the resolution. While in the ones used to develop this project, the magnification was 20x, the high resolution ones have a 100x magnification. As mentioned above, in the 20x images, 10 2D planes are obtained. The 100x images represent a volume of the sample taken from the subject's forearm. As this set was small, it was thought that it would be difficult to work well with, but in the future, when more images can be obtained, it would be good to be able to develop a model to deal with these images as, according to the doctors, they contain much more information than those considered so far.

Bibliography

[1] A. Nadeau, M. Kinali, M. Main, C. Jimenez-Mallebrera, A. Aloysius, E. Clement, B. North, A. Y. Manzur, S. A. Robb, E. Mercuri, and F. Muntoni. "Natural history of Ullrich congenital muscular dystrophy". In: 73.1 (2009), pp. 25–31. DOI: https://doi.org/10.1212/WNL.0b013e3181aae851.

[2] S. R. Lamandé and J. F. Bateman. "Collagen VI disorders: Insights on form and function in the extracellular matrix and beyond". In: 71-72 (2018), pp. 348–367. DOI: https://doi.org/10.1016/j.matbio.2017.12.008.

[3] J. S. Amberger, C. A. Bocchini, F. Schiettecatte, A. F. Scott, and A. Hamosh. ", OMIM.org: Online mendelian inheritance in man (OMIM®), an online catalog of human genes and genetic disorders". In: 43 (2014).

[4] C. Jimenez-Mallebrera, M. Maioli, J. Kim, S. Brown, L. Feng, A. Lampe, K. Bushby, D. Hicks, K. Flanigan, C. Bonnemann, C. Sewry, and F. Muntoni. "A comparative analysis of collagen vi production in muscle, skin and fibroblasts from 14 Ullrich congenital muscular dystrophy patients with dominant and recessive col6a mutations". In: 16.9 (2006), 571–582. DOI: https://doi.org/10.1016/j.nmd.2006.07.015.

[5] K. Anthony, V. Arechavala-Gomeza, L. E. Taylor, A. Vulin, Y. Kaminoh, S. Torelli, L. Feng, N. Janghra, G. Bonne, M. Beuvin, R. Barresi, M. Henderson, S. Laval, A. Lourbakos, G. Campion, V. Straub, T. Voit, C. A. Sewry, J. E. Morgan, K. M. Flanigan, and F. Muntoni. "Dystrophin quantification". In: 83.22 (2014), 2062–2069, doi=https://doi.org/10.1212/WNL.0000000000001025.

[6] Adrián Bazaga, Mònica Roldán, Carmen Badosa, Cecilia Jiménez-Mallebrera, and Josep M. Porta. "A Convolutional Neural Network for the automatic diagnosis of collagen VI-related muscular dystrophies". In: *Applied Soft Computing* (2019). DOI: https://doi.org/10.1016/j.asoc.2019.105772. URL: http://www.sciencedirect.com/science/article/pii/S1568494619305538.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: 1.15 (2012), 1097–1105. URL: http://dl.acm.org/citation.cfm?id=2999134.2999257.

[8] Daniel Berrar. "Cross-Validation". In: (2018). DOI: https://doi.org/10.1016/B978-0-12-809633-8.20349-X.

[9] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Network". In: *arXiv preprint arXiv:1905.11946v5* (2020).

[10] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. "AutoAugment: Learning Augmentation Policies from Data". In: *arXiv preprint arXiv:1805.09501v3* (2019).

[11] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks For Large-Scale Image Recognition". In: *arXiv preprint arXiv:1409.1556v6* (2015).

[12] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. "Rethinking the Inception Architecture for Computer Vision". In: *arXiv preprint arXiv:1512.00567v3* (2015).

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *arXiv preprint arXiv:1512.03385v1* (2015).

[14] Yin, Xiangnan, Chen, Weihai, Wu, Xingming, Yue, and Haosong. "Fine-tuning and visualization of convolutional neural networks". In: *2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. 2017, pp. 1310–1315. DOI: 10.1109/ICIEA.2017.8283041.

[15] In: ().