



Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Technical University of Cartagena

School of Industrial Engineering

Development and Research of Skin Lesion Detection and Classification Method

Bachelor's Final Degree Project

María Jesús García Aparicio

Project author

Prof. Dr. Vidas Raudonis

Assoc Prof. Dr. Miguel Almonacid Kroeger

Supervisors

Cartagena, Kaunas 2021



Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Technical University of Cartagena

School of Industrial Engineering

Development and Research of Skin Lesion Detection and Classification Method

Bachelor's Final Double Project
Intelligent Robotics Systems (6121EXO013)
Industrial Electronics and Automation Engineering (5071)

María Jesús García Aparicio

Project autor

Prof. Dr. Vidas Raudonis

**Assoc. Prof. Dr. Miguel Almonacid
Kroeger**
Supervisors

Lect Dr. Julius. Gelšvartas

Reviewer

Cartagena, Kaunas 2021



Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Technical University of Cartagena

School of Industrial Engineering

María Jesús García Aparicio

Development and Research of Skin Lesion Detection and Classification Method

Declaration of Academic Integrity

I confirm the following:

1. I have prepared the final degree project independently and honestly without any violations of the copyrights or other rights of others, following the provisions of the Law on Copyrights and Related Rights of the Republic of Lithuania, the Regulations on the Management and Transfer of Intellectual Property of Kaunas University of Technology (hereinafter – University) and the ethical requirements stipulated by the Code of Academic Ethics of the University;
2. All the data and research results provided in the final degree project are correct and obtained legally; none of the parts of this project are plagiarised from any printed or electronic sources; all the quotations and references provided in the text of the final degree project are indicated in the list of references;
3. I have not paid anyone any monetary funds for the final degree project or the parts thereof unless required by the law;
4. I understand that in the case of any discovery of the fact of dishonesty or violation of any rights of others, the academic penalties will be imposed on me under the procedure applied at the University; I will be expelled from the University and my final degree project can be submitted to the Office of the Ombudsperson for Academic Ethics and Procedures in the examination of a possible violation of academic ethics.

María Jesús García Aparicio

APPROVED BY:

KTU Faculty of Electrical and Electronics

Head of the Department of Automation

Assoc. prof. dr. Gintaras Dervinis

2021 05 04

TASK OF FINAL PROJECT OF UNDERGRADUATE (BACHELOR) STUDIES

Issued to the Student:	<i>María Jesús García Aparicio</i>	Group	ERB7/2
1. Project Subject:			
Lithuanian Language:	Odos apgamo aptikimo ir klasifikavimo metodo sukūrimas ir tyrimas		
English Language:	Development and Research of Skin Lesion Detection and Classification Method		

Approved 2021 May . 6d. Decree of Dean Nr. V25-03-8

2. Goal of the Project:	To develop and investigate the skin lesion detection and classification method
--------------------------------	--

3. Specification of Final Project:	The work must meet the methodological requirements for the preparation of final projects of KTU Faculty of Electrical and Electronics Engineering.
---	--

4. Project's Structure. *The content is concretized together with supervisor, considering the format of the final project, which is listed in 14 and 15 points of Combined Description of Preparation, Defence and Keeping of Final Projects Methodical Requirements*

- | |
|--|
| <p>4.1 Analyze the state-of-the-art literature about skin lesion detection and classification algorithms.</p> <p>4.2 Provide detailed information about dangerous lesion classification.</p> <p>4.3 Apply selected skin lesion segmentation methods to skin images of known database.</p> <p>4.4 Develop skin lesion classification algorithm and test it.</p> |
|--|

5. Economical Part. *If economical substantiation is needed; content and scope is concretized together with supervisor during preparation of final projects*

None

6. Graphic Part. *If necessary, the following schemes, algorithms and assembly drawings; content and scope is concretized together with supervisor during preparation of final projects*

Show algorithms and mathematical reasoning
--

5. This Task is Integral Part of Final Project of Undergraduate (Bachelor) Studies

6. The Term of Final Project Submission to Defense Work at a Public Session of Qualification Commission. *until 2020-05-11*

Received this task:	<i>María Jesús García Aparicio</i>	<i>2021-03-02</i>
	<i>(student's name, surname, signature)</i>	<i>(date)</i>
Supervisor:	Assoc. Prof. Dr. Miguel Almonacid Kroeger	<i>2021-03-02</i>
	Prof. Dr. Vidas Raudonis	
	<i>(position, name, surname, signature)</i>	<i>(date)</i>

García Aparicio, María Jesús. Development and Research of Skin Lesion Detection and Classification Method. Bachelor's Final Degree Project / supervisors Prof. Dr. Vidas Raudonis and Prof. Dr. Miguel Almonacid Kroeger; Faculty of Electrical and Electronics Engineering, Kaunas University of Technology; School of Industrial Engineering; Technical University of Cartagena.

Study field and area (study field group): electronics and automation engineering, technological science, robotic engineering.

Keywords: Computer Vision, skin lesion classification, melanoma, U-net

Kaunas, Cartagena, 2021. p.97

Summary

The aim of the final project is to develop a method for the detection and classification of skin moles. The written work of the bachelor's project consists of an introduction, in which the reader is introduced to the problems to be solved, an analysis of the latest literature, which provides information on similar solutions. The project pays great attention to the mathematical substantiation of the proposed method, which explains the principles of digital image segmentation and the classification of moles using the ABCD rule. The section of the proposed method details the method of skin mole detection and classification, which consists of three main steps, i.e. 1) segmentation of skin moles using UNet deep neural network, 2) extraction of marks from segmented images, and 3) classification of skin moles. The experimental research section of the bachelor's final project presents the results of the accuracy of the proposed skin mole detection and classification algorithm. At the end of the project, the obtained conclusions and the list of used literature are presented.

García Aparicio María Jesús. Desarrollo e Investigación del Método de Clasificación y Detección de Lesiones Cutáneas. Trabajo de Fin de Grado / supervisores: Prof. Dr. Vidas Raudonis, Prof. Dr. Miguel Almonacid Kroeger; Universidad Tecnológica de Kaunas, Facultad de Ingeniería Eléctrica y Electrónica; Universidad Politécnica de Cartagena, Facultad de Ingeniería Industrial

Campo de estudio y área: ingeniería electrónica y automática, ciencia de la tecnología, ingeniería robótica.

Palabras Clave: Visión artificial, clasificación de lesiones cutáneas, melanoma, U-net

Kaunas, Cartagena 2020, 97 páginas.

Sumario

El objetivo del proyecto final es desarrollar un método para la detección y clasificación de lunares cutáneos. El presente trabajo fin de grado consta de una introducción, en la que se presenta al lector los problemas a resolver, un análisis de la última literatura, que aporta información sobre soluciones similares. El proyecto presta gran atención a la fundamentación matemática del método propuesto, que explica los principios de la segmentación de imágenes digitales y la clasificación de lunares utilizando la regla ABCD. La sección del método propuesto detalla el método de detección y clasificación de lunares cutáneos, que consta de tres pasos principales 1) segmentación de lunares cutáneos utilizando la red neuronal profunda UNet, 2) extracción de marcas de imágenes segmentadas y 3) clasificación de lunares cutáneos. La sección de investigación experimental del trabajo fin de grado presenta los resultados de la precisión del algoritmo de clasificación y detección de lunares cutáneos propuesto. Al final del proyecto se presentan las conclusiones obtenidas y el listado de literatura utilizada.

García Aparicio, María Jesús. Odos pažeidimų nustatymo ir klasifikavimo metodo sukūrimas ir tyrimas. Bakalauro baigiamojo darbo projektas / vadovai prof. Dr. Vidas Raudonis ir prof. Dr. Miguel Almonacid Kroeger; Kauno technologijos universiteto Elektros ir elektronikos inžinerijos fakultetas; Pramonės inžinerijos mokykla; Kartachenos technikos universitetas.

Studijų kryptis ir sritis (studijų krypčių grupė): elektronikos ir automatikos inžinerija, technologijos mokslas, robotų inžinerija.

Reikšminiai žodžiai: Kompiuterinis matymas, odos pažeidimų klasifikacija, melanoma, U-net

Miestas, metai. Puslapių sk. 97p.

Santrauka

Baigiamojo projekto tikslas yra sukurti odos apgamų aptikimo ir klasifikavimo metodą. Bakalauro projekto rašto darbas yra sudarytas iš įvado, kuriame skaitytojas yra supažindinamas su sprendžiama problematika, naujausios literatūros analizės, kurioje pateikiama informacija apie panašius sprendimo būdus. Projekte yra skiriamas didelis dėmesys matematiniam siūlomo metodo pagrindimui, kuriame paaiškinami skaitmeninių vaizdų segmentacijos principai, apgamų klasifikavimas taikant ABCD taisyklę. Siūlomo metodo skyriuje yra detalai pristatomas odos apgamų aptikimo ir klasifikavimo metodas, kuris susideda iš trijų pagrindinių žingsnių, t.y., 1) odos apgamų segmentacija naudojant UNet gilųjį neuronų tinklą, 2) pažymių ištrauktimas iš segmentuotų vaizdų ir 3) odos apgamų klasifikavimas. Bakalauro baigiamojo projekto eksperimentinių tyrimų skyriuje yra pateikiami pasiūlyto odos apgamų aptikimo ir klasifikavimo algoritmo tikslumo rezultatai. Projekto gale yra pristatomos gautos išvados, naudotos literatūros sąrašas ir priedai.

Table of Contents

List of figures	10
List of tables	12
List of abbreviations and terms	14
1. Introduction	15
1.1. Medical Motivation	15
1.2. The 2030 Agenda for Sustainable Development	19
1.3. Objectives	21
2. State of Art	22
2.1. Convolutional Neural Network.....	22
2.1.1. “Melanoma Detection by Analysis of Clinical Images using Convolutional Neural Network” by E. Nasr-Esfahani [30]	22
2.1.2. “Acral Melanoma Detection using a Convolutional Neural Network for Dermoscopy Images” by Yu C, Yang S, Kim W, Jung J, Chung KY, et al. [31].....	23
2.1.3. “Skin Lesion Analysis Towards Melanoma Detection via End-to-end Deep Learning of Convolutional Neural Networks” by Katherine M. Li and Evelyn C. Li, et al. [32].....	24
2.2. ABCD Rule	25
2.2.1. “Melanoma Skin Cancer Detection based on Image Processing” by Zghal, N. S., & Derbel, N. [1].	25
2.2.2. “Melanoma Recognition Framework based on Expert Definition of ABCD for Dermoscopic Images” by Abbas, Q., Emre Celebi, M., Garcia, I. F., & Ahmad, W. [33].....	25
2.2.3. “Classification of Malignant Melanoma and Benign Skin Lesions: Implementation of Automatic ABCD Rule” by Reda Kasmi and Karim Mokrani. [34].....	25
3. General Theoretical Concepts.....	27
3.1. Computer Vision.....	27
3.1.1. Digitalization of the Images	27
3.1.2. Colour Spaces	28
3.1.3. Operation on the Histogram and Filtering the Image	30
3.1.4. Operation on the Histogram and Filtering the Image	32
3.1.5. Contrast Enhancement.....	35
3.2. U-Net Architecture	36
4. Programming Language and Environment.....	43
4.1. Python.....	43
4.2. Google Colaboratory	44
5. Proposed Method.....	45
5.1. Image Acquisition.....	47
5.2. Image Pre-processing.....	47
5.3. Segmentation.	53
5.4. Feature Extraction.....	58
5.4.1. Image Preparation.....	58
5.4.2. Calculation of Asymmetry.....	67
5.4.2.1 Calculation of Form Asymmetry	67
5.4.2.2 Calculation of Colour Asymmetry.....	73
5.4.3. Calculation of Border Irregularity	76

5.4.4. Calculation of Colour	79
5.4.5. Calculation of Diameter	84
5.5. Classification.	85
6. Experimental Investigation.	87
Conclusions and Results.....	93
List of references	94

List of figures

Fig. 1. Incidence of melanoma in the USA according to SEER Cancer Statistics Review 1975 - 2009	15
Fig. 2. Incidence trend by periods, comparing Lithuania and Murcia [4]	16
Fig. 3. Estimated age-standardized incidences rates in 2020, both sexes and all ages, non-melanoma skin cancer [6].....	17
Fig. 4. Estimated age-standardized incidences rates in 2020, both sexes and all ages, melanoma skin cancer [6]	17
Fig. 5. Estimated age-standardized mortality rates in 2020, both sexes and all ages, non-melanoma skin cancer [6].....	18
Fig. 6. Estimated age-standardized mortality rates in 2020, both sexes and all ages, melanoma skin cancer [6]	18
Fig. 7. Sustainable Development Goals [42]	20
Fig. 8. Goal 3: Ensure healthy lives and promote well-being for all at all ages [42]	20
Fig. 9. Block diagram of the proposed melanoma detection method. [30]	22
Fig. 10. Architecture of the proposed CNN. [30].....	23
Fig. 11. Schematic overview of their CNN architecture. [31].....	23
Fig. 12. Confusion matrix from DenseNet201, ResNet152, Inception_V4 and DenseNet201 [32] .	24
Fig. 13. Example of digitalization of an image. At left, the image; at centre, the pixels labelled with numbers from 0-255, representing their brightness; and at right, these numbers by themselves. [10]	27
Fig. 14. RGB Image Representation [11]	28
Fig. 15. Representation of colour in RGB colour space. [12]	29
Fig. 16. Example of convolution operation. [14]	30
Fig. 17. Sets A and B. [15]	32
Fig. 18. Union between A and B. [15]	32
Fig. 19. Intersection between A and B. [15].....	32
Fig. 20. Complement of set A [15].....	33
Fig. 21. Example of dilation. [15]	34
Fig. 22. Example of erosion. [15].....	34
Fig. 23. Example of opening. [16].....	35
Fig. 24. Example of closing. [16]	35
Fig. 25. Diagram of the relation between Artificial Intelligence, Machine Learning and Deep Learning. [21]	36
Fig. 26. Example of the architecture of an artificial neural network. [24]	37
Fig. 27. Biological and artificial neuron design. [25].....	38
Fig. 28. Most common artificial neural networks. [26]	39
Fig. 29. A simple CNN architecture, extracted from the work of K. O’Shea and R. Nash [27]	40
Fig. 30. Common activation functions. [28].....	41
Fig. 31. Visualizing a CNN layers. [29].....	41
Fig. 32. Example of U-Net architecture where each blue box corresponds to a multi-channel feature map and white box represents copied feature maps. The arrows indicate the different operations. [20]	42
Fig. 33. Flow chart of the method proposed.....	46
Fig. 34. Benign Lesion. ISIC_0024597.....	47

Fig. 35. Malignant Lesion. AUGmented_0_602.....	47
Fig. 36. In the left image, the original benign lesion is shown while in the right part, the image after the application of the median filter.....	48
Fig. 37. In the left image, the original melanoma lesion is shown while in the right part, the image after the application of the median filter.....	49
Fig. 38. Comparison of the original benign lesion, in the left position, in the centre is the after the median filter, and in the right part is after the closing operation.	50
Fig. 39 Comparison of the original melanoma lesion, in the left position, in the centre it is after the median filter, and in the right part is after the closing operation.	50
Fig. 40 Comparison between all the pre-processing steps of the benign lesion, since the original image (the one in the left), until the image after the application of the CLAHE method.....	51
Fig. 41 Comparison between all the pre-processing steps of the melanoma lesion, since the original image (the one in the left), until the image after the application of the CLAHE method.....	52
Fig. 42 Histogram of the benign lesion. The blue line is referred to the original image and the blue line is referred to the final image obtained	52
Fig. 43 Histogram of the melanoma lesion. The blue line is referred to the original image and the blue line is referred to the final image obtained	53
Fig. 44 Output of the segmentation process in the benign lesion image	58
Fig. 45 Output of the segmentation process in the melanoma lesion image	58
Fig. 46. Demonstration of the pre-process in a benign lesion.	65
Fig. 47. Demonstration of the pre-process in a melanoma lesion.	66
Fig. 48. Division of the benign image lesion in right and left part	67
Fig. 49. Division of the melanoma image lesion in right and left part.....	68
Fig. 50. Division of the benign image lesion in up and down part.....	68
Fig. 51. Division of the melanoma image lesion in up and down part.....	68
Fig. 52. Comparison of the right and up images with flipped left and down parts, respectively, in a benign lesion.....	69
Fig. 53. Comparison of the right and up images with flipped left and down parts, respectively, in a melanoma lesion.	69
Fig. 54. Overlapping in the benign lesion.	71
Fig. 55. Overlapping in the melanoma lesion.....	72
Fig. 56. Asymmetry Form score of the benign lesion.	72
Fig. 57. Asymmetry Form score of the melanoma lesion.....	73
Fig. 58. Histogram of the benign lesion.	74
Fig. 57. Histogram of the melanoma lesion.	75
Fig. 60. Reference colours. “Melanoma Skin Cancer Detection based on Image Processing” by Zghal, N. S., & Derbel, N. [1].....	79
Fig. 61. Criteria of ABCD rule. “Melanoma Skin Cancer Detection based on Image Processing” by Zghal, N. S., & Derbel, N. [1].....	85

List of tables

Table 1. Estimated age-standardized incidences and mortality rates.....	19
Table 2. Estimated age-standardized incidences and mortality rates.....	48
Table 3. Closing Operation.	49
Table 4. CLAHE algorithm	50
Table 5. U-Net model	54
Table 6. Model training.....	56
Table 7. Adjust data	56
Table 8. Train Generator	57
Table 9. Binary Crossentropy.....	58
Table 10. Resize of the image and mask.....	59
Table 11. Negative of the Mask.....	59
Table 12. Remove the background.....	60
Table 13 . Crop Image.....	61
Table 14. Calculate the number of black pixels	62
Table 15. Obtain the coloured image	63
Table 16. Division of the image in 4 regions	67
Table 17. Inversion of the images	69
Table 18. Percentage of overlapping RIGH-LEFT.....	70
Table 19. Percentage of overlapping UP-DOWN.....	71
Table 20. Calculation of the overlapping percentage.....	72
Table 21. Calculation of the histogram of the right part of the image.....	73
Table 22. Calculation of the histogram of the left part of the image	73
Table 23. Calculation of the Chi-Square distance between histograms.....	75
Table 24. Final asymmetry value	76
Table 25. Find the biggest contour and lesion	76
Table 26. Crop the lesion area.....	77
Table 27. Division of the lesion in eight sectors	77
Table 28. Calculation of the border irregularity score	78
Table 29. White colour.....	80
Table 30. Black colour	80
Table 31. Red colour.....	80
Table 32. Light-Brown colour	81
Table 33. Dark-Brown colour	81
Table 34. Blue-Gray colour	81
Table 35. Ck value for white colour.....	82
Table 36. Ck value for black colour.....	82
Table 37. Ck value for red colour	83
Table 38. Ck value for light-brown colour.....	83
Table 39. Ck value for dark-brown colour.....	83
Table 40. Ck value for blue-gray colour	84
Table 41. Diameter Calculation	84
Table 42. Diameter score value calculation.	85
Table 43. Calculation of the Total Dermatoscopic Value (TDV) and classification	86
Table 44. Results of the benign lesions.....	88

Table 45. Results of the melanoma lesions.....	89
Table 46. Results of the images taken with a phone.....	91

List of abbreviations and terms

Abbreviations:

Assoc. prof. – associate professor;

Lect. – lecturer;

Prof. – professor.

Terms:

SEER – The Surveillance, Epidemiology, and End Results program.

ASR – Age Standardised Rate.

ECIS – European Cancer Information System.

UV – Ultraviolet Light.

SGDs – Sustainable Development Goals.

ABCD – Asymmetry, Border, Contour and Diameter.

CNN – Convolutional Neural Network.

AM – Acral Melanoma.

RCNN – Region Based Neural Network.

TDV – Total Dermatoscopic Value.

SVM – Support Vector Machine.

RGB – Red, Green and Blue.

CLAHE – Contrast Limited Adaptive Histogram Equalization.

AI – Artificial Intelligence.

ANN – Artificial Neural Network.

DCN – Deep Convolutional Network.

MNIST – Modified National Institute of Standards and Technology.

ReLU – Rectified Linear Unit.

BSD – Berkeley Software Distribution.

OpenCV – Open-Source Computer Vision.

API – Application Programming Interface.

GPU – Graphics Processing Unit.

1. Introduction

Nowadays, computer vision is booming, so it is used in every field that you can imagine, even in medicine. Computer vision involves computer graphics, machine learning, artificial intelligence, sensor technology and image and signal pre-processing.

Melanoma detection usually required an experiment dermatologist who, based on visual recognition, decided if the lesion may be suspicious or not. But it is not an easy task, and it requires a lot of experience.

This project aims to use these kinds of techniques to process medical imaging, improve diagnosis, and make more accessible doctors' work. This system could also help with an earlier first detection because it does not need any experience, so it could be an excellent tool to use in centres of primary attention to estimate the urgency of the lesion.

This project is based on the article "Melanoma Skin Cancer Detection based on Image Processing" by Zghal, N. S., & Derbel, N. [1].

1.1. Medical Motivation

Melanoma is not among the most common skin cancer, but malignant melanoma accounts for most of the growing mortality from skin cancer.

The incidence of melanoma has been increasing considerably in the last years worldwide. For example, in the United States, melanoma incidence has grown between 1975 and 2009: in men from 7.5/100.000 to 31.6/100.000 and women from 7.5/100.000 to 19.9/100.000, see Fig 1 [2].

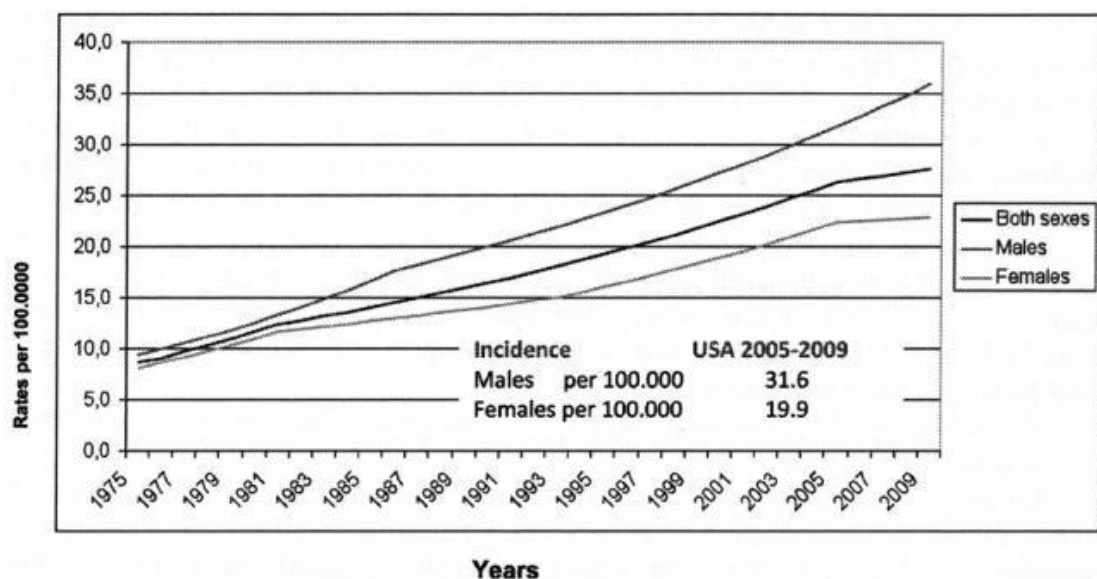


Fig. 1. Incidence of melanoma in the USA according to SEER Cancer Statistics Review 1975 - 2009

Looking the statistics about Lithuania and Murcia (The data from Spain is divided by regions) is observed that also the number of cases has been growing during the last years.

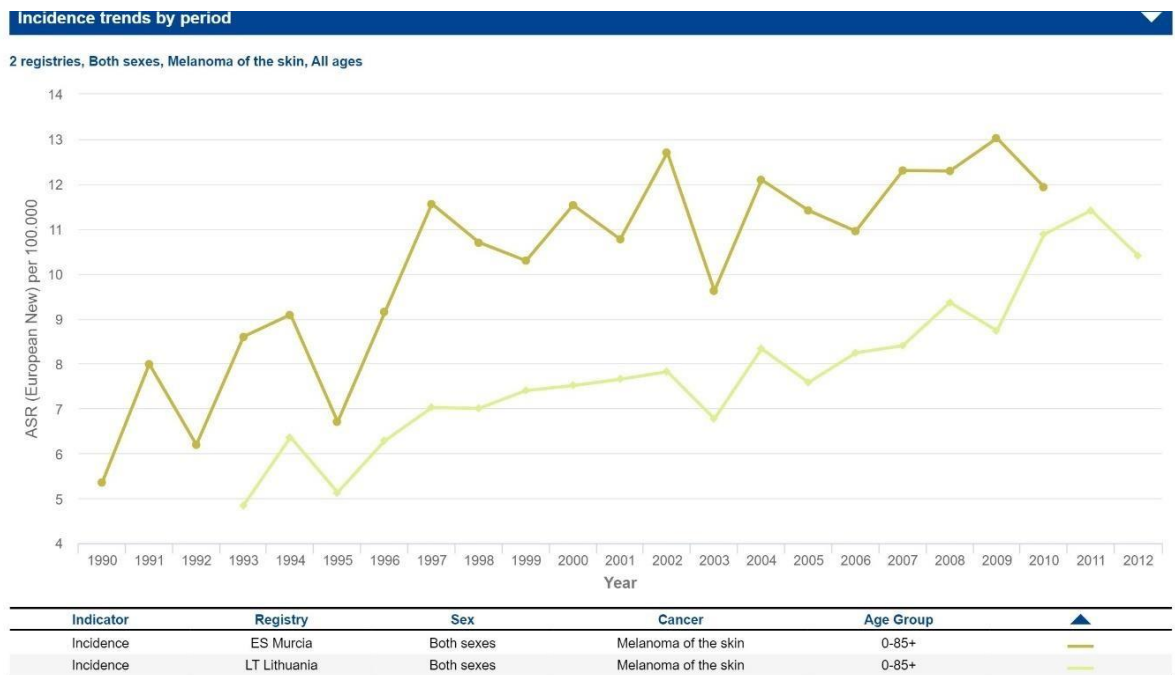


Fig. 2. Incidence trend by periods, comparing Lithuania and Murcia [4]

In this graph, the data is presented as a function of the period and the ASR parameter, which is, according to the European Cancer Information System (ECIS) [4]: “ASR (age-standardised rate) is a weighted mean of the age-specific rates where the weights are taken from the population distribution of a standard population; the ASR is expressed per 100,000. Comparing rates referring to different periods or different geographical areas is only possible after considering the differences in the age structure of the underlying populations. The age-standardisation allows the comparison of the rates that are arithmetically adjusted to have the same age structure as the standard population. The standard population used in the ECIS are the following old European Standard Population, new European Standard Population, and World Standard Population.”

Moving to some data more actualised, the International Agency for Research on Cancer (s.f.) [6], provides us with these graphs, which compares the incidence and the mortality of melanoma skin cancer with the rest of non-melanoma skin cancers.

Estimated age-standardized incidence rates (World) in 2020, non-melanoma skin cancer, both sexes, all ages

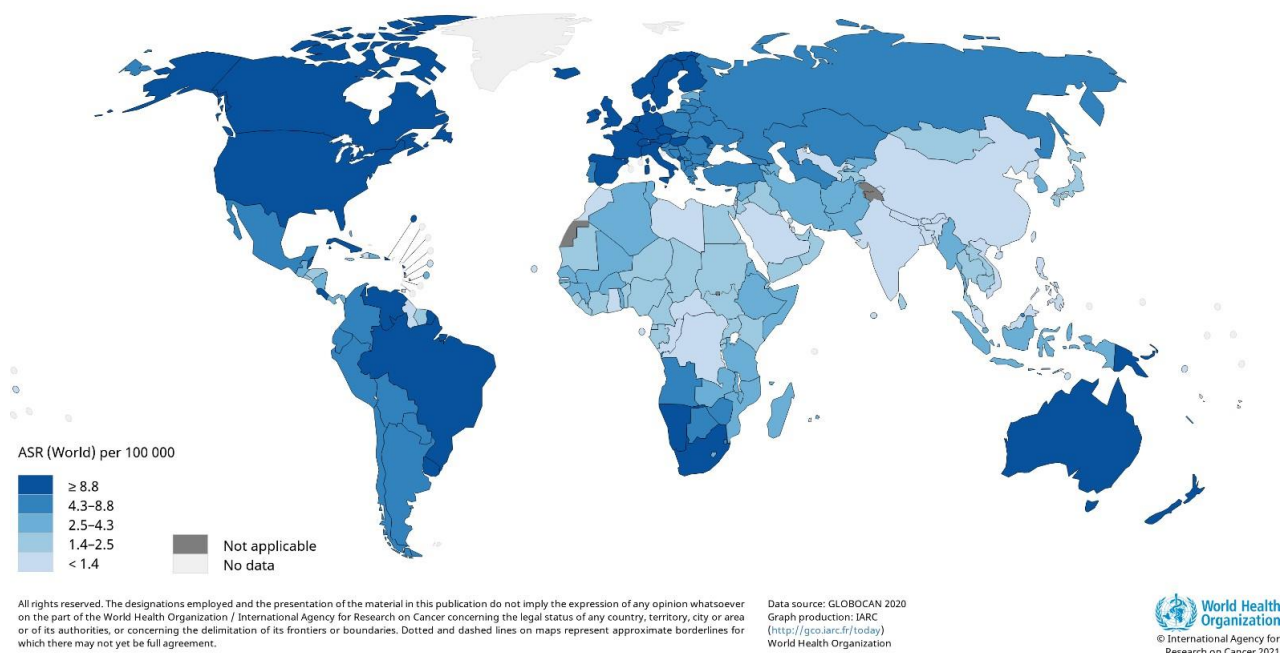


Fig. 3. Estimated age-standardized incidences rates in 2020, both sexes and all ages, non-melanoma skin cancer [6]

Estimated age-standardized incidence rates (World) in 2020, melanoma of skin, both sexes, all ages

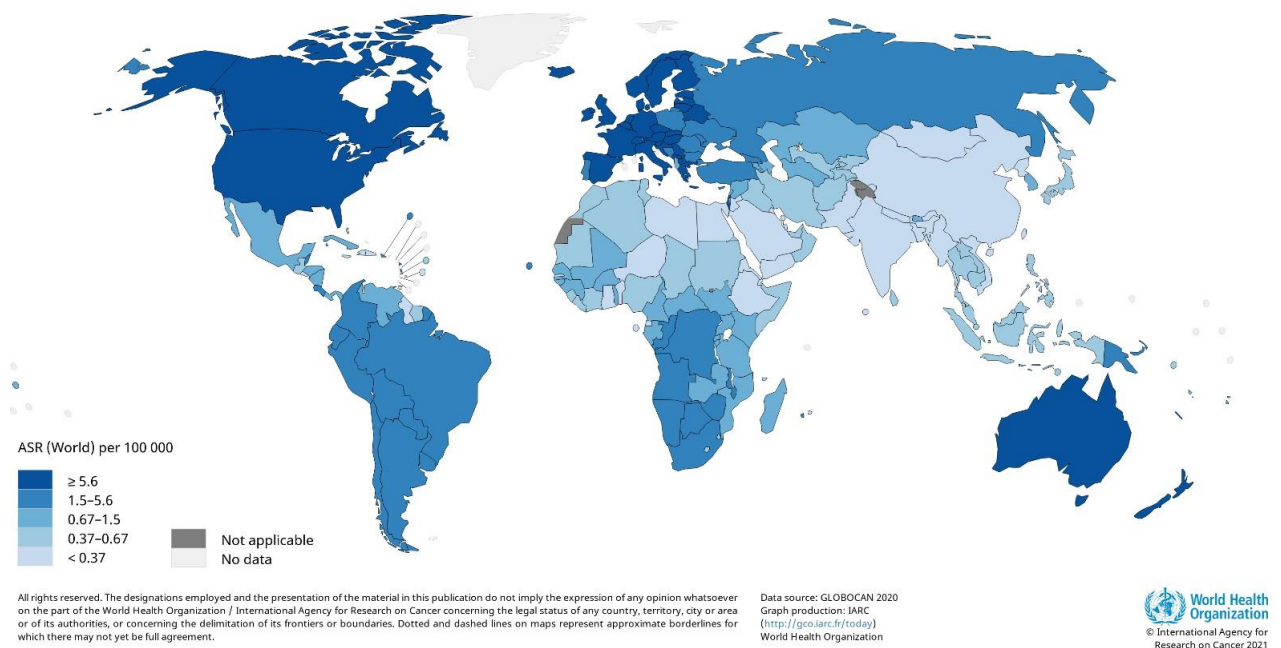


Fig. 4. Estimated age-standardized incidences rates in 2020, both sexes and all ages, melanoma skin cancer [6]

Estimated age-standardized mortality rates (World) in 2020, non-melanoma skin cancer, both sexes, all ages

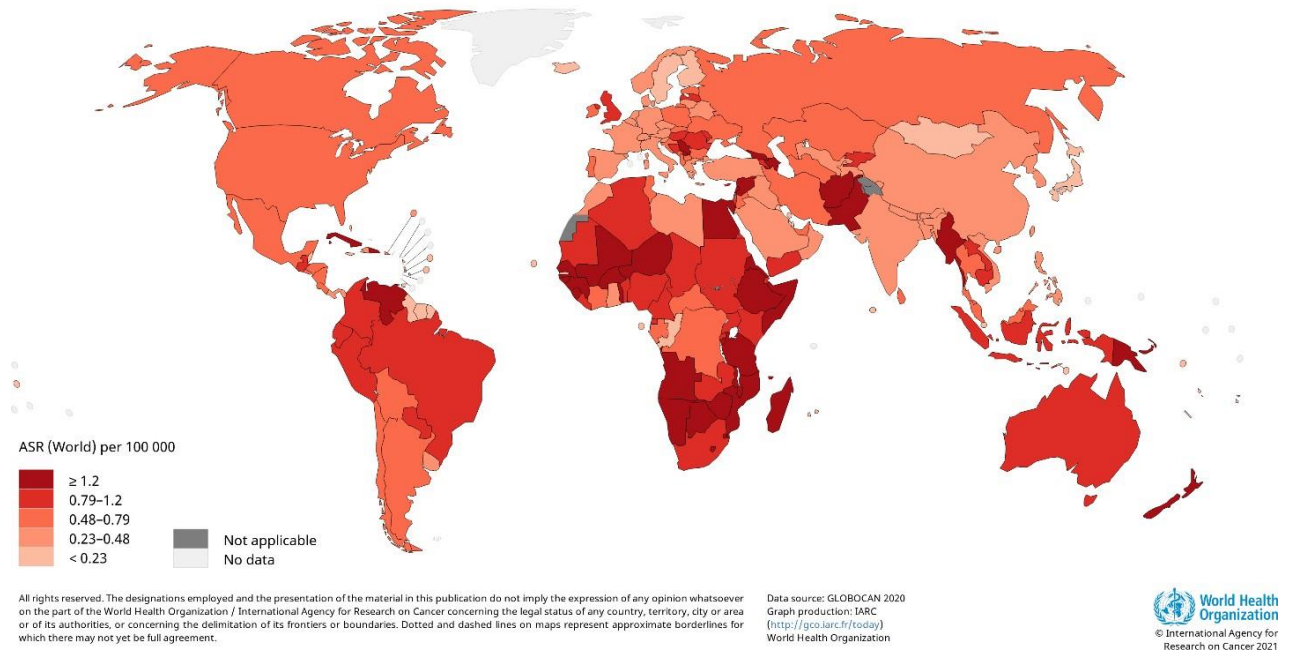


Fig. 5. Estimated age-standardized mortality rates in 2020, both sexes and all ages, non-melanoma skin cancer [6]

Estimated age-standardized mortality rates (World) in 2020, melanoma of skin, both sexes, all ages

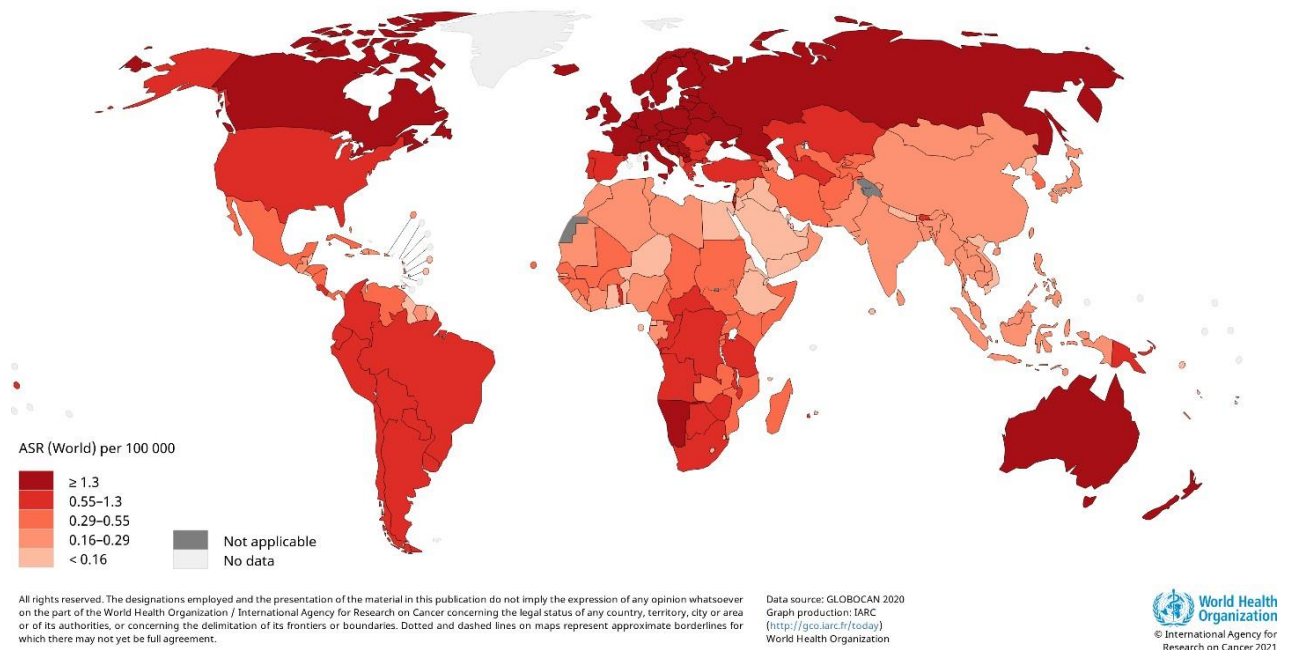


Fig. 6. Estimated age-standardized mortality rates in 2020, both sexes and all ages, melanoma skin cancer [6]

The following data is selected from the graphics above, and they are the estimated age-standardized incidences and mortality rates worldwide in 2020, taking into account both sexes and all ages. The data is presented as a function of the ASR per 100 000.

Table 1. Estimated age-standardized incidences and mortality rates.

Estimated Age-Standardized Rates		LITHUANIA	SPAIN
INCIDENCES	Non-Melanoma	6	12.7
	Melanoma	9.8	6.8
MORTALITY	Non-Melanoma	0.47	0.36
	Melanoma	1.8	0.98

As a general trend, it is noticeable that the mortality is much higher in melanoma skin cancer, remarkable in Lithuania, where the mortality rate of melanoma is four times bigger than the mortality of non-melanoma cancers. Therefore, even if, as an overall trend worldwide, non-melanoma skin cancer incidences are usually higher, melanoma cancer is more dangerous and should be taken more under research.

As in most cancers, prevention is the best way to reduce mortality and indices rates. According to the "Macmillan dictionary, "prevention is "The act or process of stopping something". In preventing melanoma, it is referred at the actions that should be taken to reduce as much as a possible risk, and it is divided into primary, secondary, and tertiary prevention. The primary entails reducing known risk factors in high-risk populations, i.e., reducing UV exposure (avoiding excessive sun exposure, covering skin with clothing, wearing hats, avoiding sunburn in childhood...). Secondary prevention is accomplished by diagnosis and treatment of early-stage (highly curable) melanoma. Finally, tertiary prevention involves limiting the morbidity and extending survival in patients with advanced disease. [7]

This work is about secondary prevention because it provides a fast and accurate system able to detect melanoma without the need for a specialist, reducing the time before the final diagnosis.

1.2. The 2030 Agenda for Sustainable Development

The 2030 Agenda for Sustainable Development sets the 17 Sustainable Development Goals (SGDs), which should be taken into account by all countries, developed and developing. These goals were approved by all United Nations Member States in 2015. The objective of this Agenda is to provide a shared plan for peace and prosperity. [42]



Fig. 7. Sustainable Development Goals [42]

Due to the importance of this Agenda, I highly believe that developed countries should be the first in the fight to meet these goals, being universities a vital place to raise awareness into the new generations about the strategies that build economic growth and address a range of social needs while tackling climate change and environmental protection.

In my thesis, I aspire to take action into Goal 3: "Ensure healthy lives and promote well-being for all at all ages." It is an indispensable requirement for prosperous societies, but inequalities in access to healthcare persist, and millions of people die each year due to late diagnosis and lack of care.



Fig. 8. Goal 3: Ensure healthy lives and promote well-being for all at all ages [42]

To be more specific, my project has been developed to cope with the target: "Strengthen the capacity of all countries, particularly developing countries, for early warning, risk reduction and management of national and global health risks". As was explained before, early detection is one of the best weapons to fight cancer. Nevertheless, early warning is not such a easy task in developing countries where the health system is not powerful enough to face it. To solve it, I would like in future investigation to create an app or a web page, which, using my algorithm, be able with just a photo, predict if the lesion skin is a benign lesion or a melanoma. With this tool, not only dermatologist at hospitals from developed countries will be able to use it, but also doctors without the specialization, and with less resources, will need just a phone with camera to have an accurate diagnosis.

1.3. Objectives

This project aims to implement a Computer-Aided Diagnosis system and its function is to help doctors with a “second opinion” about melanoma detection. For that reason, the system should be able to classify different melanocytic lesions in benign, suspicious, or malignant lesions.

To achieve this primary objective, the following partial objectives have been determined:

- Image pre-processing: using techniques of computer vision.
- Segmentation of the image: based on the U-Net system.
- Feature extraction: Following the algorithm ABCD, extracting data about the asymmetry, border irregularity, colours, and diameter.
- Classification: Each parameter is weighted depending on an algorithm.

2. State of Art

In this chapter, a brief review of how others have solved this problem is described. Many different techniques can be used; the most important will be explained: convolutional neural network and the ABCD rule.

2.1. Convolutional Neural Network

The CNN classifier is trained by a large number of training samples to, with a new image, which was not in the training set, determine which kind of lesion is.

2.1.1. “Melanoma Detection by Analysis of Clinical Images using Convolutional Neural Network” by E. Nasr-Esfahani [30]

In the research of Nasr-Esfahani et al., a method of melanoma detection is implemented based on the use of a CNN to classify lesions between melanoma and benign cases. The method used is shown in Fig9.

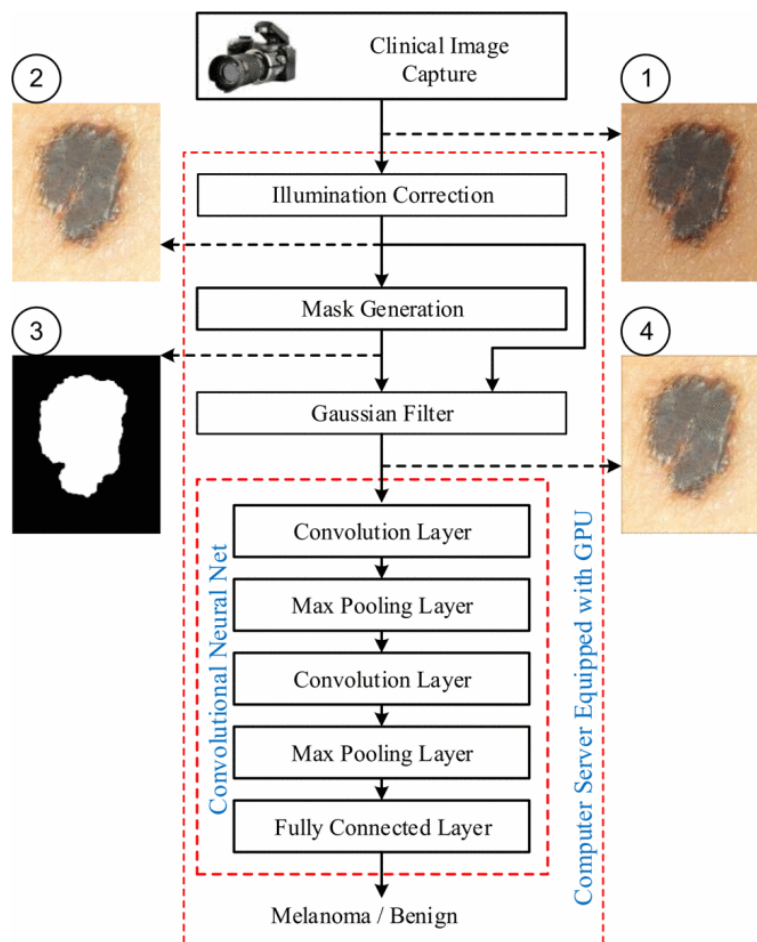


Fig. 9. Block diagram of the proposed melanoma detection method. [30]

As can be seen, it is based on two main steps; the first one, the pre-processing step, where the reduction of the effects of the noise is made. The second step is the application of the CNN architecture which consists of two convolving layers with 5 x 5 kernel, with twenty feature maps in the first convolution layer and 50 in the second. There is also one pooling layer after each convolution layer. The outputs are fed to a 2-layer, fully connected stage. The configuration proposed can be seen in Fig10.

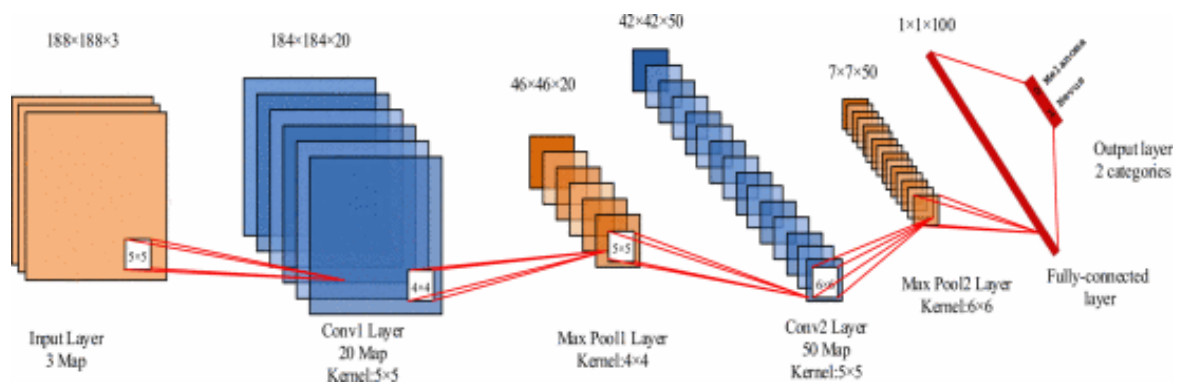


Fig. 10. Architecture of the proposed CNN. [30]

The result obtained are:

Sensitivity: 81%

Specificity: 80%

Accuracy: 81%

2.1.2. “Acral Melanoma Detection using a Convolutional Neural Network for Dermoscopy Images” by Yu C, Yang S, Kim W, Jung J, Chung KY, et al. [31]

In this study, the CNN framework is used to detect acral melanoma (AM), a particular type of melanoma that forms on the palms, feet, under fingers or toenails. It is the most common among people with darker skin. They fine-tuned a modified VGG model with 16 layers (13 convolutional and three fully connected layers), which use the convolutional kernels of 3 x 3. The structure of the architecture can be observed in Fig.11.

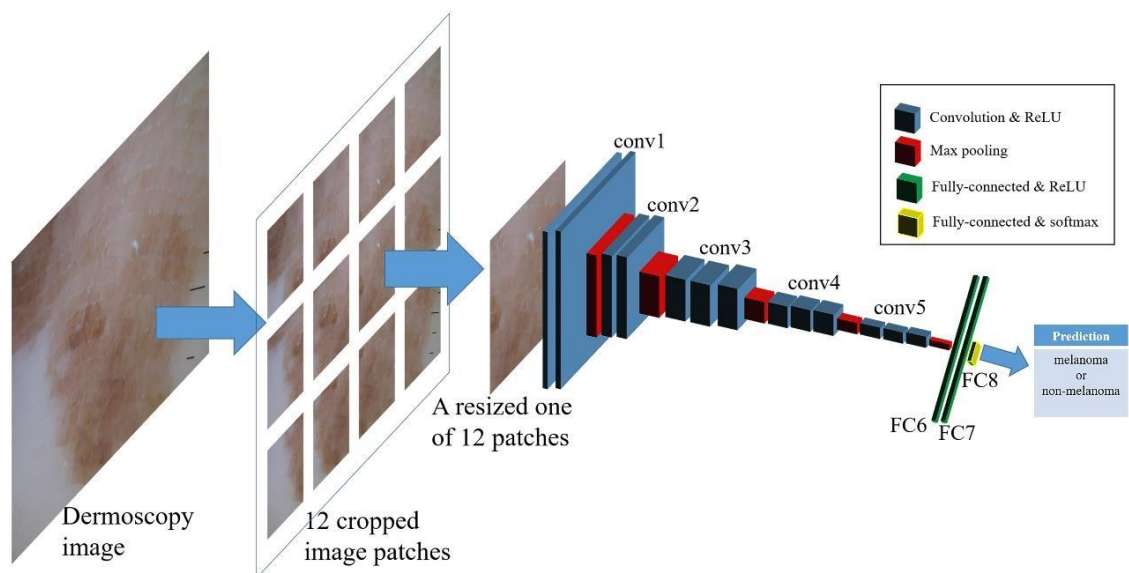


Fig. 11. Schematic overview of their CNN architecture. [31]

The result obtained are:
 Sensitivity: 92.57%
 Specificity: 75.39%
 Accuracy: 85.31 %

2.1.3. “Skin Lesion Analysis Towards Melanoma Detection via End-to-end Deep Learning of Convolutional Neural Networks” by Katherine M. Li and Evelyn C. Li, et al. [32]

In this study, the detection is not just binary (melanoma or benign lesion) due to they realized classification between 7 different classes (melanoma, melanocytic nevus, basal cell carcinoma, actinic keratosis, benign keratosis, and vascular lesion). The main difficulty is that the number of images of each category is highly imbalanced. The work is divided into two tasks:

- Task 1: Lesion Boundary Segmentation. The model MASK RCNN and a ResNet50 as backbone are used. In this task, they did not separate the seven categories in the training set; they treated all the lesion regions as one class and added a background class during training.
- Task 2: Lesion Detection: They did it with four variants of the CNN model. The result of each CNN model is shown by the confusion matrix obtained, Fig.12.

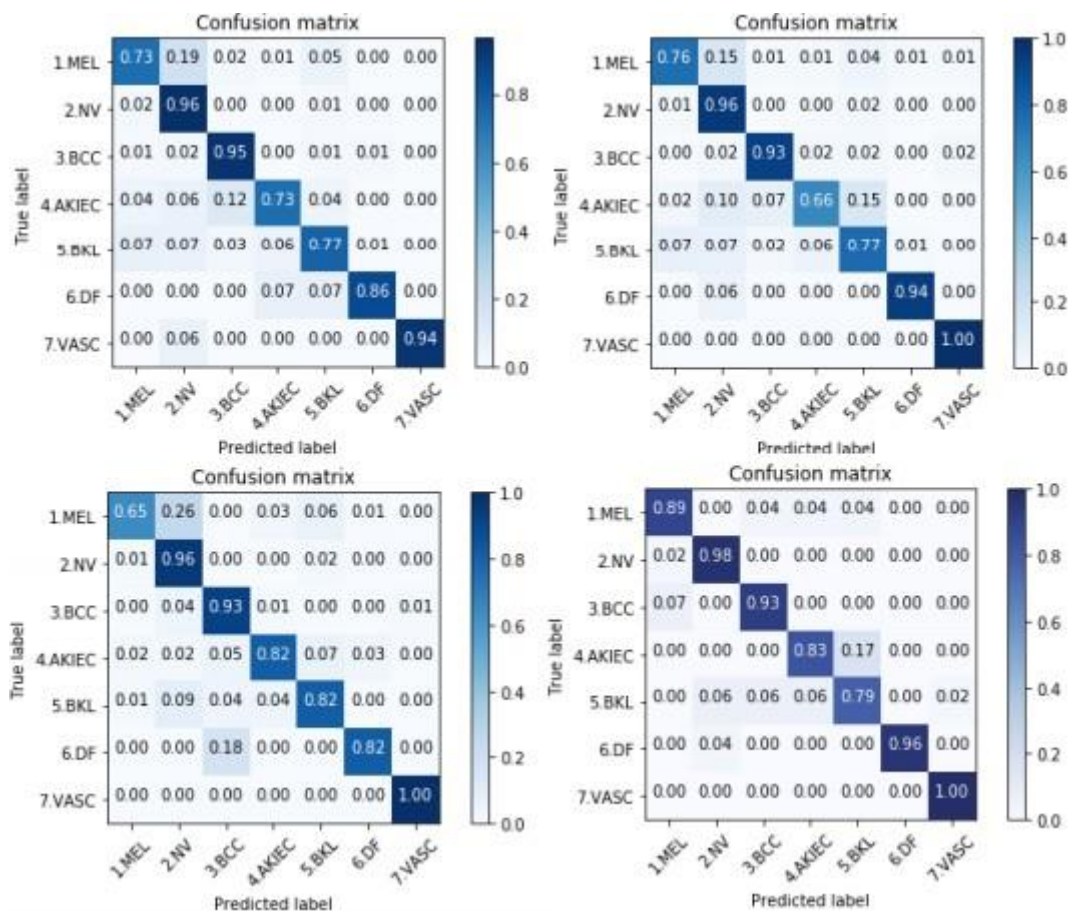


Fig. 12. Confusion matrix from DenseNet201, ResNet152, Inception_V4 and DenseNet201 [32]

The average values obtained are:

DenseNet201: 0.8485

ResNet152: 0.8600

Inception_V4: 0.8571

DenseNet201: 0.9114

So, it is noticeable that the one which obtained the best result is the DenseNet201.

2.2. ABCD Rule

The ABCD algorithm was developed almost 40 years ago, and since then, it is one of the essential techniques for the early detection of melanomas.

2.2.1. “Melanoma Skin Cancer Detection based on Image Processing” by Zghal, N. S., & Derbel, N. [1].

This article implements a Computer-Aid Diagnosis system which has three stages: lesion detection, feature extraction based on the ABCD rule and classification, based on the Total Dermatoscopic Value (TDV). It attends the following steps: First, the skin lesion mask is created, and it is applied to the image obtained in the pre-processing step to obtain the segmented image. This segmentation is performed on both moles and melanoma lesions. After that, the ABCD rule is applied, and with the result obtained, the TDV is calculated. The results are:

Sensitive: 87.5%

Specificity: 90.8%

Accuracy: 91.5%

2.2.2. “Melanoma Recognition Framework based on Expert Definition of ABCD for Dermoscopic Images” by Abbas, Q., Emre Celebi, M., Garcia, I. F., & Ahmad, W. [33].

Their method follows the subsequent process: The image collection is pre-processed to enhance contrast, then the hair pixels are detected and replaced by a fast-marching inpainting technique. A skin tumour area extraction algorithm is applied. Next, the expert features are extracted to define the clinical ABCD attributes and then construct the feature vector, which is normalized. After that, the optimal features are selected using the sequential floating forward selection technique. Finally, the classification decision is performed using a support vector machine (SVM) algorithm.

The result obtained are:

Sensitivity: 87.35%

Specificity: 89.75%

2.2.3. “Classification of Malignant Melanoma and Benign Skin Lesions: Implementation of Automatic ABCD Rule” by Reda Kasmi and Karim Mokrani. [34].

This study used image processing techniques for an automatic ABCD rule implementation to discriminate benign lesion from malignant melanomas. A pre-processing step is done first to remove

artefacts like bubbles and thin hair, as in the anterior methods. Thick hairs are detected using Galbor filters with different frequencies and orientations, and the image is repaired using linear interpolation based on the hair mask. Then, lesions are segmented. The next step is the estimation of the values A (Asymmetry), B (Border Irregularity), C (Colour) and D (Diameter). Finally, the TDV value is computed. The obtained results are:

Sensitivity: 91.25%

Specificity: 95.83%

Accuracy: 94%

3. General Theoretical Concepts

In this part of the paper, some of the most important theoretical concepts will be explained, as they are a must to understand the used techniques.

3.1. Computer Vision

Computer vision is a vast field that includes numerous different techniques focusing on the extraction of information about a scene by analysing images of that scene. [8]

It is used in a wide range of different application; for instance, it is used in the automotive industry, agriculture, healthcare, finance, advertising...

As a colossal field, it includes too many different techniques. However, in this document, the main points treated are the digitalisation of the images, the colour spaces, operations on the histogram and filtering the image, morphological processing, and contrast enhancement.

3.1.1. Digitalization of the Images

An image is a bidimensional representation of a three-dimensional scene; it results from the acquisition of a signal provided by a sensor, which converts the collected information into numerical encoding, which can be read by a computer [9].

A digital image is a matrix of pixels. Each pixel has information about the brightness level, about each position of the image, and its value ranges from 0 to 255, being 0 black and 255 white. One example is the Fig.13, where the process of digitalization of an image is shown.

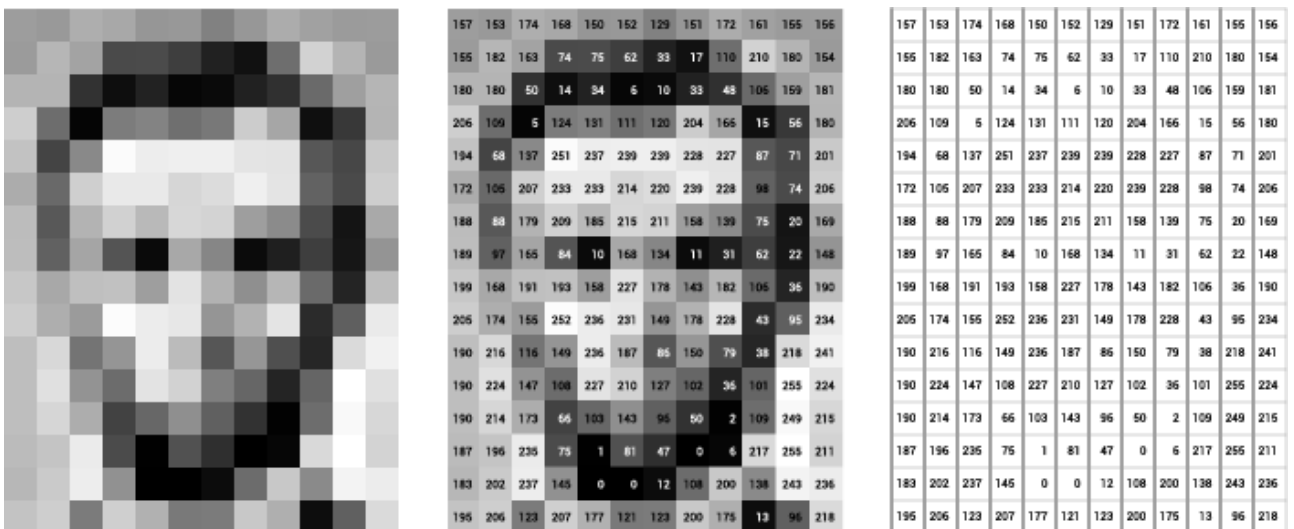
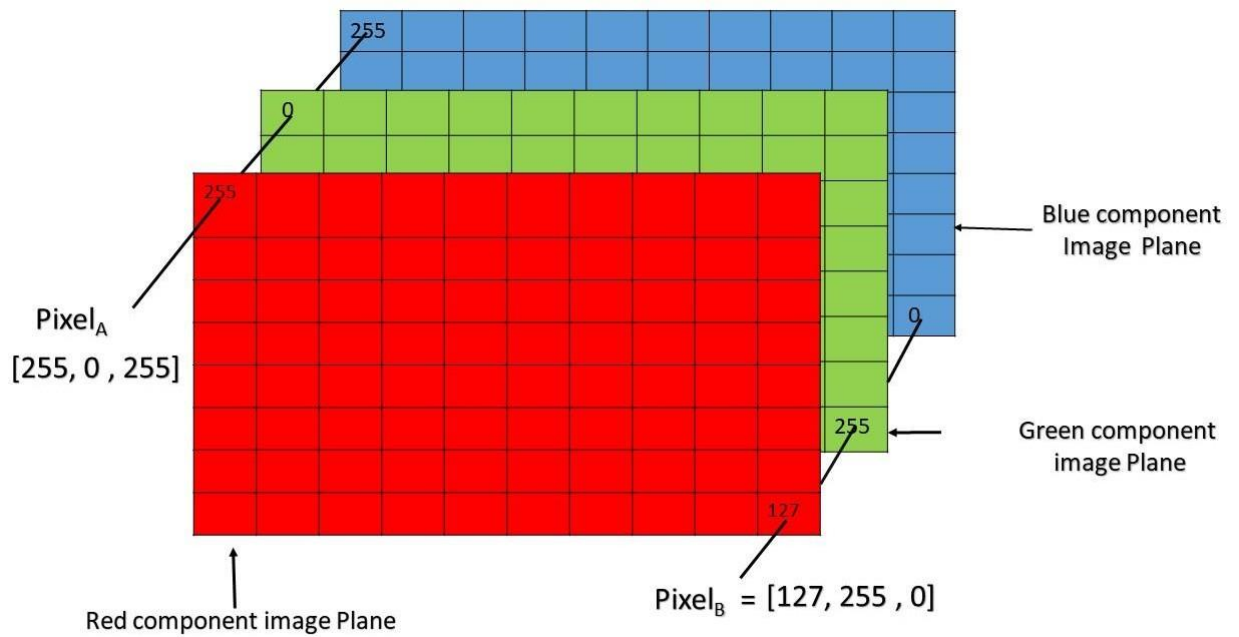


Fig. 13. Example of digitalization of an image. At left, the image; at centre, the pixels labelled with numbers from 0-255, representing their brightness; and at right, these numbers by themselves. [10]

The size of an image is expressed as $M \times N$, where M is the width while N is the height of the image.

A coloured image is done associating three matrices, the ones corresponding to the colour red, green, and blue (RGB images), so the size of these images is $M \times N \times 3$. The composition of the colours is as it is shown in the Fig.14.



Pixel of an RGB image are formed from the corresponding pixel of the three component images

Fig. 14. RGB Image Representation [11]

3.1.2. Colour Spaces

The example shown before is just one of the ways to represent colours in computer vision. Therefore, there are several ways to represent colours, depending on the colour space selected. Due to the number of different colour spaces available, only the ones used in this project will be referred to.

- RGB: It is the most common colour space because cameras, scanner, printer... are most often provided with RGB signal input or output, so no colour transformations are needed. The colour spectrum forms a cube as can be seen in Fig.15. Each colour described by its RGB components is represented by a point found in the cube. All grey colours are on the main diagonal of the cube. The main disadvantages of this space colour are: the correlation between its components make this space unsuitable for compression; also, it is psychological non-intuitive, which mean that it is not easy to visualize a specific colour in the cube, and finally non-uniformity. [12]

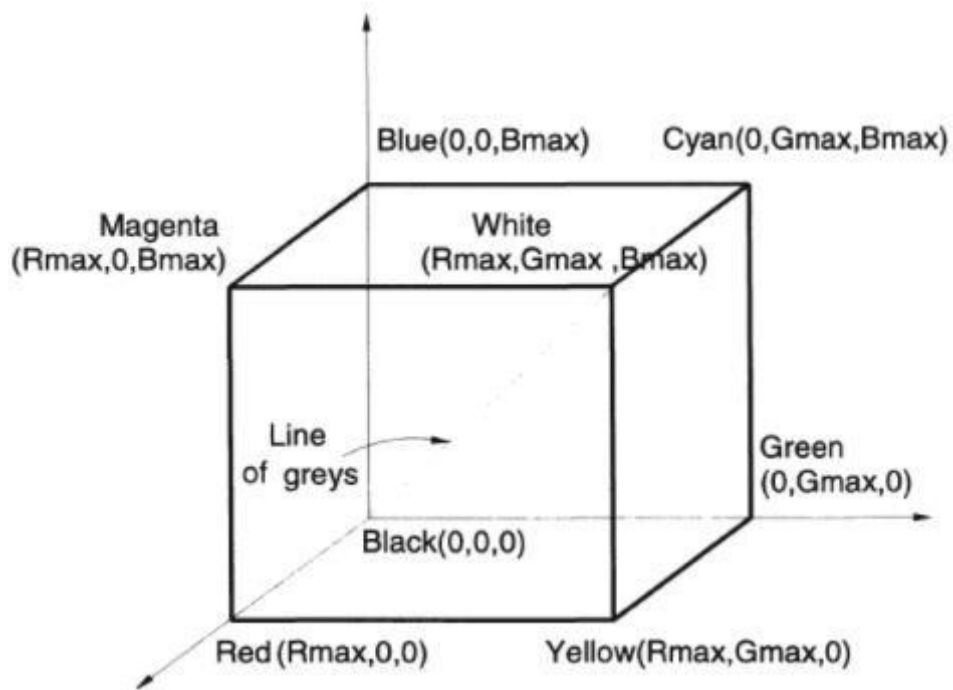


Fig. 15. Representation of colour in RGB colour space. [12]

- CIEL*a*b space: it is one of the uniform colour spaces. It is used because it represents all the colours mathematically, and it is defined by the expressions: [12]

Formula [1]. Definition of CIEL*a*b colour space

$$L^* = 116 f\left(\frac{Y}{Y_0}\right) - 16$$

$$a^* = 500 \left[f\left(\frac{X}{X_0}\right) - f\left(\frac{Y}{Y_0}\right) \right]$$

$$b^* = 200 \left[f\left(\frac{Y}{Y_0}\right) - f\left(\frac{Z}{Z_0}\right) \right]$$

where

$$f(x) \begin{cases} x^{1/3}, & x > 0.008856 \\ 7.787x + \frac{16}{116} & \text{otherwise,} \end{cases}$$

L^* is the lightness value.

a^* denotes relative redness-greenness.

b^* denotes relative yellow-blueness.

Due to the L^* value is the referee to the lightness; this colour space is typically used to modify the luminosity of an image more quickly. For that reason, this space colour was used.

3.1.3. Operation on the Histogram and Filtering the Image

The main aim of these operations is to enhance the characteristics of images by reducing the noise of the image. Most of them are based on the modification of the histogram of the images. According to the research of Bankman (2000) [13], “An image histogram is a grey-scale value distribution showing the frequency of occurrence of each grey-level value”. So, the histogram of image i is defined by the histogram $h(i)$:

Formula [2]. Histogram of an image i .

$$h(i) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \delta(f(m,n) - i), \quad i = 0, 1, \dots, P - 1$$

$$\delta(w) = \begin{cases} 1 & w = 0, \\ 0 & \text{otherwise.} \end{cases}$$

One of the most helpful enhancement operations is the convolution using kernels, also called filters, which are local operators. The convolution operation is an algebraic operation of two functions that produces a third function. It is represented by the operator “*”. Graphically, it overlaps one matrix over another, multiplying each cell by the inferior and adding the products. [16]

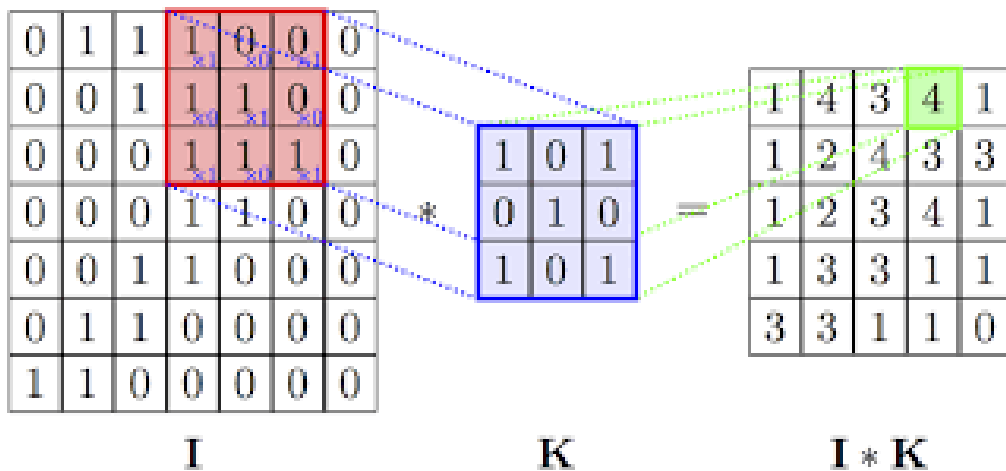


Fig. 16. Example of convolution operation. [14]

As shown in Fig.16., the matrix of the image (I) is multiplied, pixel by pixel, by the pixels of the kernel (K), and the result is the sum of values obtained in the multiplication of each pixel.

There are two different types of filters, linear and non-linear filters.

- Linear Filters:

- Mean Filter: It consists of replacing each pixel of the matrix for the average value of its pixels with its neighbours.

For example, for a filter the kernel (K) of size 3 x 3, the mean filter will be:

$$1/9 \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

It is used to reduce the impulse noise because it is easy to implement, and it does not require a high computational cost. On the other hand, it is sensitive to local changes, and it can create new grey values which were not present in the original image.

- Gaussian Filter: It is used to blur images and eliminate noise. It is like the mean filter but using another kernel, which is obtained by the modelling of the gaussian function:

Formula [3]. Gaussian Filter
$G(x,y) = \frac{1}{2\pi\sigma^2} * e^{-\frac{x^2+y^2}{2\sigma^2}}$

This filter is used because it produces a more uniform smoothing than the mean filter and because the losses in the sharpness are minor.

- Non-linear filters:

- Median Filter: It orders the values in the neighbourhood of each pixel, from smaller to largest and selects the value at the intermediate position. One example could be:

15	25	36	89	88
28	48	85	41	85
53	86	21	52	85
58	88	12	28	94
15	15	25	23	36

The ordered values are
{12, 21, 28, 41, 48, 52, 85, 86, 88}

The median value would be: 48

15	25	36	89	88
28	48	85	41	85
53	86	48	52	85
58	88	12	28	94
15	15	25	23	36

So, after it, the new matrix is:

This filter is so standard because it reduces the impulsive noise, preserves the borders, and it does not create new grey values. As a disadvantage, it needs a higher computational cost.

3.1.4. Operation on the Histogram and Filtering the Image

Morphological operations are image processing techniques that deal with the shape of features in an image. These operations are used to remove imperfections or noise introduced in the images during segmentation.

Taking the article called “Morphological Image Processing” by M. Goyal [15], it is essential to explain the basic mathematical operations that will be needed later. It is too important the concept of set, which represent objects in an image. As M. Goyal did, these sets are taken:

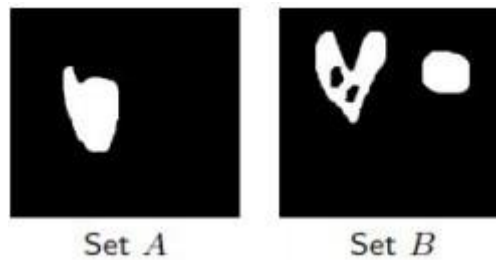


Fig. 17. Sets A and B. [15]

The basic operations between sets are:

- Union: It produces a new set that includes the elements which are in sets, A and B. It is represented by the expression $C = A \cup B$:

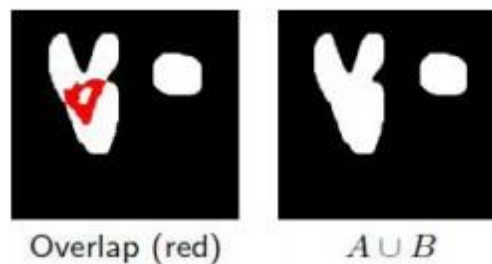


Fig. 18. Union between A and B. [15]

- Intersection: It includes the common element of both sets: $C = A \cap B$:

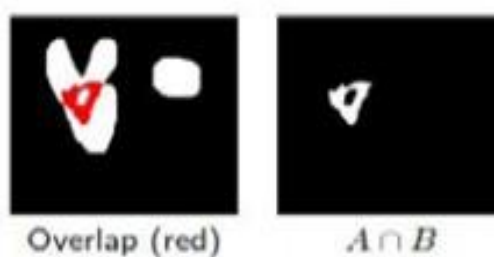


Fig. 19. Intersection between A and B. [15]

- Complement: The complement of A is the set of elements which are not contained in A: A^c

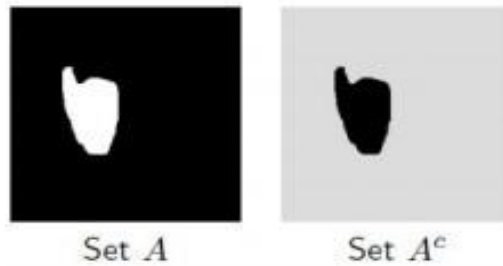


Fig. 20. Complement of set A [15].

- Reflection: It is defined as:

Formula [4]. Definition of Reflection.

$$\hat{B} = \{ w \mid w = -b, \text{ for } b \in B \}$$

The basic morphological operations are dilatation and erosion. Both are produced by the interaction of a set called the “structuring element” with the pixel set of the image.

The principal structuring elements used are crosses, rectangles, and ellipses.

- Dilatation: It is the process of structuring the element B on image A, moving it across the image doing the convolution operation. A is the input image while B is the structure element, or kernel, which determines the exact effect of the dilatation.

Formula [5]. Operation of dilation. [17]

$$A \oplus B = \text{Dil}(A, B) = \{ z \mid (B)_z \cap A \neq \phi \}$$

Where ϕ is for the empty set, B is the kernel and \hat{B} is for the reflection of set B.

Dilatation is used for repairing intrusions or breaks.



Fig. 21. Example of dilation. [15]

- Erosion: It is the same operation as in the dilation, but, in this case, pixels are converted to “white” not to “black”, so the object is reduced. The same inputs as in the dilation are used, A is the image input, and B is the structural element or kernel.

Formula [6]. Operation of Erode. [17]

$$A \ominus B = \text{Ero}(A, B) = \{z \mid (B)_z \cap A^c \neq \phi\}$$

Where ϕ is for the empty set, B is the kernel and B is the kernel and A^c is the complement of the collection A.



Fig. 22. Example of erosion. [15]

- Opening: It consists of an erosion followed by a dilation, both with the same structuring element. It is used to eliminate the noise of white pixels in the black part of the image.

Formula[7] . Operation of Open. [17]

$$\text{Opening: } A = (A \ominus B) \oplus B$$



Fig. 23. Example of opening. [16]

As can be seen in the Fig 23, the noise of white pixels is disappeared after the use of the opening operation.

- Closing: It consists of a dilation followed by an erosion, both with the same structuring element. It is used to eliminate the noise of black pixels in the white part of the image.

Formula [8]. Operation of Closing. [17]
$\text{Closing: } A = (A \oplus B) \ominus B$



Fig. 24. Example of closing. [16]

In Fig.24., it is easy to see how the black pixels in the white region of the lesion disappear after using this function.

3.1.5. Contrast Enhancement

There are several ways to enhance contrast in image processing. However, one of the most important in medical application is the method “Contrast-limited adaptive histogram equalization”, more commonly known as CLAHE, which is used for assigning displayed intensity levels in medical images. It is designed to easily allow the observer to see all contrast of an image quickly. This method is based on histograms modified because the contrast enhancement induced by the method at each intensity level is limited to a maximum. CLAHE method examines that histogram of intensities in a contextual region centred at each pixel. It sets the displayed intensity at the pixel as the rank of that pixel’s intensity in its histogram.

3.2. U-Net Architecture

Artificial intelligence (AI) is the field which is trying to imitate the human way of thinking. The main difference between artificial intelligence and other methods is that it is unnecessary to program it for each different possible situation. The principal objective of AI is to teach the program to solve this situation by itself. There are many options to do it, but the main fields of investigation are:

- Machine Learning: It is based on computer training by the developers, so it can recognise a pattern and make predictions according to the data provided.
- Deep Learning: It is not necessary that no one teach the program how to obtain the conclusions because it can learn by itself.

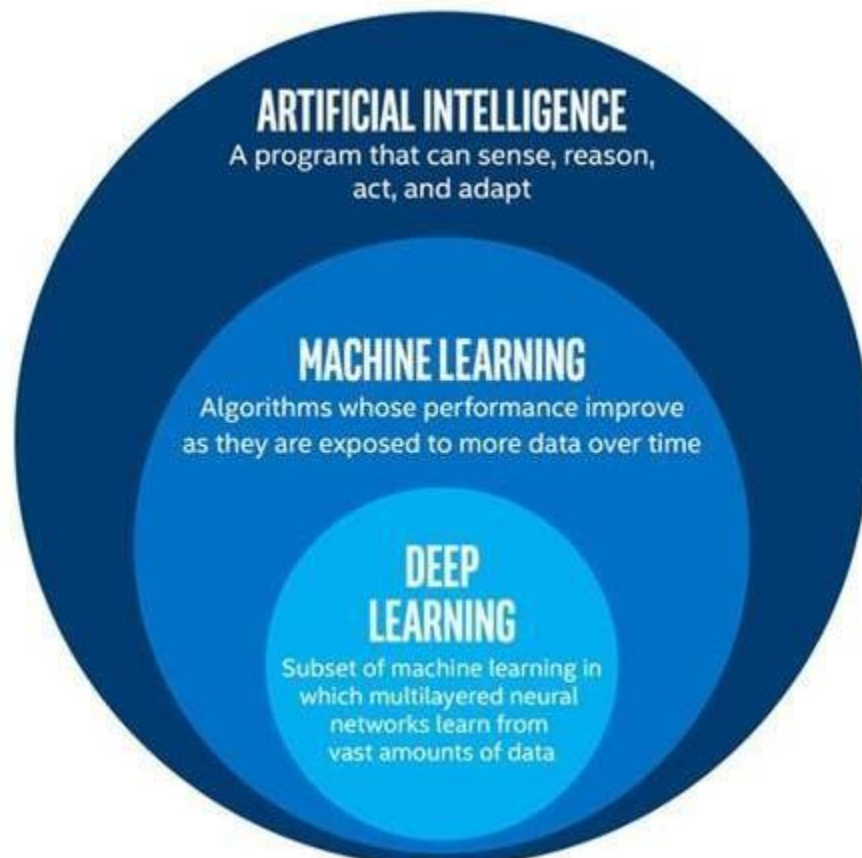


Fig. 25. Diagram of the relation between Artificial Intelligence, Machine Learning and Deep Learning. [21]

In the diagram above, Fig.25., it can be seen how artificial intelligence is a vast field that has multiple sub-fields as machine learning and deep learning. It means that all of them are entirely related and cannot possibly understand one of them without the others.

Inside the field of “Deep Learning” is the U-Net architecture, but what U-Net is?

The U-Net architecture is an essential convolutional neural network developed to use in the segmentation of medical images. It was developed by Olaf Ronneberger, Philipp Fischer, Thomas Brox at the University of Freiburg [20].

It is a type of convolutional neural networks, which in turn are a type of neural network. So, firstly, it is important to understand what a neural network is.

An artificial neural network (ANN) is based on the sophisticated functionality of human brains. The human brain works thanks to billions of interconnected neurons which process information in parallel. As is explained in the book writing by Gallant (1993, p. 3) [23], it consists of a set of computational cells and a set of one-way data connection units. A unit examines its inputs at some particular times, and then it computes its output as a number, call activation. This activation is passed to the next cell, and in this way, it is going through them. Each connection has a signed number, known as a “weight”, that determines whether the activation which is travelling along it influence the cell to produce a similar or a different activation. The size of the weight determines the magnitude of their influence.

The architecture of an artificial neural network consists of an input layer of neurons (or cells), some hidden layers and a final layer of output neurons. [24]

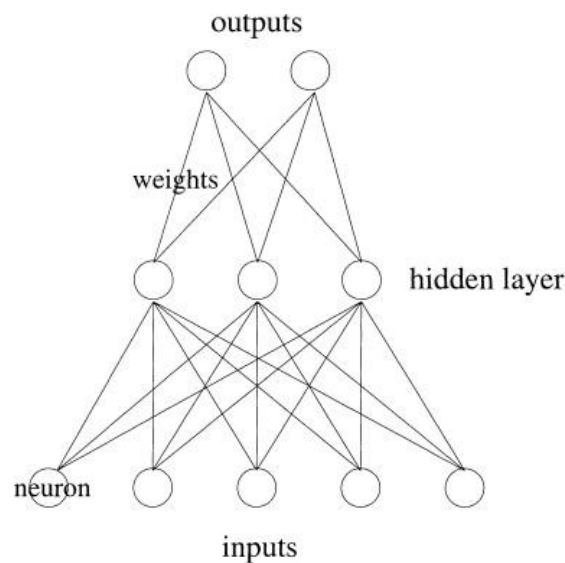


Fig. 26. Example of the architecture of an artificial neural network. [24]

As we can see in Fig 26, the structure of an ANN is composed of three phases, inputs, hidden layer, and outputs. Each phase is also composed of neurons which are the basic unit of the system and are all interconnected.

The neurons or cells are derived from the observation of a biological neuron that is the basic unit of the biological neural network, which includes the brain, spinal cord, and peripheral ganglia. The similarities can be observed in Fig.27 [25].

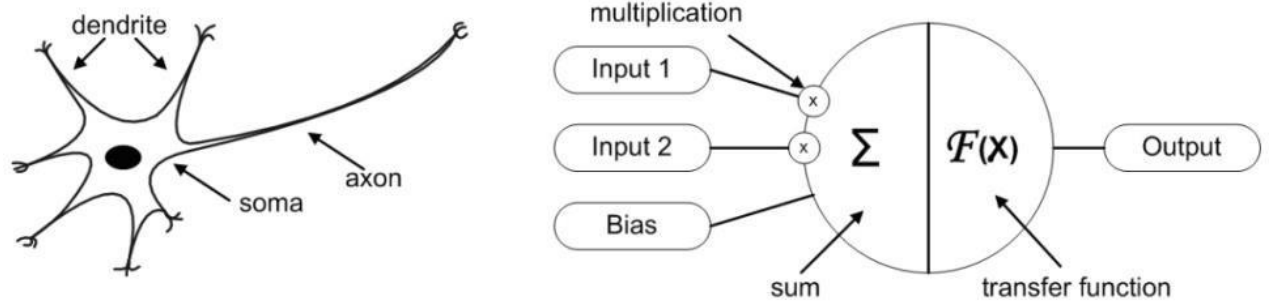


Fig. 27. Biological and artificial neuron design. [25]

By the comparison between biological and artificial neuron, it is easy to see how the artificial one is based on the biological one, due to the dendrites are the inputs in artificial neuron, the soma corresponds with the body of the neuron, which in the artificial one it is the sums of the weights, bias, and the processing of the sum with a transfer function. Finally, this information is sent to the followed neuron by the inputs, while in biological neuron, this process is done by the axon.

There are a lot of different types of ANN, depending on how these neurons are connected. The following chart shows a recompilation done by Andrew Tch [26]

A mostly complete chart of
Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

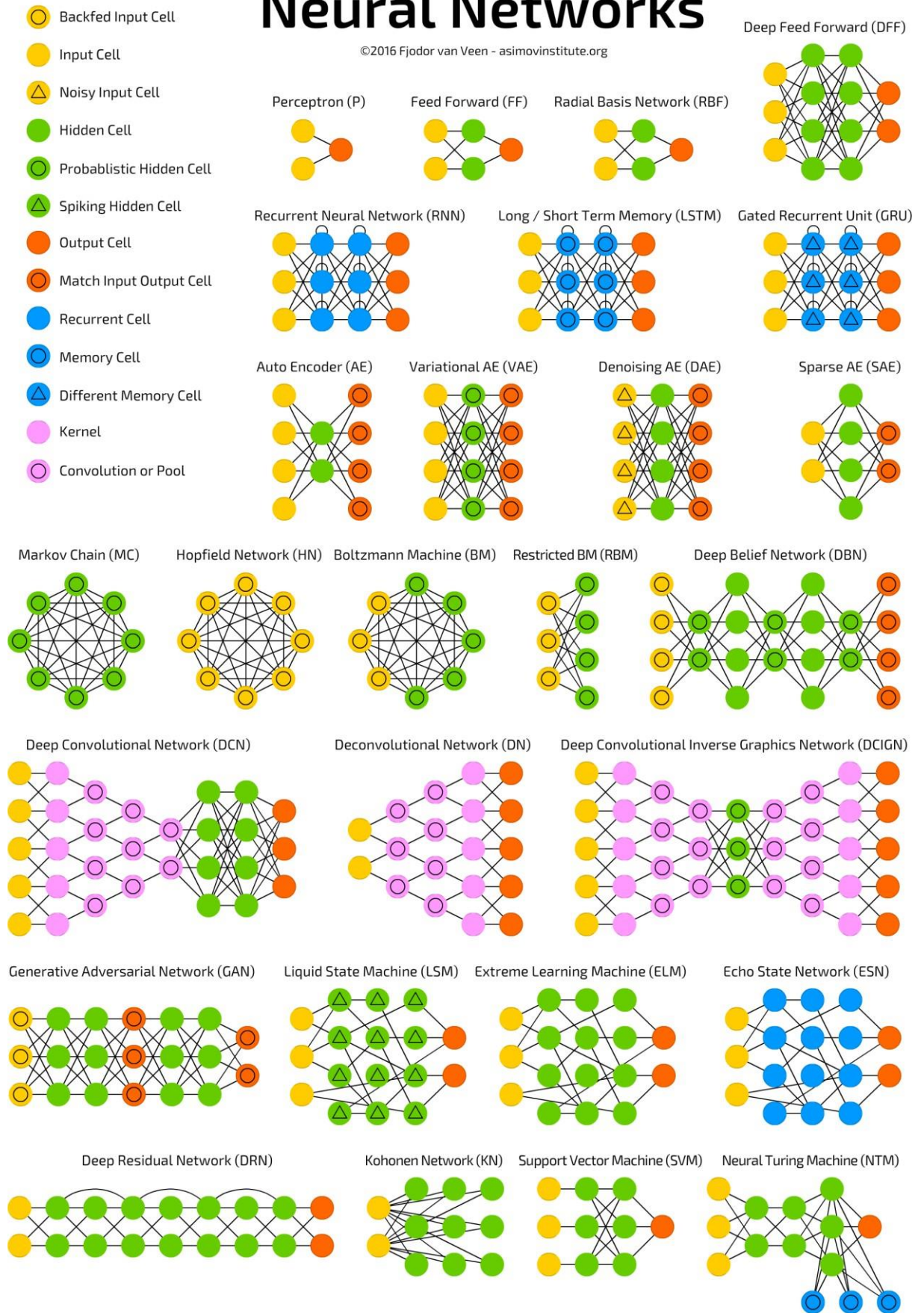


Fig. 28. Most common artificial neural networks. [26]

Due to the large number of different types, only the one in which the U-net architecture is used will be explained, the convolutional neuronal network (CNN) or, as it is named in Fig.28, the deep convolutional network (DCN).

Convolutional Neural Networks have been growing fast during the last decades. Its more significant achievement is that it needs fewer parameters than Artificial Neural Networks. An important aspect to consider is that problems solved by CNN should not have features spatially dependent. The main difference between them is that CNN is primarily focused on the basis that the input will be images, so its architecture is specially prepared for it. CNNs are comprised of four types of layers, the input layer, the convolutional layer, the max-pooling layer, and the fully connected layers. A simplified example of CNN architecture for MNIST is shown in Fig.29.

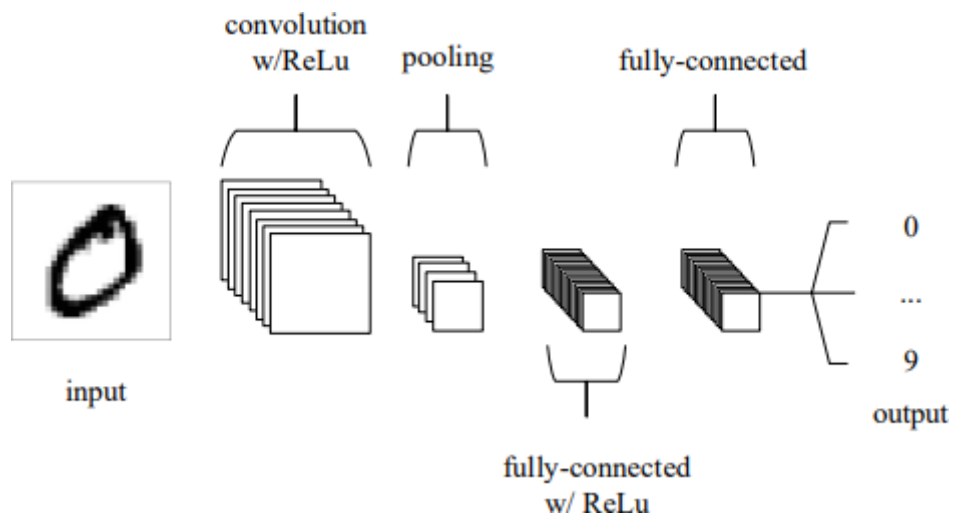


Fig. 29. A simple CNN architecture, extracted from the work of K. O’Shea and R. Nash [27]

As said before, the four layers are [27]:

- **Input Layer:** It holds the pixels values of the image.
- **Convolutional Layer:** It determines the output of neurons by calculating the product between their weights and the region connected to the input volume. This operation is performed by filters that are applied to the images. The **rectified linear unit (ReLU)** aims to apply an “elementwise” operation. First, it sets all negative pixels to zero; it introduces a non-linearity to the network. Finally, the output is rectified.

Formula [9]. ReLU activation function.

$$f(x) = \max(0, x)$$

Even the ReLU is one of the most common activation functions, it is important to remark that it is not the only one; as can be seen in Fig.30, there are several different options.

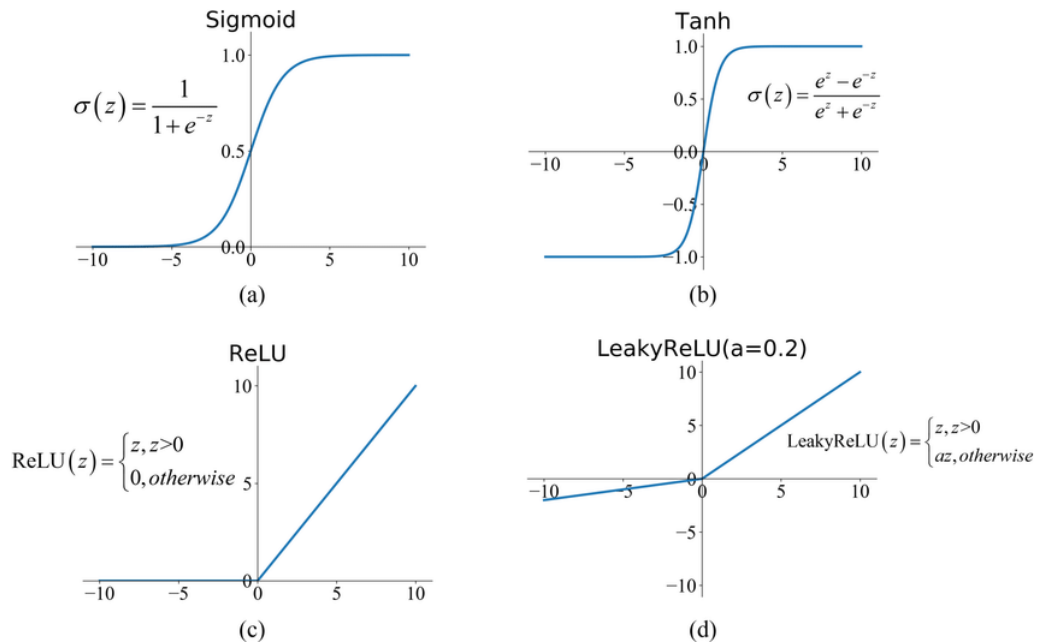


Fig. 30. Common activation functions. [28]

- The pooling layer: It performs downsampling along with the spatial dimensionality of the given input, further reducing the number of parameters within the activation.

- The fully connected layers: It will produce class scores from the activations, to be used for classification.

CNN's can transform the original input layer by layer using its simple method of transformation to produce class scores for classification and regression purposes. [27]

An example is shown in Fig.31.

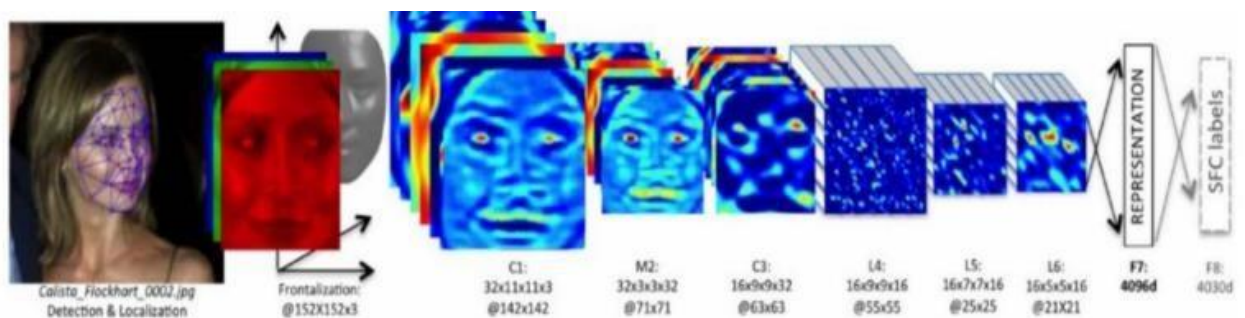


Fig. 31. Visualizing a CNN layers. [29]

In this image, face recognition is done. It is possible to see the transformation of the image after the different operations done by the CNN until obtained the label value.

Now that the principles of the ANNs and CNNs have been explained, it is easier to understand what the U-Net architecture is. As was mentioned before, the U-Net architecture is based on the CNNs, so its layers are similar. Nevertheless, the typical use of CNN is on classification task, where the input is an image, and the output is a single class level. However, the U-Net architecture aims that the

localization is an important feature to consider, so the output should assign a class level to each pixel, no just one class level to the whole image. [20]

The U-Net takes its name because of its architecture, as shown in Fig.32; it is similar in form to a “U” letter.

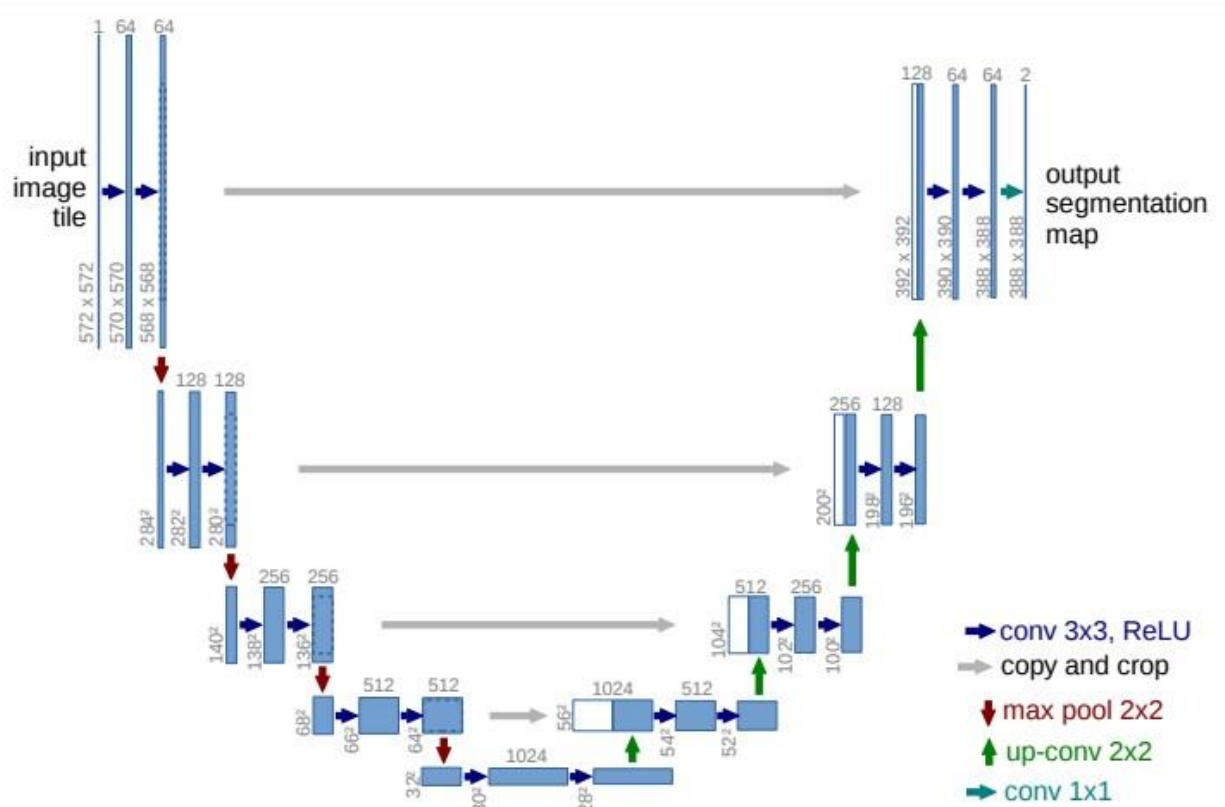


Fig. 32. Example of U-Net architecture where each blue box corresponds to a multi-channel feature map and white box represents copied feature maps. The arrows indicate the different operations. [20]

The U-Net architecture is divided into two regions; the left part of the figure is related to the contracting path, which follows the typical architecture of a convolutional network. It consists of the repeated application of convolutions, each followed by an activation function (ReLU) and max pooling operation for reducing the sampling. At each downsampling step, the number of feature channels is double. The right part of the architecture is the expansive path. Each of its steps consists of an upsampling of the feature map followed by a convolution which divided by two the number of feature maps. The concatenation with the correspondingly cropped feature map from the contracting path is done. Finally, the convolution operation is done again, followed by the activation function (ReLU). [20]

4. Programming Language and Environment

The method proposed was developed by using Python as a programming language and “Google Colaboratory” as environment. In this chapter, a small introduction of both of them is done.

4.1. Python

As it is defined in its web page [35]: “Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.”

Its syntax is accessible and readable, which make it easy to learn. It also admits libraries and modules, which increased the modularity of the program and its reuse. Thanks to the wide number of libraries available, Python is one of the most used program languages nowadays in computer vision, machine learning, and artificial intelligence.

The libraries used are:

- NumPy: It is one of the packages more used for scientific computing in Python. It provides a multidimensional array object, various derived object (such as arrays or matrices) and an assortment of routines for fast operation in arrays. It is under the licence of BSD so that it can be used without restrictions. [36]
- OpenCV (Open-Source Computer Vision): it is an open-source software library used in computer vision and machine learning. It was created to provide a common infrastructure for computer vision applications. It has more than 2500 optimized algorithm, and this number is growing every day. [37]
- Matplotlib: It is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is used to plot images and graphs. [38]
- Math: It is a module that provides access to the mathematical functions defined by the C standard. One example used in the program is `math.sqrt` to calculate the square root. [39]
- TensorFlow: It is an end-to-end open-source platform for machine learning, and it is used in the developing and training of neural networks, which can detect and decode patterns. [40]
- Keras: As it is defined in its webpage [41]: “Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, minimizes the number of user actions required for common use cases, and provides clear & actionable error messages. It also has extensive documentation and developer guides.” It is used to develop complex models of deep learning.

4.2. Google Colaboratory

Google Colaboratory is a product of "Google Research", and it is usually called "Colab". It allows everyone to write and run a Python program in the navigator. It is unnecessary to download or install any program on the computer, so it could be a good solution for everybody to program in Python.

It is a free Jupyter notebook environment working entirely in the cloud. To use it only is needed a Google account to access. It provides the users with pre-installed libraries, free GPU and TPU use and the possibility to save programs in the cloud.

Colab primarily focuses on personal machine learning programs due to Google Research lets the user utilise their dedicated GPU's and TPU's, which, talking about acceleration, make a huge difference even for a small project.

5. Proposed Method

In this chapter, the method used for the detection of melanoma will be explained. Firstly, a brief introduction is done to give an overview about it, and then, the program will be explained step by step, giving all the details about the programming and the performance.

As shown in Fig.33, the first step is image acquisition; in our case, the image will be taken from an online dataset. These images are selected because they were taken with a dermatoscope, which assure us the best quality of the images.

The images are introduced in the pre-processing step, which improves the quality of the image to make it more manageable for the program image detection and feature extraction by filtering the images, removing the hairs and the contrast enhancement.

When the images are pre-processed, the next step consists of image segmentation; the method used to do it involves using a specific convolutional neural network that follows the U-Net architecture designed for medical image segmentation. As will be explained later, the model used was provided by Professor Vidas Raudonis, and it was trained by me.

Once the images are segmented, and the lesions are detected correctly, the following step is the feature extraction. Due to the classification will be done according to the ABCD algorithm, the characteristics which have to be analysed are: Asymmetry, Border irregularity, Colour and Diameter.

It is one of the most challenging steps due to the final results is highly conditioned by the accuracy with which these parameters are calculated.

Finally, the score calculated about each parameter will be introduced in an algorithm that will determine if one lesion is benign, suspicious or if it is a melanoma lesion.



Fig. 33. Flow chart of the method proposed.

To explain all the process, two cases are going to be shown, one of a benign lesion and the other of a melanoma lesion. The two images selected are shown in Fig.34 and Fig.35.



Fig. 34. Benign Lesion. ISIC_0024597

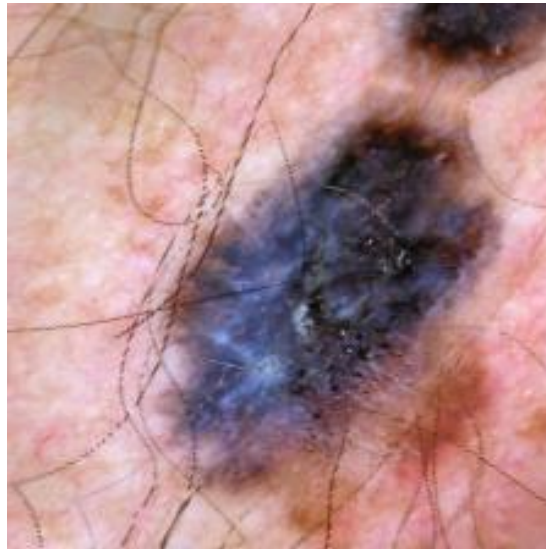


Fig. 35. Malignant Lesion. AUGmented_0_602.

5.1. Image Acquisition

Image acquisition is the selection of the images which will be used to test the model. In our case, the images were taken from the official dataset of the SIIM-ISIC Melanoma Classification Challenge, which was generated by the International Skin Imaging Collaboration (ISIC), and the images are from different hospitals and universities as: Hospital Clínic de Barcelona, Medical University of Vienna, Memorial Sloan Kettering Cancer Center, Melanoma Institute Australia, University of Queensland, and the University of Athens Medical School. [21]

5.2. Image Pre-processing

The aim of this step is treating the images to give a better idea of the malignancy of the lesions, to achieve it, it has been reduced the noise, removed the hair and other imperfections and a contrast enhancement was done. The following steps have been done:

- **Median Filter:** It is used to reduce the noise of the images. In table 2, the code for the median filter is shown.

Table 2. Estimated age-standardized incidences and mortality rates.

1. `ksize = 9`
2. `median_filter = cv2.medianBlur(imagen, ksize)`

The function used is from open cv library [16]: `cv2.medianBlur(src, ksize[])`

- `src`: the input, an image with 1, 3 or 4 channels.
- `ksize`: is the size of the mask, it must be odd and greater than 1. The larger this parameter is the greater is the blurring of the image.

As is visible in the Fig.36 the scale of the left part and the hairs are disappeared after the application of the median filter. The same effect is observed in the Fig.37 but due to the hairs are darker and bigger, the effect obtained is worse.



Fig. 36. In the left image, the original benign lesion is shown while in the right part, the image after the application of the median filter.

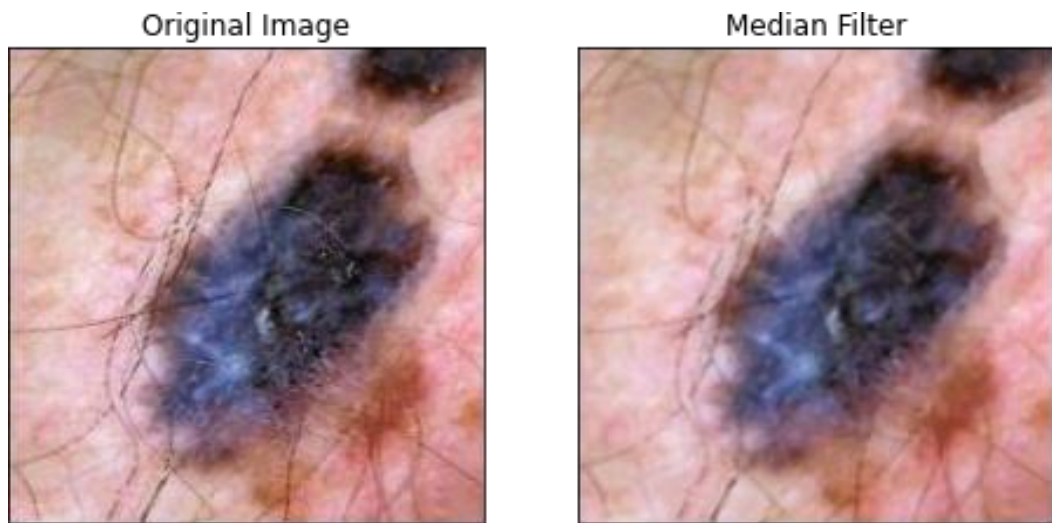


Fig. 37. In the left image, the original melanoma lesion is shown while in the right part, the image after the application of the median filter.

- **Morphological Operations:** The dilatation and erosion are done to complete a closing operation. It is used to eliminate all the hears and other artefacts that are in the images.

Table 3. Closing Operation.

1. kernel = cv2.getStructuringElement(shape=cv2.MORPH_CROSS, ksize = (5,5))
2. dilatation = cv2.dilate(median_filter,kernel,iterations = 1)
3. erosion = cv2.erode(dilatation,kernel,iterations = 1)

The functions used are:

cv2.getStructuringElement(shape, ksize,[,anchor]) [16]: It creates a structuring element of the specified shape and size for morphological operations.

- shape: One of the MorphShapes(which could be MORPH_RECT, MORPH_CROSS or MORPH_ELLIPSE).
- ksize: Size of the structuring element.
- anchor: Anchor position within the element.

cv2.dilate (scr, kernel[,dst[, anchor[, iterations[, borderType[, borderValue]]]]) [16]: It is used in the line 2 and it dilates the image using a specifies structuring element that determines the shape of a pixel neighbourhood.

- src: input image.
- dst: output image of the same size and type as scr.
- kernel: the structuring element used.
- anchor: position of the anchor within the element.
- iterations: number of times that the dilatation is applied.
- borderType: pixel extrapolation method.
- borderValue: in case of a constant border.

cv2.erode (scr, kernel[,dst[, anchor[, iterations[, borderType[, borderValue]]]]) [16]: This function is in the third line of the code and it erodes the image in the same way that the operation of dilation. The same parameters are needed.

Now, the result of the application of the morphological operations are shown in both images, the benign lesion, and the melanoma lesion.

As can be seen in the Fig.38., some of the hairs are removed with the median filter, while just few of them need the closing operation to disappear.

On the other hand, in the Fig.39., the median filter has almost no effect on the hairs, but, by contrast, the closing operation is advantageous and finally most of the hairs are removed.



Fig. 38. Comparison of the original benign lesion, in the left position, in the centre is the after the median filter, and in the right part is after the closing operation.

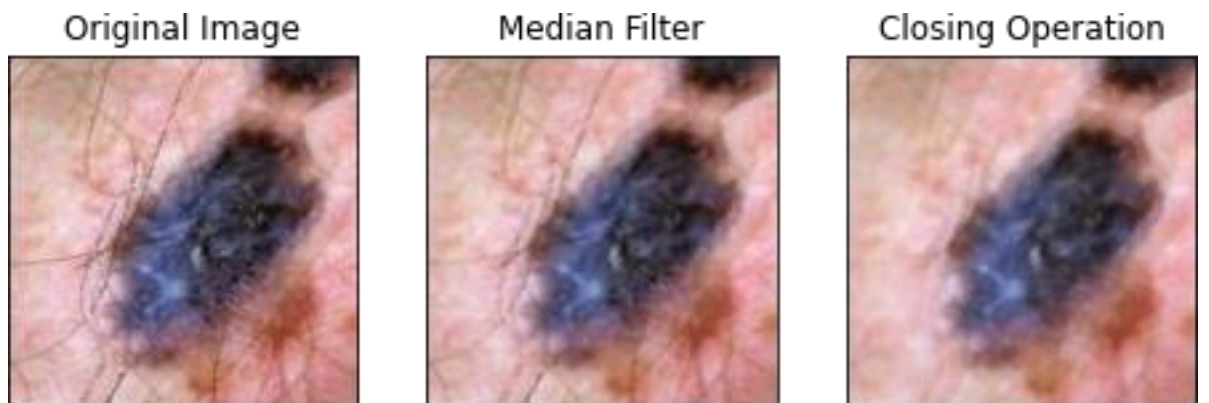


Fig. 39 Comparison of the original melanoma lesion, in the left position, in the centre it is after the median filter, and in the right part is after the closing operation.

- **Contrast limited adaptative histogram equalization (CLAHE):** It is used to improve the contrast of the image.

Table 4. CLAHE algorithm

1. `clahe = cv2.createCLAHE(clipLimit = 1, tileGridSize = (2,2))`
2. `lab = cv2.cvtColor(erosion1, cv2.COLOR_BGR2LAB)`
3. `l, a, b = cv2.split(lab)`
4. `l2 = clahe.apply(l)`
5. `lab = cv2.merge((l2, a, b))`
6. `clahe = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)`

The functions used are:

cv2.createCLAHE([, clipLimit [, tileGridSize]])[16]: In the line 1, it create a pointer to a cv::CLAHE class and initializes it.

- clipLimit: Threshold for contrast limiting.
- tileGridSize: Size of the grid for histogram equalization. Input image will be divided into equally sized rectangular tiles, and this parameter defines the number of tiles in row and column.

cv2.cvtColor(src, code [, dst [, dstCn]])[16]: It is in the second line and in the line 6, it converts an image from one colour space to another.

- src: input image.
- dst: output image of the same size and depth than src.
- Code: colour space conversion code, the most used are: cv2.COLOR_BGR2RGB, cv2.COLOR_RGB2BGR, cv2.COLOR_BGR2LAB, cv2.COLOR_LAB2BGR.
- dstCn: number of channels in the destination image.

cv2.split(m [, mv]): It is in the line 3 and it divides a multi-channel array, in this case the image lab, into several single-channel arrays. It is done because the CLAHE method is applied only to the L component of the image.

- m: input multi-channel image.
- mv: output vector of arrays.

cv2.clahe.apply(src,[, dst]) [16]: Located in the line 4, it equalizes the histogram of a grayscale image using Contrast Limited Adaptive Histogram Equalization.

- src: source image.
- dst: destination image.

cv2.merge(mv[, dst]) [16]: It is used at the end, in the line 5, to unite the different colour channels, when the l parameter has been modified, by the creation of one multi-channel array out of several single-channel ones.

- mv: input vector of matrices to be merged, all of them must have the same shape.
- dst: output array of the same size and depth than mv[0].

In the following images (Fig.40. and Fig.41), the pre-processing process can be seen, as at the beginning, in the original images there are several hairs and other artefacts which have been removed at the end. The last step (the CLAHE method) provides the image with a better contrast necessary to the segmentation process.



Fig. 40 Comparison between all the pre-processing steps of the benign lesion, since the original image (the one in the left), until the image after the application of the CLAHE method.

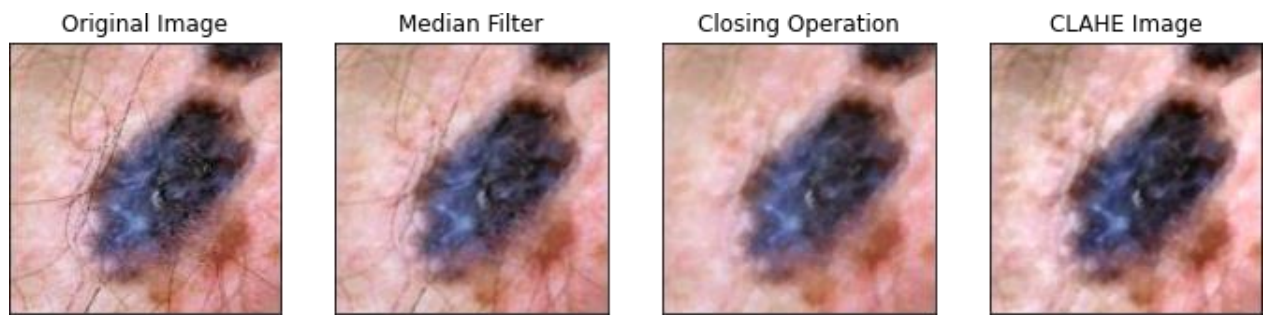


Fig. 41 Comparison between all the pre-processing steps of the melanoma lesion, since the original image (the one in the left), until the image after the application of the CLAHE method.

The Fig.42 and Fig.43. are the comparison between the histograms of the images before and after the pre-processing operation. The new image after the pre-processing step is much better because as we can see in the figures, the red line indicates a higher number of white pixels.

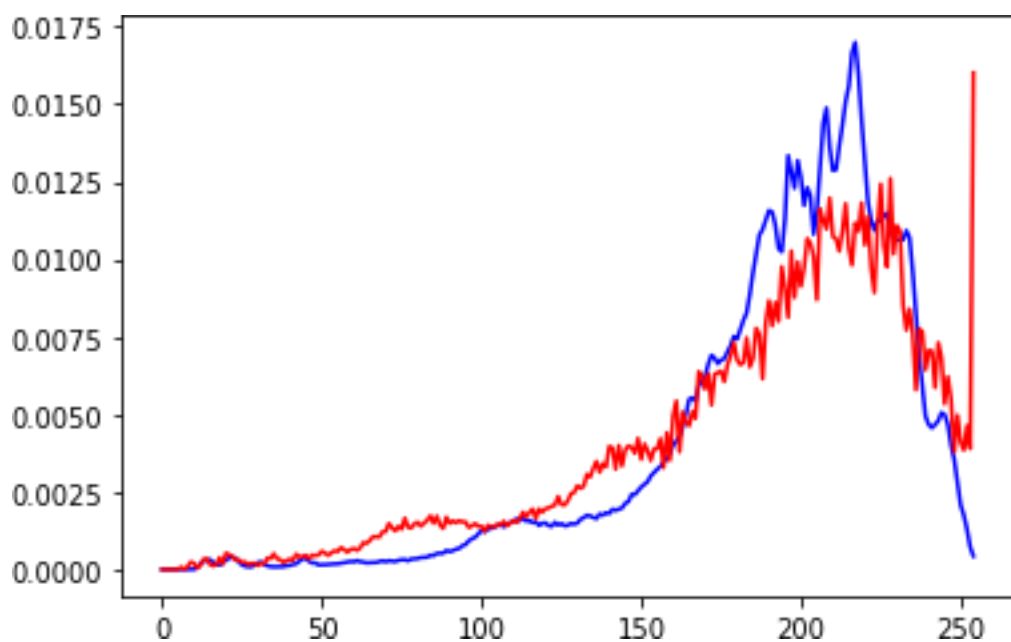


Fig. 42 Histogram of the benign lesion. The blue line is referred to the original image and the red line is referred to the final image obtained.

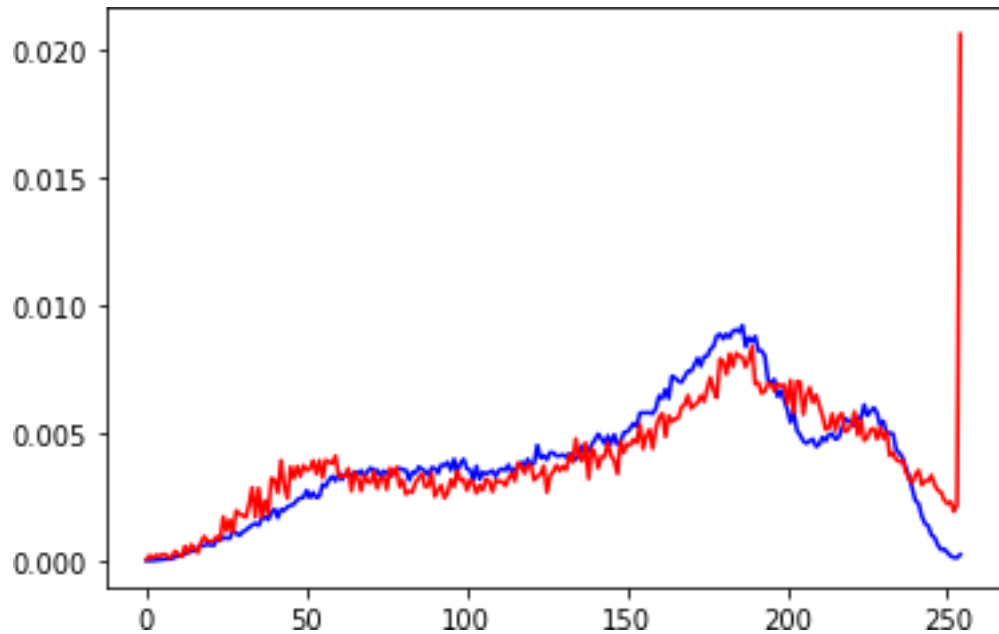


Fig. 43 Histogram of the melanoma lesion. The blue line is referred to the original image and the red line is referred to the final image obtained.

5.3. Segmentation.

The segmentation step is in charge of the lesion detection. The main aim is being able to detect where the skin lesion is and obtain a mask of the lesion, which means an image where all the pixels are black unless those that belong to the lesion.

To achieve it, a convolutional neural network, as U-Net, is used. As was explained before, U-Net architecture was developed for biomedical image segmentation, so it is one of the best solutions to the segmentation of skin lesions. As it is a concrete and complicated architecture, it was given by professor Vidas Raudonis, and it was just trained by me.

Table 5. U-Net model

```
1. def unet(pretrained_weights = None, input_size = (640,640,3)):
2.     inputs = Input(input_size)
3.     conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(inputs)
4.     conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv1)
5.     pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
6.         conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool1)
7.     conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv2)
8.     pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
9.     conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool2)
10.    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv3)
11.    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
12.    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool3)
13.    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv4)
14.    drop4 = Dropout(0.5)(conv4)
15.    pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)
16.
17.    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool4)
18.    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv5)
19.    drop5 = Dropout(0.5)(conv5)
20.
21.    up6 = Conv2D(512, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(drop5))
22.    merge6 = concatenate([drop4, up6], axis = 3)
23.    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge6)
24.    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv6)
25.25.
26.    up7 = Conv2D(256, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv6))
27.    merge7 = concatenate([conv3, up7], axis = 3)
28.
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge7)
29.    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv7)
30.
31.    up8 = Conv2D(128, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv7))
```

```

32. merge8 = concatenate([conv2,up8], axis = 3)
33. conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge8)
34. conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv8)
35. up9 = Conv2D(64, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv8))
36. merge9 = concatenate([conv1,up9], axis = 3)
37. conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge9)
38. conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv9)
39. conv9 = Conv2D(2, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv9)
40. conv10 = Conv2D(1, 1, activation = 'relu')(conv9)
41.
42. model = Model(input = inputs, output = conv10)
43.
44. model.compile(optimizer = Adam(lr = 1e-4), loss = 'binary_crossentropy', metrics = ['accuracy'])
45. model.compile(optimizer = Adam(lr = 1e-4), loss = 'mse', metrics = ['accuracy'])
46. model.compile(optimizer = Adam(lr = 1e4),
    loss = weighted_cross_and_dice_loss(0.5, 0.5), metrics = [mean_iou])
47.47.
48. model.summary()
49.
50. if(pretrained_weights):
51.     model.load_weights(pretrained_weights)
52.
53. return model

```

Analysing the model, we can see how the first five steps correspond to the left part of Fig.30, the contracting party; meanwhile, steps 6 to 10 correspond to the correct part of this image, the expansive path.

Once that the model is designed, the next step is its training.

Table 6. Model training

```

1. from model import *
2. from data import *
3. from losses import Loss
4. from tensorflow.keras.layers import ReLU
5.
6. image_size = 512
7.
8. data_gen_args = dict(rotation_range=0.2,
                       width_shift_range=0.05,
                       height_shift_range=0.05,
                       shear_range=0.05,
                       zoom_range=0.05,
                       horizontal_flip=True,
                       fill_mode='nearest')
9. myGene = trainGenerator(2,'D:/melonoma/forUnet_512x512/,
                          'image','label',data_gen_args,target_size = (image_size,image_size),save_to_dir =
None)

10. model = unet(input_size = (image_size,image_size,3))
11. model_checkpoint = ModelCheckpoint('unet_lesion_{ }x{ }.h5'.format(image_size,image_
size), monitor='loss',verbose=1, save_best_only=True)
12. model.fit_generator(myGene,steps_per_epoch=300,                                epochs=500,
                       callbacks=[model_checkpoint])

```

The training of the model is crucial. Firstly, the functions included are:

- Model: it is explained in the table__.
- Data: From data, the functions loaded are:

Table 7. Adjust data

```

1. def adjustData(img,mask,flag_multi_class,num_class):
2.     if(flag_multi_class):
3.         img = img / 255
4.         mask = mask[:, :, :, 0] if(len(mask.shape) == 4) else mask[:, :, 0]
5.         new_mask = np.zeros(mask.shape + (num_class,))
6.         for i in range(num_class):
7.             new_mask[mask == i,i] = 1
8.         new_mask = np.reshape(new_mask,(new_mask.shape[0],new_mask.shape[1]*
new_mask.shape[2],new_mask.shape[3])) if flag_multi_class else np.reshape(new_ma
sk,(new_mask.shape[0]*new_mask.shape[1],new_mask.shape[2]))
9.         mask = new_mask
10.
11. elif(np.max(img) > 1):
12.     img = img / 255
13.     mask = mask /255
14.     mask[mask > 0.5] = 1
15.     mask[mask <= 0.5] = 0
16. return (img,mask)

```


The function of Adjust Data is used to convert the data and the mask to the same size and shape.

Table 8. Train Generator.

```
1. def trainGenerator(batch_size,train_path,image_folder,mask_folder,aug_dict,im
   age_color_mode = "rgb", mask_color_mode = "grayscale",
   image_save_prefix = "image",mask_save_prefix = "mask",
   flag_multi_class = False,num_class = 2,
   save_to_dir = None,target_size = (256,256),seed = 1):
2.
3.   image_datagen = ImageDataGenerator(**aug_dict)
4.   mask_datagen = ImageDataGenerator(**aug_dict)
5.   image_generator = image_datagen.flow_from_directory(
       train_path,
       classes = [image_folder],
       class_mode = None,
       color_mode = image_color_mode,
       target_size = target_size,
       batch_size = batch_size,
       save_to_dir = save_to_dir,
       save_prefix = image_save_prefix,
       seed = seed)
6.   mask_generator = mask_datagen.flow_from_directory(
       train_path,
       classes = [mask_folder],
       class_mode = None,
       color_mode = mask_color_mode,
       target_size = target_size,
       batch_size = batch_size,
       save_to_dir = save_to_dir,
       save_prefix = mask_save_prefix,
       seed = seed)
7.   train_generator = zip(image_generator, mask_generator)
8.   for (img,mask) in train_generator:
9.     img,mask = adjustData(img,mask,flag_multi_class,num_class)
10.  yield (img,mask)
```

It generates images and masks simultaneously using the same speed for both of them to ensure that its transformation is the same.

- Losses: From it, the function “Binary Crossentropy” is loaded, and it includes the function used to calculate the loss in the model training.

Table 9. Binary Crossentropy

```
1. def binary_crossentropy(y_true, y_pred):
2.   loss = tf.reduce_mean(tf.keras.losses.binary_crossentropy(y_true = y_true, y_pred
   = y_pred, from_logits = False))
3.   return loss
```

After the loading of the function needed, the image size is specified; in this case, our images are (512, 512, 3). After that, the generation of the training data is done. In line 10, the U-Net model, which was declared in table 9, is used.

The output of the segmentation process are the image segmented and the image. It can be seen in Fig.44 and Fig.45.

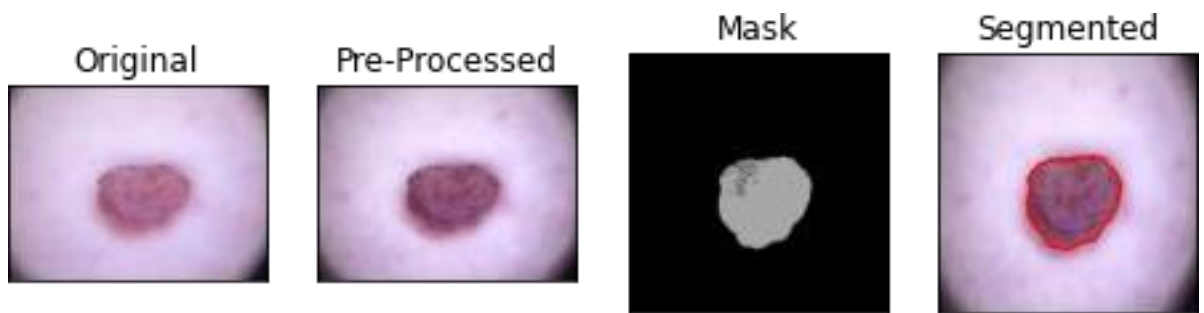


Fig. 44 Output of the segmentation process in the benign lesion image.

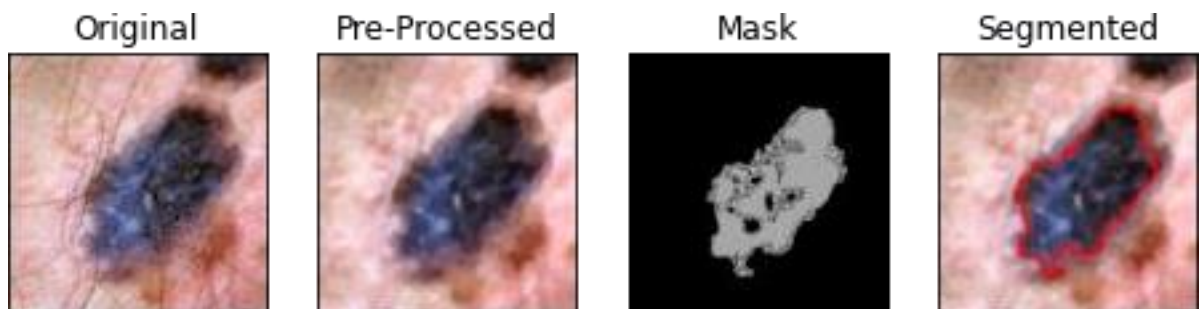


Fig. 45 Output of the segmentation process in the melanoma lesion image.

5.4. Feature Extraction

As was shown before, the ABCD rule is one of the most popular methods for evaluating possible melanoma lesions. The feature extraction is an essential aspect of the problem because the success or not of the classification is directly related to the accuracy of obtaining these parameters.

5.4.1. Image Preparation

Before the calculation of the parameters, the pre-processing images (obtained in the first step) and the masks (obtained in the segmentation step) are prepared. The following steps are:

1. Resize the image: To be sure that both, image, and mask have the same shape.

Table 10. Resize of the image and mask

```

1. width = 512
2. height = 512
3. dim = (width, height)
4. imagen = cv2.resize(imagen, dim, interpolation = cv2.INTER_AREA)
5. mask = cv2.resize(mask, dim, interpolation = cv2.INTER_AREA)

```

The function used is from the library Open CV, and it is:

cv2.resize(src, dsize [, dst [, fx [, fy[, interpolation]]]) [16]: It changes the dimensions of an image.

Its parameters are:

- src: input image.
- dsize: desired size for the output image.
- fx: scale factor along the horizontal axis.
- fy: scale factor along the vertical axis.
- interpolation: it is a flag which takes one of these methods: INTER_NEAREST, INTER_LINEAR, INTER_AREA, INTER_CUBIC or INTER_LANCZOS4.

The images obtained can be seen in the Fig.46 and Fig.47 image (A), which corresponds to the lesion image while the image (B) corresponds to the mask resized.

2. Obtain the negative of the mask: The colours of the mask, which are represented binary, have to be changed in order to obtain the lesion black and the background white. A morphological operation of the opening is done to eliminate the noise of the mask.

Table 11. Negative of the Mask

```

1. grey = cv2.cvtColor(mask, cv2.COLOR_BGR2GRAY)
2. _, mask1 = cv2.threshold(grey, 160, 255, cv2.THRESH_BINARY)
3.
4. row, col = grey.shape
5. negativo = np.zeros((row, col), dtype=np.uint8)
6.
7. for a in range(0, row):
8.     for b in range(0, col):
9.         negativo[a, b] = 255 - grey[a, b]
10.        if negativo[a,b] < 250:
11.            negativo[a,b] = 0
12.
13. kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (10,10))
14. negativo = cv.morphologyEx(negativo, cv.MORPH_CLOSE, kernel)

```

What are done here is, in the first line, the image is changed to greyscale. After that, an operation of threshold is done to convert the image to a binary channel (white and black). Then is calculated the shape of the matrix and this information is used to create an empty matrix of the same size as the mask. This empty matrix is filled using the structures “for,” which compares each pixel of the mask

with a threshold value (which does not matter his value due to our image is binary, so all the pixels have an intensity of 0 or 255). Finally, a closing (dilatation and erosion) morphological operation is done to eliminate the noise inside the white part of the mask.

The function used are:

cv.threshold(src, thresh, maxval, type[, dst]) [16]: It applies a fixed-level thresholding to a multiple-channel array.

- src: input array.
- dst: output array of the same size, type and number of channels than scr.
- thresh: threshold value.
- maxval: maximum value to use with the THRESH_BINARY and THRESH_BINARY_INV thresholding types.
- type: thresholding type.

.shape: It is a vector with information about the shape of the image; in a greyscale image, the first element of the vector is the number of rows and the second the number of columns. On the other hand, if it is a coloured image, the vector will have one element more, which is referred to as the number of colour channels.

numpy.zeros(shape, dtype=None, order='C', *, like=None) [18]: Which creates an array filled with ones.

- Shape: int or sequence of int. It is the shape of the new array.
- Type (optional): The desired data type for the array.
- order (optional): Whether to store multi-dimensional data in row-major or column-major order in memory.
- like: Reference object to allow the creation of arrays which are not NumPy arrays.

cv2.morphologyEx(src, op, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]) [16]: It perform advanced morphological operations as closing, opening, erode, dilatation...

- src: source image.
- dst: destination image of the same size and type than the input image.
- op: Type of morphological operation, as cv.MORPH_OPEN, cv.MORPH_CLOSE, cv.MORPH_GRADIENT...
- kernel: Structing element.
- anchor: anchor position with the kernel.
- iterations: number of times erosion and dilation are applied.
- borderType: Pixel extrapolation method.
- borderValue: Border value in cas of a constant border.

The result of this operation can be seen in Fig.46 and Fig.47 in the image (C)

3. Removal the skin of the background: The objective is to obtain an image with the lesion in colour while the background is white. To achieve it, the addition of the negative mask to the image pre-processed is done.

Table 12. Remove the background.

1. gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
2. th, threshed = cv2.threshold(gray, 240, 255, cv2.THRESH_BINARY_INV)

The image obtained correspond with the image (D) of Fig.46 and Fig.47

4. Image Cropped: It is essential to crop the image to focus our attention on the lesion, not in the background. The first step is to remove the noise with a morphological operation, then the contours of the lesion have to be found, and finally, the contours are used to approximate it as a rectangle to crop the image.

Table 13 . Crop Image.

<ol style="list-style-type: none">1. kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (11,11))2. morphed = cv2.morphologyEx(threshed, cv2.MORPH_CLOSE, kernel)3. cnts = cv2.findContours(morphed, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2]4. cnt = sorted(cnts, key=cv2.contourArea)[-1]5. x,y,w,h = cv2.boundingRect(cnt)6. lesion = img[y:y+h, x:x+w]
--

In lines 1 and 2, the morphological operation is done to reduce the noise that can appear after applying the thresholding operation. Then, the contours of the lesion are detected and sorted. Finally, the contours are used to obtain the shape of a rectangle as smaller as possible to, in the last step, crop the image. The functions used are:

cv2.findContours(image, mode, method [, contours[, hierarchy[, offset]]) [16]: It calculate the contours of the binary image.

- image: input source.
- mode: contour retrieval mode.
- method: contour approximation method.
- contours: detected contours.
- hierarchy: Optional output vector which contains the information about the image topology.
- offset: Optional offset by which every contour point is shifted.

cv2.contourArea(contour [, oriented]) [16]: This function computes a contour area. Its parameters are:

- contour: Input vector of “D points.
- oriented: Oriented flag area. If it is true, it returns a signed area.

sorted(iterable, key=None, reverse=False): It returns a list of the elements, of an array given, in a specific order.

- iterable: It is the input, and it can be a sequence, collection or any other iterator.
- reverse: if it is “True” the list is reversed. If it not specified, it will be “False”.
- key: It serves as a key for the sort comparison.

cv2.boundingRect(array) [16]: It calculate the minimal up-right bounding rectangle for the specified point set.

- array: input point set.

The result after this step can be seen in Fig.46 and Fig.47 in image (E).

5. Calculation of the number of black pixels: This measure will be useful to compare the number of pixels of the whole lesion with other measures.

Table 14. Calculate the number of black pixels.

<ol style="list-style-type: none">1. <code>lesionneg=cv2.cvtColor(lesion, cv2.COLOR_BGR2GRAY)</code>2. <code>row=lesionneg.shape[0]</code>3. <code>col=lesionneg.shape[1]</code>4.5. <code>th, lesionneg = cv2.threshold(lesionneg, 240, 255, cv2.THRESH_BINARY_INV)</code>6.7. <code>numpix = row*col</code>8. <code>count = cv2.countNonZero(lesionneg)</code>9. <code>black=(numpix-count)</code>
--

To count the number of black pixels, what must be done is firstly transform the image from BRG to greyscale, then, it should be changed to a binary scale. After that, it is possible to count the number of non-zero pixels. Finally, the number of non-zero pixels must be subtracted from the total number of pixels.

The image where the black pixels are calculated is the image (F) of Fig.46 and Fig.47.

6. Obtain the lesion in colour with a black background: It is so helpful to the feature extraction because if the background is white, the value of these pixels will be 255, affecting the calculations done in the histograms. Meanwhile, if the background is black, the value of these pixels will be zero, so it could not affect the calculations.

Table 15. Obtain the coloured image.

```
1. grey = cv2.cvtColor(lesion,cv2.COLOR_BGR2GRAY)
2. ret, lesionwhite = cv2.threshold(grey,0,255,cv2.THRESH_BINARY_INV)
3. thresh=cv2.cvtColor(lesionwhite,cv2.COLOR_GRAY2BGR)
4. #BLUE CHANNEL
5. threshb = thresh[:,:,0]
6. lesionb = lesion[:,:,0]
7. row = lesionb.shape[0]
8. col = lesionb.shape[1]
9. unionblue = np.zeros((row, col), dtype=np.uint8)
10. for i in range(0,row):
11.     for j in range(0,col):
12.         if (threshb[i,j] == 255):
13.             unionblue[i,j] = lesionb[i,j]
14.         if (threshb[i,j] == 0):
15.             unionblue[i,j]= 0
16. #GREEN CHANNEL
17. threshg = thresh[:,:,1]
18. lesiong = lesion[:,:,1]
19. row = lesiong.shape[0]
20. col = lesiong.shape[1]
21. uniongreen = np.zeros((row, col), dtype=np.uint8)
22. for i in range(0,row):
23.     for j in range(0,col):
24.         if (threshg[i,j] == 255):
25.             uniongreen[i,j] = lesiong[i,j]
26.         if (threshg[i,j] == 0):
27.             uniongreen[i,j]= 0
28.
29. #RED CHANNEL
30. threshr = thresh[:,:,2]
31. lesionr = lesion[:,:,2]
32. row = lesionr.shape[0]
33. col = lesionr.shape[1]
34. unionred = np.zeros((row, col), dtype=np.uint8)
35. for i in range(0,row):
36.     for j in range(0,col):
37.         if (threshr[i,j] == 255):
38.             unionred[i,j] = lesionr[i,j]
39.         if (threshr[i,j] == 0):
40.             unionred[i,j]= 0
41.
42. lesion_colour = cv2.merge((unionblue,uniongreen,unionred))
```

To change the colour of the background but keeping the lesion coloured, the process is the following: First, the negative of the image calculated before must be done, and this image is called “thresh”, and it will be transformed into a BGR image, then “thresh” and the lesion must be split into blue, green, and red scale. For each colour scale, a matrix of zeros has to be done, and it will be filled depending on if the pixel of the thresh:

- If the pixel of the “thresh” is equal to 255, the pixel on the new matrix created will have the same value as the same pixel of the lesion.
- If the pixel of the “thresh” is equal to 0, the pixel of the matrix will be 0.

It is important to remember that the pixels of the “thresh” image only has values of 255 or 0. When a binary image is transformed into a BRG image, it copies the same values to the three colour spaces without changing them.

This operation is the same for the three colour spaces, so finally, it only is necessary to merge the three images obtained.

The images (G) of Fig.46 and Fig.47 correspond to the negative of the binary lesion used to calculate the final-coloured lesions, which can be observed in the image (H) of Fig.46 and Fig.47.

It is a demonstration of how it is working in an image. To do it, the same images as before are used.

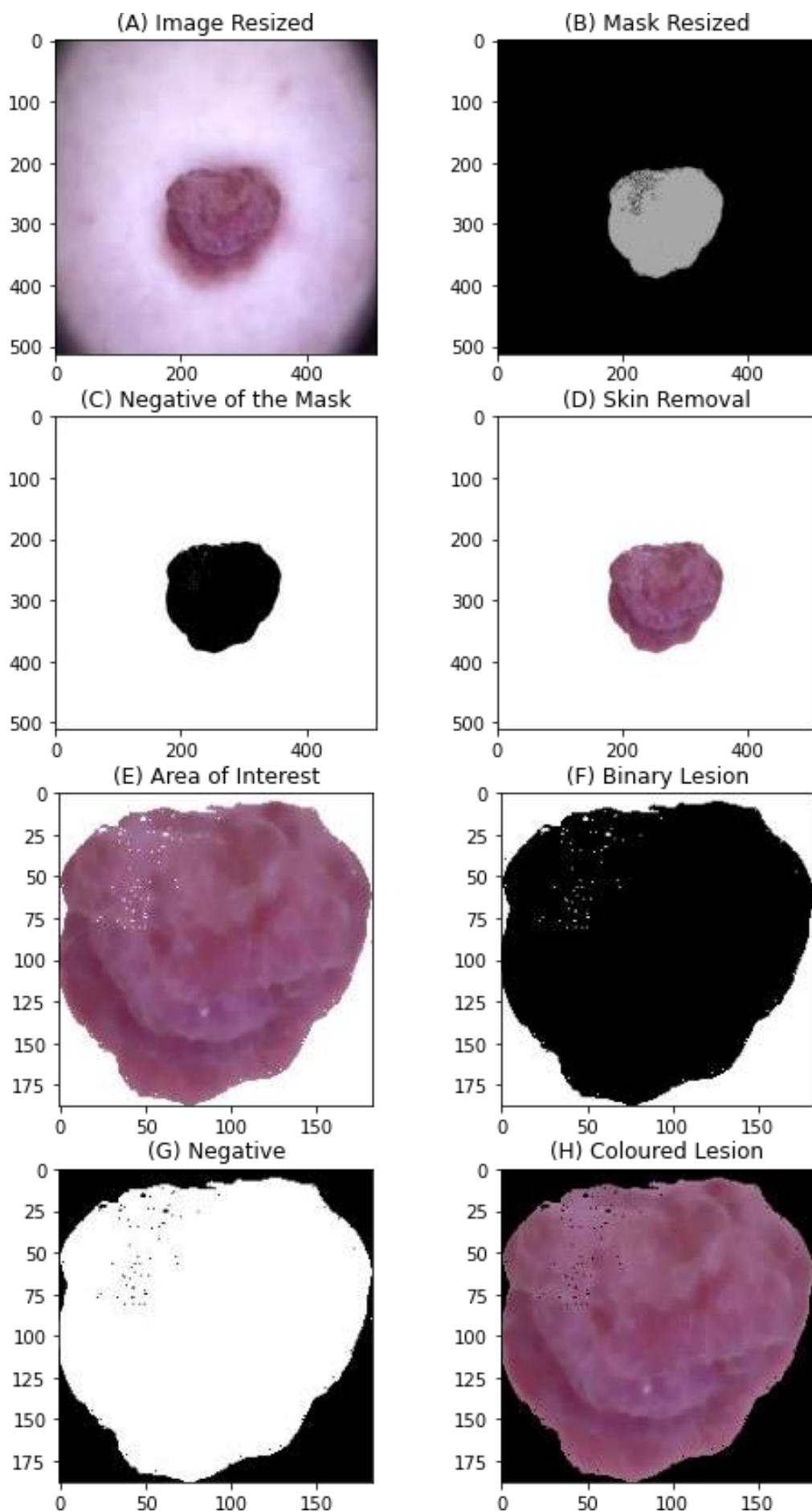


Fig. 46. Demonstration of the pre-process in a benign lesion.

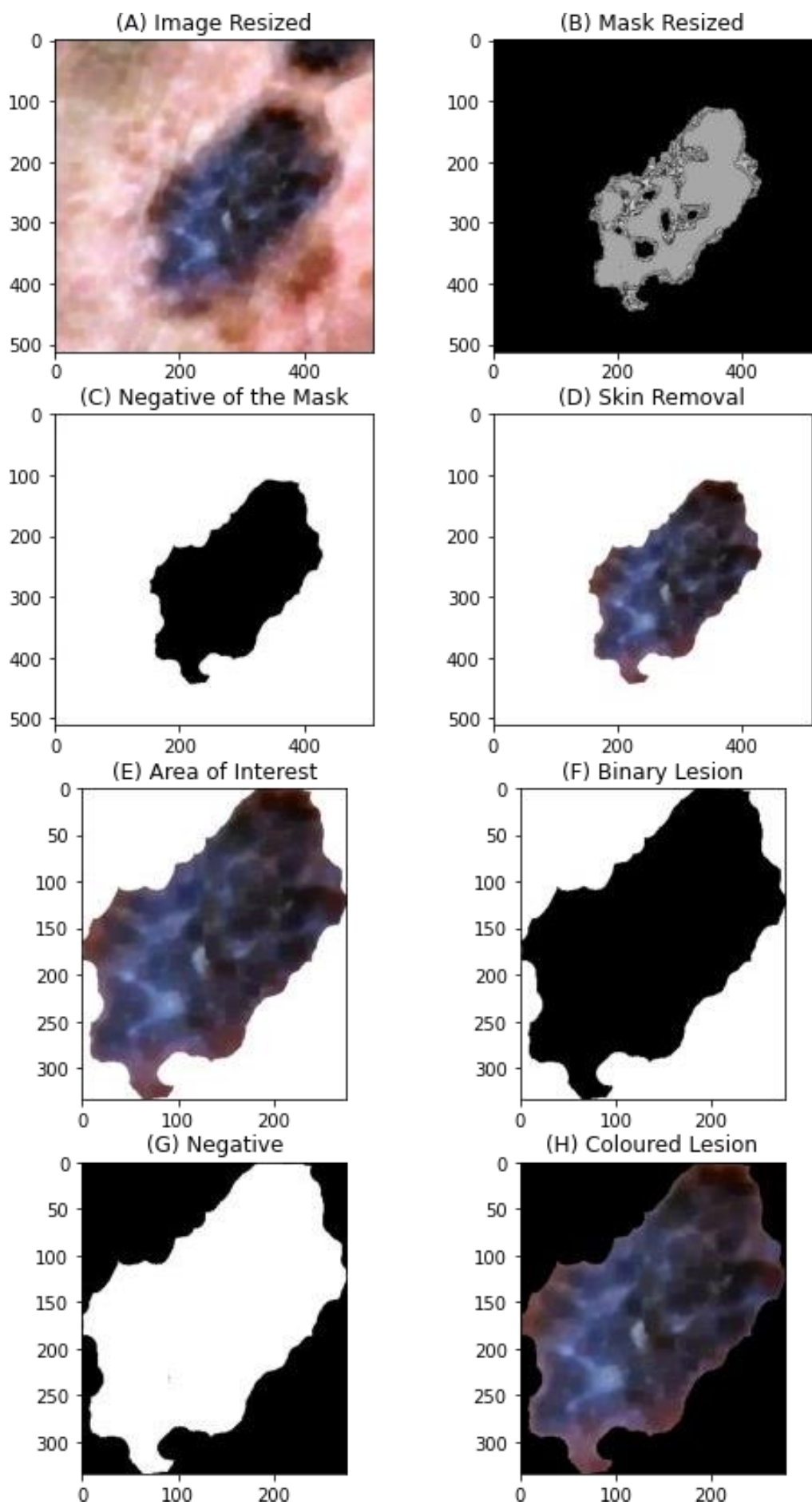


Fig. 47. Demonstration of the pre-process in a melanoma lesion.

When all the different images are obtained, it is possible to start with the feature extraction. The main idea about how to do it was extracted from the article “Melanoma Skin Cancer Detection based on Image Processing” by Zghal, N. S., & Derbel, N. [1].

The four parameters measured are:

5.4.2. Calculation of Asymmetry

It is evaluated according to two different parameters, and the final value is calculated doing the media between both form and colour.

5.4.2.1 Calculation of Form Asymmetry

The form asymmetry is related to the shape of the lesion, and it measures the grade of overlapping resulting from comparing the right/left and up/down parts of the images. It is calculated following the next steps:

1. The image is divided into four images: right, left, up and down parts.

Table 16. Division of the image in 4 regions

1. `row = lesion.shape[0]`
2. `col = lesion.shape[1]`
- 3.
4. `left = lesion[0:row,0:int(col/2)]`
5. `right = lesion[0:row,int(col/2):col]`
6. `up = lesion[0:int(row/2),0:col]`
7. `down = lesion[int(row/2):row,0:col]`

As an output, these images are obtained:

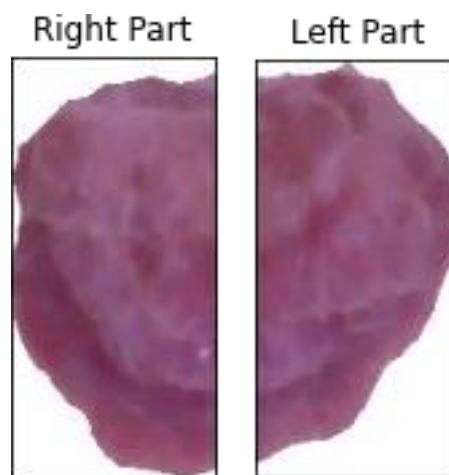


Fig. 48. Division of the benign image lesion in right and left part.

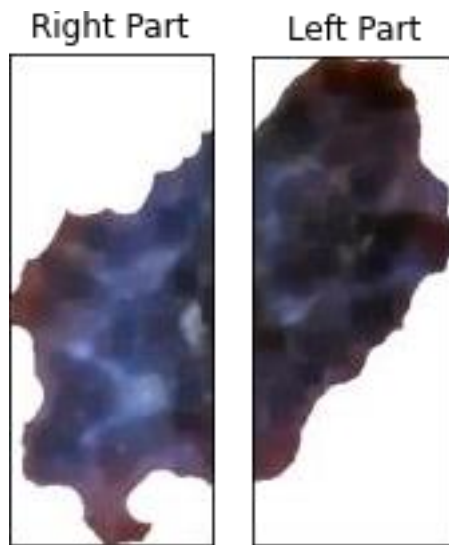


Fig. 49. Division of the melanoma image lesion in right and left part.

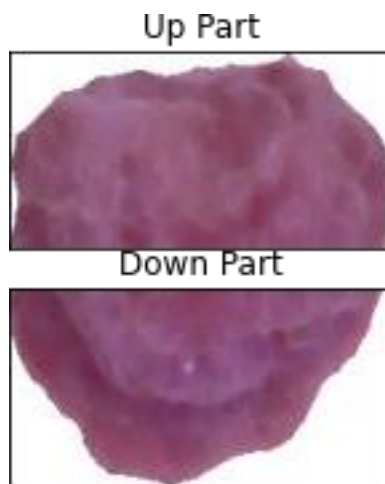


Fig. 50. Division of the benign image lesion in up and down part.

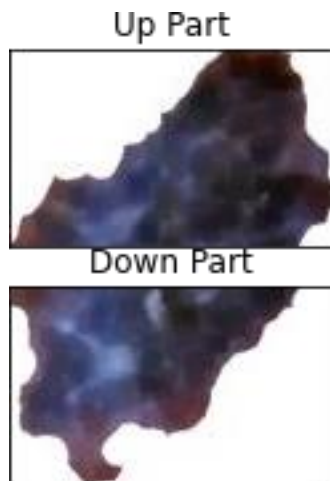


Fig. 51. Division of the melanoma image lesion in up and down part.

2. The left and down images are flip to the same orientation as right and up images, respectively to be able of comparing them.

Table 17. Inversion of the images.

1. `flipleft = cv2.flip(left,-1)`
2. `flipdown = cv2.flip(down,-1)`

The function used to do the inversion of the images is:

`cv2.flip(src, dst [, flipCode])` [16]: Reverses the order of the rows, columns or both in a matrix.

-src: matrix input.

-dst: destination array.

-flipCode: it indicates how to flip the array:

0 means flipping around the x-axis.

1 means flipping around the y-axis.

-1 means flipping in both axes.

As a result, it can be seen in Fig.52 and Fig.53 that now both right and left and down and up are in the same orientation, so we can compare them to obtain the overlap region.

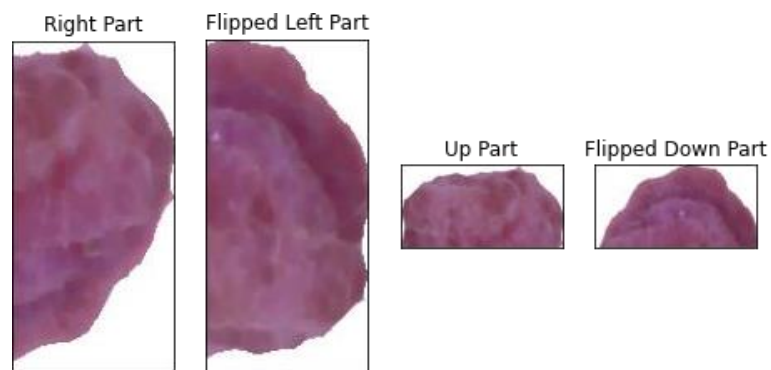


Fig. 52. Comparison of the right and up images with flipped left and down parts, respectively, in a benign lesion.

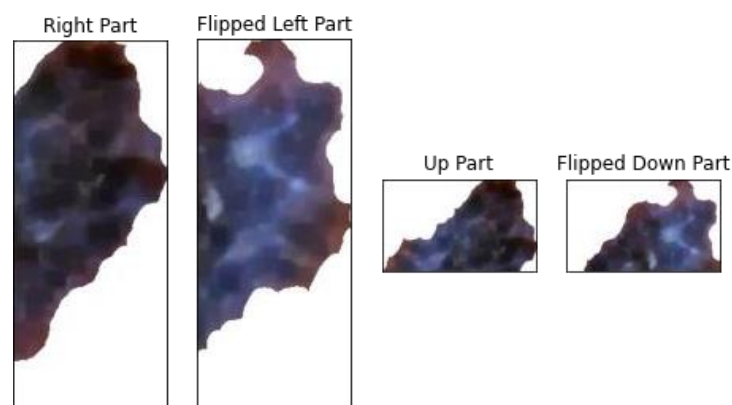


Fig. 53. Comparison of the right and up images with flipped left and down parts, respectively, in a melanoma lesion.

3. The overlapping region is calculated between right and left images. First, both are resized to the same size to avoid problems; then, the overlap region is calculated by subtracting the right part to the left part. After that, the overlapping region calculated is transformed to grey and then to binary scale. The next step is obtaining the negative of the overlapping to be able to count the non-zero pixel, subtract them to the total number of pixels of the image to obtain the number of black pixels. Finally, it is compared with the number of black pixels of all the lesion calculated before to achieve the percentage of overlap between right and left images. The same procedure is done with the down and up parts of the images.

Table 18. Percentage of overlapping RIGH-LEFT

```

1. col = flopleft.shape[0]
2. row = flopleft.shape[1]
3. size= (row,col)
4.
5. rflopleft=cv2.resize(flopleft,size)
6. rright=cv2.resize(right,size)
7.
8. overlap=cv2.subtract(rflopleft,rright)
9. grey = cv2.cvtColor(overlap, cv2.COLOR_BGR2GRAY)
   (thresh, im_bw) = cv2.threshold(grey, 128, 255, cv2.THRESH_BI
   NARY | cv2.THRESH_OTSU)
10.10.
11. row, col = im_bw.shape
12. negative = np.zeros((row, col), dtype=np.uint8)
13. for a in range(0, row):
14.     for b in range(0, col):
15.         negative[a, b] = 255 - im_bw[a, b]
16.         if negative[a,b] < 250:
17.             negative[a,b] = 0
18.
19. numpix = row*col
20. count = cv2.countNonZero(negative)
21. numpixneg=(numpix-count)
22. perct_LR = numpixneg*100/(black*2)

```

Table 19. Percentage of overlapping UP-DOWN

```
1. col = flipdown.shape[0]
2. row = flipdown.shape[1]
3. size= (row,col)
4.
5. rflipdown=cv2.resize(flipdown,size)
6. rup=cv2.resize(up,size)
7.
8. overlap=cv2.subtract(rflipdown, rup)
9. grey = cv2.cvtColor(overlap, cv2.COLOR_BGR2GRAY)
(thresh, im_bw) = cv2.threshold(grey, 128, 255, cv2.THRESH_BI
NARY | cv2.THRESH_OTSU)
10.10.
11. row, col = im_bw.shape
12. negative = np.zeros((row, col), dtype=np.uint8)
13. for a in range(0, row):
14.     for b in range(0, col):
15.         negative[a, b] = 255 - im_bw[a, b]
16.         if negative[a,b] < 250:
17.             negative[a,b] = 0
18.
19. numpix = row*col
20. count = cv2.countNonZero(negative)
21. numpixneg=(numpix-count)
22. perct_Ud = numpixneg*100/(black*2)
```

In the following images (Fig.54 and Fig.55), can be observed the overlapping region (white part of the image). In the left image is the overlapping between the right and left part of the image, while in the right image is the overlapping region between up and down parts of the image.

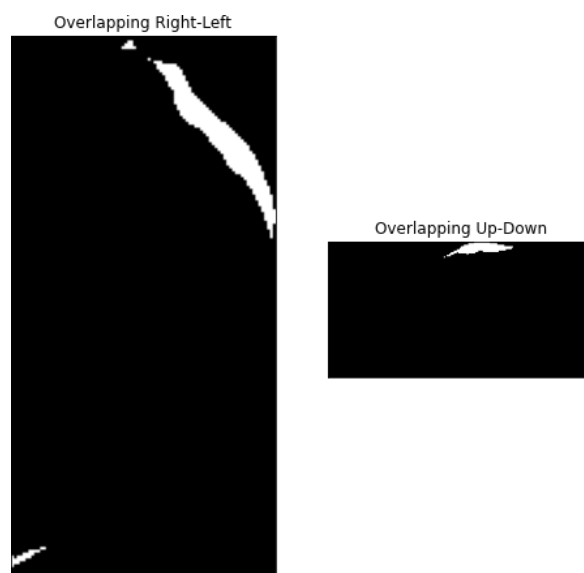
**Fig. 54.** Overlapping in the benign lesion.



Fig. 55. Overlapping in the melanoma lesion.

4. Finally, depending on if the percentages obtained are any of them, one or both of them higher than a threshold, the asymmetry form score will be 0, 1 or 2.

Formula [10]. Obtaining the asymmetry form score value. “Melanoma Skin Cancer Detection based on Image Processing” by Zghal, N. S., & Derbel, N. [1].

$$\forall A \quad \begin{cases} 0, & \text{OVL(L,R)} \leq T0 \text{ and } \text{OVL(U,d)} \leq T0 \\ 1, & \text{OVL(L,R)} \leq T0 \text{ or } \text{OVL(U,d)} \leq T0 \\ 2, & \text{otherwise} \end{cases}$$

Where T0 is the overlapping threshold, and it is equal to 5%

Table 20. Calculation of the overlapping percentage.

1. $A = 0$
2. **if** (perct_LR ≤ 5 **and** perct_Ud ≤ 5):
3. $A=0$
4. **elif** (perct_LR ≤ 5 **or** perct_Ud ≤ 5):
5. $A=1$
6. **else:**
7. $A=2$
- 8.
9. FormAsymmetry = A

The results obtained by each of the anterior images are represented in Fig.56 and Fig57.

```
percentage overlapping left-right = 3.190293848330975
percentage overlapping up-down = 1.4962956796803464
Form Asymmetry value = 0
```

Fig. 56. Asymmetry Form score of the benign lesion.


```
percentage overlapping left-right = 9.021563909071794
percentage overlapping up-down = 5.355932995350793
Form Asymmetry value = 2
```

Fig. 57. Asymmetry Form score of the melanoma lesion.

5.4.2.2 Calculation of Colour Asymmetry

This measure is related to the colour variation across the image. To measure it, the Chi distance is calculated according to the following formula:

Formula[11]. Chi Square Distance, according “Melanoma Skin Cancer Detection based on Image Processing” by Zghal, N. S., & Derbel, N. [1].

$$A_{color} = Chi_s = \sum_{i=1}^n \frac{(\text{histRight}(i) - \text{histLeft}(i))^2}{\text{histRight}(i) + \text{histLeft}(i)}$$

The histogram of the right part (histRight) is calculated for the right part of the images. Then, the three histograms are associated and divided into 3. The same procedure is followed with the left part of the lesion.

Table 21. Calculation of the histogram of the right part of the image

```
1. row = lesion_colour.shape[0]
2. col = lesion_colour.shape[1]
3.
4. #RIGHT HISTOGRAM
5. right = lesion_colour[0:row,int(col/2):col]
6. histr = np.concatenate([cv2.calHist([right],[c], None, [255],
   [1,256]) for c in [0,1,2]], axis=1)
7. histr = histr/histr.sum(axis=0)
8. histRight = histr.sum(axis=1)/3
9. plt.plot(histRight,'b')
```

Table 22. Calculation of the histogram of the left part of the image

```
1. left = lesion_colour[0:row,0:int(col/2)]
2. histl = np.concatenate([cv2.calcHist([left],[c], None, [255],
   [1,256]) for c in [0,1,2]], axis=1)
3. histl = histl/histl.sum(axis=0)
4. histLeft = histl.sum(axis=1)/3
5. plt.plot(histLeft,'r')
```

To calculate these histograms, the following functions are used:

np.concatenate ((a1, a2, ...), axis, out, dtype, casting) [18]: Its function is join a sequence of arrays along an existing axis.

- a1, a2,... : array.
- axis: The axis along which the arrays will be joined.
- out: The destination to place the result.
- dtype: The destination array of this dtype.
- casting: Controls what kind of data casting may occur.

cv2.calcHist((images, channels, mask, histSize, ranges[, hist[, accumulate]])) [16]: It calculates the histogram of one or more arrays.

- images: Source arrays.
- channels: List of the dims channels to compute the histogram.
- Mask: Optional mask.
- histSize: Array of histogram sizes in each dimension.
- ranges: Array of the dims arrays of the histogram bin boundaries in each dimension.
- hist: Output histogram.
- accumulate: Accumulation flag.

.sum(a, axis, dtype, out, keepdims, initial, where) [18]: Sum of array element over the given axis.

- a: elements to sum.
- axis: Axis or axes along which a sum is performed.
- dtype1: The type of the returned array and of the accumulator in which the elements are summed.
- out: Alternative output array in which to place the result.
- keepdims: if it is selected as true, the axes which are reduced are left in the result as dimensions with size one.
- initial: Starting value for the sum.
- where: Elements to include in the sum.

The histograms obtained of each of the two images studied are plotted in Fig.58 and Fig.59, the blue line corresponds to the right part while the blue is the left part.

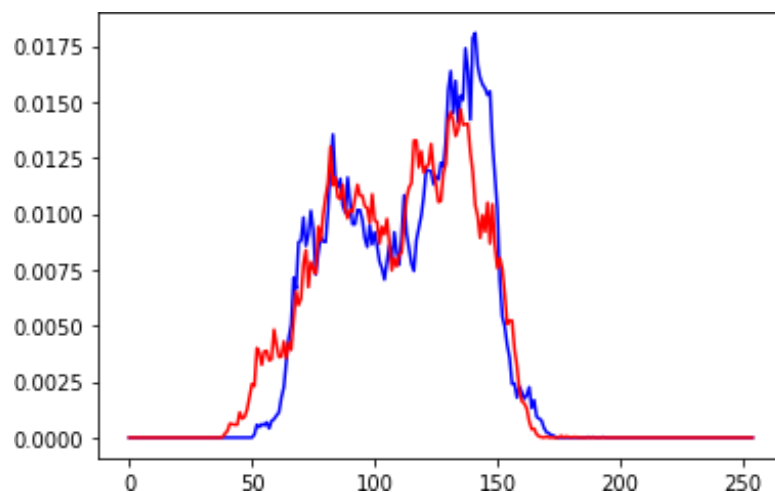


Fig. 58. Histogram of the benign lesion.

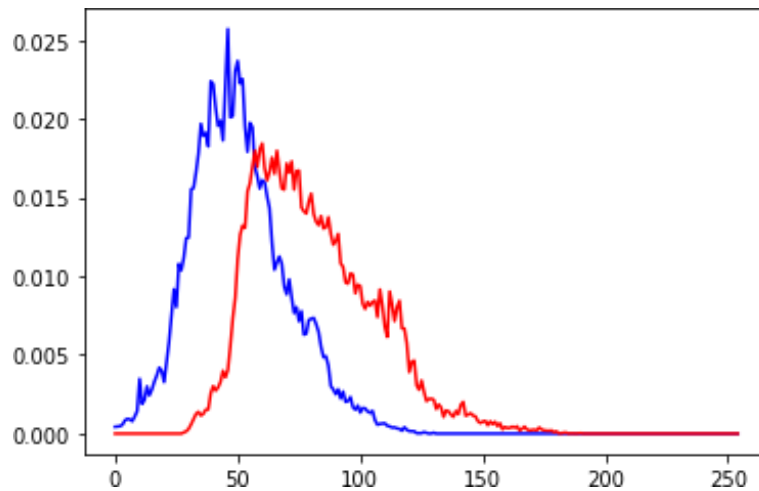


Fig. 59. Histogram of the melanoma lesion.

Finally, the Chi-square value is calculated with the formula shown before.

Table 23. Calculation of the Chi-Square distance between histograms.

```

1. ColourAsymmetry = 0
2.
3. for I in range(255):
4.     num = (histRight[i]-histLeft[i])**2
5.     den = (histRight[i]+histLeft[i])
6.     div = num/den
7.     if den == 0:
8.         continue
9.
10.    ColourAsymmetry = ColourAsymmetry + div
11.
12.    ColourAsymmetry = abs(ColourAsymmetry)
13.
14.    print("Colour Asymmetry value = ",ColourAsymmetry)

```

Firstly, the chi-square formula is applied for each pixel of the image. In each iteration, the value of colour asymmetry is added to the sum, and the final value of the asymmetry will be the colour asymmetry score.

The final value of asymmetry is calculated by doing the media between the form and colour asymmetry:

Formula [12]. Final asymmetry value

$$\text{Asymmetry Value Score} = \frac{\text{FormAsymmetry} + \text{ColourAsymmetry}}{2}$$

Passing this formula to the program, the calculation of the final asymmetry score is:

Table 24. Final asymmetry value.

1. `AsymmetryScore = (FormAsymmetry + ColourAsymmetry)/2`
2. `print("The asymmetry Value is = ", AsymmetryScore)`

5.4.3. Calculation of Border Irregularity

To calculate this parameter, the distance between the centre point of the lesion and each point of the perimeter of this lesion should be calculated. To achieve it, the lesion is divided into eight sectors, and the standard deviation (formula 13) is calculated for each sector. The border irregularity score will be the same as the number of sectors with a value of standard deviation higher than 20, which is the selected value.

Formula [13]. Standard deviation. [1]

$$s = \left(\frac{1}{n-1} \sum_{l=1}^n (x_i - \bar{x})^2 \right)^{1/2}$$

where

$$\bar{x} = \frac{1}{n} \sum_{l=1}^n x_i$$

And n is the number of elements in the sector.

To calculate this parameter, firstly, the biggest contour and lesion are found.

Table 25. Find the biggest contour and lesion

1. `blkwhite = np.zeros(np.shape(bw),dtype=np.uint8)`
2. `ctrs, hierh = cv2.findContours(bw,mode=cv2.RETR_EXTERNAL, method=cv2.CHAIN_APPROX_SIMPLE)`
- 3.
4. `AreaMax = -1`
5. `idx = -1`
6. `for i in range(0,len(ctrs)):`
7. `cnt = ctrs[i]`
8. `area = cv2.contourArea(cnt)`
9. `if AreaMax < area:`
10. `AreaMax = area`
11. `idx = i`

When it is found, what have to be done is get the bounding box, then, make an image black and white from contours. Using the bounding box calculated, the image should be cropped.

Table 26. Crop the lesion area.

```

1. if idx > -1:
2.   cnt = ctrs[idx]
3.   bbox = cv2.boundingRect(cnt)
4.
5.   cv2.drawContours(blckwhite, [cnt], -1, (255, 255, 255), -1)
6.
7.   lesion = blckwhite[bbox[1]:bbox[1]+bbox[3],bbox[0]:bbox[0]+bbox[2]]
8.   gray_lesion = gray[bbox[1]:bbox[1]+bbox[3],bbox[0]:bbox[0]+bbox[2]]
9.   rgb_lesion = img[bbox[1]:bbox[1]+bbox[3],bbox[0]:bbox[0]+bbox[2],:]

```

cv2.drawContours((image, contours, contourIdx, color[, thickness[, lineType[, hierarchy[, maxLevel[, offset]]]])) [16]: Its function is draw contour outlines or fill them. Its parameters are described below.

- image: Destination image.
- contours: The inputs contours which have to be drawn.
- contourIdx: If it is negative, all the contours are drawn.
- colour: Colour of the contours.
- thickness: The thickness of lines the contours are drawn with.
- lineType: Line connectivity.
- hierarchy: Optional information about hierarchy.
- maxLevel: Maximal level for drawn contours.
- Offset: Optional contour shift parameter.

Once that the cropped image is done, what is needed is divide the lesion into the eight sectors.

Table 27. Division of the lesion in eight sectors.

```

1. angles = [0, 45, 90, 135, 180, 225, 270, 315]
2. for I in range(0,len(angles)):
3.   cx = int(bbox[2]*0.5)
4.   cy = int(bbox[3]*0.5)
5.   ccx = int(cx + 100 * np.cos(np.radians(angles[i])))
6.   ccy = int(cy + 100 * np.sin(np.radians(angles[i])))
7.   cv2.line(77esion,(cx,cy),(ccx,ccy),(0,0,0),2)

```

Finally, the standard deviation is obtained for each sector; if it is bigger than 20, the border irregularity score is incremented by one. Due to not always the eight sectors are done, the final border score is calculated by the comparison between the score value obtained, and the number of sectors detected.

Table 28. Calculation of the border irregularity score.

```

1. ctrs_8, hierh_8 = cv2.findContours(78esion,mode=cv2.RETR_EXTERNAL,method=cv2.
   CHAIN_APPROX_SIMPLE)
2. h, w = np.shape(78esion)
3.
4. #get each pixel inside of one contour
5. std_score = 0
6. std_no = 0
7. for I in range(0,len(ctrs_8)):
8.     cnt8 = ctrs_8[i]
9.     area = cv2.contourArea(cnt8)
10.    if area > 20:
11.        tmp_bw = np.zeros(np.shape(78esion),dtype=np.uint8)
12.        cv2.drawContours(tmp_bw, [cnt8], -1, (255, 255, 255), -1)
13.        flt_bw = np.asarray(tmp_bw).reshape((h*w,1))
14.        flt_gray = np.asarray(gray_lesson).reshape((h*w,1))
15.
16.        #list of pixels in the region
17.
18.        print("mean pixel value: {}, std: {}".format
   (np.mean(flt_gray[flt_bw==255]), np.std(flt_gray[flt_bw==255])))
18.18.
19.        std_value = np.std(flt_gray[flt_bw==255])
20.        std_no = std_no + 1
21.        if (std_value > 20):
22.            std_score= std_score+ 1
23.
24.        #make black and white image from countour
25.        cv2.drawContours(rgb_lesson, ctrs_8, -1, (255, 255, 255), 1)
26.
27. BorderScore = (std_score/std_no)*8

```

The library NumPy has very useful commands which had helped in the calculation of the standard deviations.

np.mean(a, axis, dtype, out, keepdims*, where) [18]: It computes the arithmetic mean along the specified axis.

- a: array of the numbers whose mean is desired.
- axis: Axis or axes along which the mean is computed.
- dtype: Type to use in computing the mean.
- out: alternate output in which to place the result.
- keepdims: If this is set to True, the axes which are reduced are left in the result as dimensions with size one.
- where: Elements to include in the mean.

np.std (a, axis, dtype, out, ddof, keepdims, *, where) [18]: It computes the standard deviation along the specified axis. The parameters needed are:

- a: inputs values.
- axis: Axis or axes along which the standard deviation is computed.
- dtype: Type to use in computing the standard deviation.

- out: Alternative output array in which place the result.
- ddof: It is referred to the Delta degrees of freedom.
- keepdims: If this is set to True, the axes which are reduced are left in the result as dimensions with size one.
- where: Elements to include in the standard deviation.

5.4.4. Calculation of Colour

In order to measure this parameter, the idea is to calculate the distance between the colours presented in the lesion and the colours of reference. As reference colour, the following table took of the article “Melanoma Skin Cancer Detection based on Image Processing” by Zghal, N. S., & Derbel, N. [1] was used. So, there are six reference colours that will be compared with the colours of the lesion; these colours are White, Black, Red, Light-Brown, Dark-Brown and Blue-Grey.

White	Black	Red	Light-Brown	Dark-Brown	Blue-Gray
(1,1,1)	(0,0,0)	(0.7843, 0.5882, 0.3922)	(0.7843, 0.5882, 0.3922)	(0.5882, 0.3922, 0.3922)	(0.5882, 0.4902, 0.5882)
(0.9608, 0.9608, 0.9608)	(0.0392, 0.0392, 0.0392)	(1,0.1961, 0.1961)	(0.7843, 0.3922,0)	(0.4902, 0.2941, 0.2941)	(0.4902, 0.4902, 0.5882)
(0.9216, 0.9216, 0.9216)	(0.0784, 0.0784, 0.0784)	(0.7843, 0, 0)	(0.7843, 0.3922, 0.1961)	(0.3922,0.1961, 0.1961)	(0.3922, 0.3922, 0.4902)
(0.8824, 0.8824, 0.8824)	(0.1176, 0.1176, 0.1176)	(0.7843, 0.1961, 0.1961)	(0.5882, 0.3922, 0.1961)	(0.3922, 0.1961, 0)	(0.3922, 0.4902, 0.5882)
(0.8431, 0.8431, 0.8431)	(0.1569, 0.1569, 0.1569)	(0.5882, 0, 0)	(0.5882, 0.3922, 0)	(0.3922, 0,0)	(0.1961, 0.3922, 0.5882)
(0.8039, 0.8039, 0.8039)	(0.1961, 0.1961, 0.1961)	(0.5882, 0.1961, 0.1961)	(0.5882, 0.1961, 0)	(0.1961, 0,0)	(0, 0.3922, 0.5882)

Fig. 60. Reference colours. “Melanoma Skin Cancer Detection based on Image Processing” by Zghal, N. S., & Derbel, N. [1]

Firstly, this table must be introduced. To achieve it, six matrixes were created, one of each colour and is calculated for each reference colour the mean of each colour (R, G, B), then the threshold value for each reference colour is calculated with the following formula.

Formula [14]. Threshold value of reference [1].
$\text{ThresholdValue} = \sqrt{3 * (\text{Max value} - \text{Min value})^2}$

The program is split for each reference colour.

Table 29. White colour.

```

1. x1 = np.zeros((6, 3))
2. x1[0] = (1,1,1)
3. x1[1] = (0.9608, 0.9608, 0.9608)
4. x1[2] = (0.9216, 0.9216, 0.9216)
5. x1[3] = (0.8824, 0.8824, 0.8824)
6. x1[4] = (0.8431, 0.8431, 0.8431)
7. x1[5] = (0.8039, 0.8039, 0.8039)
8.
9. Twhite = math.sqrt(3*(np.amax(x1)-np.amin(x1))**2)
10. rwhite=(x1[0,0] + x1[1,0] + x1[2,0] + x1[3,0] + x1[4,0] + x1[5,0])/6
11. gwhite=(x1[0,1] + x1[1,1] + x1[2,1] + x1[3,1] + x1[4,1] + x1[5,1])/6
12. bwhite=(x1[0,2] + x1[1,2] + x1[2,2] + x1[3,2] + x1[4,2] + x1[5,2])/6

```

Table 30. Black colour.

```

1. x2 = np.zeros((6, 3))
2. x2[0] = (0,0,0)
3. x2[1] = (0.0392, 0.0392, 0.0392)
4. x2[2] = (0.0784, 0.0784, 0.0784)
5. x2[3] = (0.1176, 0.1176, 0.1176)
6. x2[4] = (0.1569, 0.1569, 0.1569)
7. x2[5] = (0.1961, 0.1961, 0.1961)
8.
9. Tblack = math.sqrt(3*(np.amax(x2)-np.amin(x2))**2)
10. rblack=(x2[0,0] + x2[1,0] + x2[2,0] + x2[3,0] + x2[4,0] + x2[5,0])/6
11. gblack=(x2[0,1] + x2[1,1] + x2[2,1] + x2[3,1] + x2[4,1] + x2[5,1])/6
12. bblack=(x2[0,2] + x2[1,2] + x2[2,2] + x2[3,2] + x2[4,2] + x2[5,2])/6

```

Table 31. Red colour.

```

1. x3 = np.zeros((6, 3))
2. x3[0] = (0.7843, 0.5882, 0.3922)
3. x3[1] = (1.0000, 0.1961, 0.1961)
4. x3[2] = (0.7843, 0.0000, 0.0000)
5. x3[3] = (0.7843, 0.1961, 0.1961)
6. x3[4] = (0.5882, 0.0000, 0.0000)
7. x3[5] = (0.5882, 0.1961, 0.1961)
8.
9. Tred = math.sqrt(3*(np.amax(x3)-np.amin(x3))**2)
10. rred =(x3[0,0] + x3[1,0] + x3[2,0] + x3[3,0] + x3[4,0] + x3[5,0])/6
11. gred =(x3[0,1] + x3[1,1] + x3[2,1] + x3[3,1] + x3[4,1] + x3[5,1])/6
12. bred =(x3[0,2] + x3[1,2] + x3[2,2] + x3[3,2] + x3[4,2] + x3[5,2])/6

```


Table 32. Light-Brown colour.

```

1. x4 = np.zeros((6, 3))
2. x4[0] = (0.7843, 0.5882, 0.3922)
3. x4[1] = (0.7843, 0.3922, 0.0000)
4. x4[2] = (0.7843, 0.3922, 0.1961)
5. x4[3] = (0.5882, 0.3922, 0.1961)
6. x4[4] = (0.5882, 0.3922, 0.0000)
7. x4[5] = (0.5882, 0.1961, 0.0000)
8.
9. Tlightbrown = math.sqrt(3*(np.amax(x4)-np.amin(x4))**2)
10. rlightbrown =(x4[0,0] + x4[1,0] + x4[2,0] + x4[3,0] + x4[4,0] +x4[5,0])/6
11. glightbrown =(x4[0,1] + x4[1,1] + x4[2,1] + x4[3,1] + x4[4,1] +x4[5,1])/6
12. blightbrown =(x4[0,2] + x4[1,2] + x4[2,2] + x4[3,2] + x4[4,2] +x4[5,2])/6

```

Table 33. Dark-Brown colour.

```

1. x5 = np.zeros((6, 3))
2. x5[0] = (0.5882, 0.3922, 0.3922)
3. x5[1] = (0.4902, 0.2941, 0.2941)
4. x5[2] = (0.3922, 0.1961, 0.1961)
5. x5[3] = (0.3922, 0.1961, 0.0000)
6. x5[4] = (0.3922, 0.0000, 0.0000)
7. x5[5] = (0.1961, 0.0000, 0.0000)
8.
9. Tdarkbrown = math.sqrt(3*(np.amax(x5)-np.amin(x5))**2)
10. rdarkbrown =(x5[0,0] + x5[1,0] + x5[2,0] + x5[3,0] + x5[4,0] +x5[5,0])/6
11. gdarkbrown =(x5[0,1] + x5[1,1] + x5[2,1] + x5[3,1] + x5[4,1] +x5[5,1])/6
12. bdarkbrown =(x5[0,2] + x5[1,2] + x5[2,2] + x5[3,2] + x5[4,2] +x5[5,2])/6

```

Table 34. Blue-Gray colour.

```

1. x6 = np.zeros((6, 3))
2. x6[0] = (0.5882, 0.4902, 0.5882)
3. x6[1] = (0.4902, 0.4902, 0.5882)
4. x6[2] = (0.3922, 0.3922, 0.4902)
5. x6[3] = (0.3922, 0.4902, 0.5882)
6. x6[4] = (0.1961, 0.3922, 0.5882)
7. x6[5] = (0.0000, 0.3922, 0.5882)
8.
9. Tbluegray = math.sqrt(3*(np.amax(x6)-np.amin(x6))**2)
10. rbluegray =(x6[0,0] + x6[1,0] + x6[2,0] + x6[3,0] + x6[4,0] +x6[5,0])/6
11. gbluegray =(x6[0,1] + x6[1,1] + x6[2,1] + x6[3,1] + x6[4,1] +x6[5,1])/6
12. bbluegray =(x6[0,2] + x6[1,2] + x6[2,2] + x6[3,2] + x6[4,2] +x6[5,2])/6

```

To calculate the distance between reference colour and the colour of each pixel of the lesion, the Euclidian distance between them is calculated:

Formula [14]. Euclidean distance “Melanoma Skin Cancer Detection based on Image Processing” by Zghal, N. S., & Derbel, N. [1]

$$Ck(i, j) = \sqrt{(rk - rij)^2 + (gk - gij)^2 + (bk - bij)^2}$$

Where r_k is the mean of the red component, g_k of the green and b_k of the blue component of the reference colour.

So, this operation will be repeated six times, one for each reference colour.

The value C_k is compared with the threshold value of each reference colour calculated before. If the C_k is smaller than the threshold value, the Colour Score will be incremented in one.

As in the last steps, the calculation for each colour reference must be done individually.

Table 35. C_k value for white colour.

```

1. Ckw = 0
2. for i in range(0,row):
3.     for j in range(0,col):
4.
5.         b = (bwhite - lesion_colour[i,j,0]/255)**2
6.         g = (gwhite - lesion_colour[i,j,1]/255)**2
7.         r = (rwhite - lesion_colour[i,j,2]/255)**2
8.         Ckw = Ckw+ math.sqrt(b+g+r)
9.
10. Ckwhite = Ckw/(row*col)
11.
12. if (Ckwhite >= Twhite):
13.     ColourScore = ColourScore + 1

```

Table 36. C_k value for black colour.

```

1. Ckb = 0
2. for i in range(0,row):
3.     for j in range(0,col):
4.
5.         b = (bblack - lesion_colour[i,j,0]/255)**2
6.         g = (gblack - lesion_colour[i,j,1]/255)**2
7.         r = (rblack - lesion_colour[i,j,2]/255)**2
8.         Ckb = Ckb+ math.sqrt(b+g+r)
9.
10. Ckblack = Ckb/(row*col)
11.
12. if (Ckblack >= Tblack):
13.     ColourScore = ColourScore + 1

```

Table 37. Ck value for red colour.

```

1. Ckr = 0
2. for I in range(0,row):
3.     for j in range(0,col):
4.
5.         b = (bred - lesion_colour[I,j,0]/255)**2
6.         g = (gred - lesion_colour[I,j,1]/255)**2
7.         r = (rred - lesion_colour[I,j,2]/255)**2
8.         Ckr = Ckr+ math.sqrt(b+g+r)
9.
10. Ckred = Ckr/(row*col)
11.
12. if (Ckred >= Tbred):
13.     ColourScore = ColourScore + 1

```

Table 38. Ck value for light-brown colour.

```

1. Ckl = 0
2. for I in range(0,row):
3.     for j in range(0,col):
4.
5.         b = (blightbrown - lesion_colour[I,j,0]/255)**2
6.         g = (glightbrown - lesion_colour[I,j,1]/255)**2
7.         r = (rlightbrown - lesion_colour[I,j,2]/255)**2
8.         Ckl = Ckl+ math.sqrt(b+g+r)
9.
10. Cklightbrown = Ckl/(row*col)
11.
12. if (Cklightbrown >= Tlightbrown):
13.     ColourScore = ColourScore + 1

```

Table 39. Ck value for dark-brown colour.

```

1. Ckd = 0
2. for I in range(0,row):
3.     for j in range(0,col):
4.
5.         b = (bdarkbrown - lesion_colour[I,j,0]/255)**2
6.         g = (gdarkbrown - lesion_colour[I,j,1]/255)**2
7.         r = (rdarkbrown - lesion_colour[I,j,2]/255)**2
8.         Ckd = Ckd+ math.sqrt(b+g+r)
9.
10. Ckdarkbrown = Ckd/(row*col)
11.
12. if (Ckdarkbrown >= Tdarkbrown):
13.     ColourScore = ColourScore + 1

```

Table 40. Ck value for blue-gray colour.

```
1. Ckbg = 0
2. for I in range(0,row):
3.     for j in range(0,col):
4.
5.         b = (bbluegray - lesion_colour[I,j,0]/255)**2
6.         g = (gbluegray - lesion_colour[I,j,1]/255)**2
7.         r = (rbluegray - lesion_colour[I,j,2]/255)**2
8.         Ckbg = Ckbg+ math.sqrt(b+g+r)
9.
10. Ckbluegray = Ckbg/(row*col)
11.
12. if (Ckbluegray >= Tdarkbrown):
13.     ColourScore = ColourScore + 1
```

5.4.5. Calculation of Diameter

Due to there are not data about the real size of the lesion and is not possible to calculate it because there is not also information about the procedure of how the images were taken, an estimation was done trying to have a dimensionless parameter, as bigger it is as more considerable is also the lesion. The procedure is:

First, the major axis is calculated comparing the number of pixels of the row and column, then the parameter is estimated with the following formula:

Formula [15]. Estimation of the diameter.

$$M = \frac{\text{majoraxis} * 2.65}{100}$$

In the program, it is done by the comparison between rows and columns. If rows are bigger than the column's value, rows will be the major axis; if not, the column's value will be the major axis. Then, just the formula has to be introduced.

Table 41. Diameter Calculation

```
1. row = lesion.shape[0]
2. col = lesion.shape[1]
3.
4. if (row>=col):
5.     majoraxis = row
6. else:
7.     majoraxis = col
8.
9. M = (majoraxis * 2.65)/100
```

Finally, depends on the M value, the Diameter Score is given.

Table 42. Diameter score value calculation.

```

1. DiameterScore=0
2.
3. if (M <= 2):
4.   DiameterScore = 0.5
5. elif (M <= 3):
6.   DiameterScore = 1
7. elif (M <= 4):
8.   DiameterScore = 1.5
9. elif (M <= 5):
10.  DiameterScore = 2
11. elif (M <= 6):
12.  DiameterScore = 2.5
13. elif (M <= 7):
14.  DiameterScore = 3
15. elif (M <= 8):
16.  DiameterScore = 3.5
17. elif (M <= 9):
18.  DiameterScore = 4
19. elif (M <= 10):
20.  DiameterScore = 4.5
21. else:
22.  DiameterScore = 5
23.
24. print("Diameter value = ",DiameterScore)

```

5.5. Classification.

When all the feature parameters are calculated, the last step is to calculate the Total Dermoscopy Value (TDV), which is obtained using the algorithm presented by Fig.61.

Criterion	Description	Score	Weight factor
Asymmetry	Assess not only contour, but also colors	0-2	X 1.3
Border	Abrupt ending of pigment pattern at the periphery in 0-8 segments	0-8	X 0.1
Color	Presence of up to 6 colors (white, red, light brown, dark brown, blue-gray, black)	1-6	X 0.5
Diameter	>6mm indicates the presence of melanoma	0.5-5	X 0.5

Fig. 61. Criteria of ABCD rule. “Melanoma Skin Cancer Detection based on Image Processing” by Zghal, N. S., & Derbel, N. [1].

Finally, the TDV value must be evaluated because depending on its value the lesion will be classify as a benign, suspicious or melanoma, following the next algorithm.

Formula[16]. Classification [1]

$TDV < 4.75 \quad \rightarrow \textit{Benign Lesion}$
 $4.75 \leq TDV < 5.45 \quad \rightarrow \textit{Suspicious Lesion}$
 $5.75 \leq TDV \quad \rightarrow \textit{Melanoma Lesion}$

The algorithm is calculated to set us in the worst of the cases.

Table 43. Calculation of the Total Dermatoscopic Value (TDV) and classification

```
1. TDV = AsymmetryScore * 1.3 + BorderScore * 0.1 + ColourScore * 0.5 + DiameterScore * 0.5
2.
3. print("TDV = ", TDV)
4.
5. if (TDV < 4.75):
6. print("The lesion is benign")
7. elif (4.75 <= TDV < 5.45):
8. print("The lesion is suspicious")
9. else:
10. print("The lesion is malignant")
```

6. Experimental Investigation.

Finally, some experiments were done to evaluate how is working the program. To achieve experimentation, 60 images were taken from the dataset for testing the program, this image were not introduced in the testing set used to training of the U-Net. Half of these images correspond to benign lesions while the other half to melanoma lesion.

In the following tables (Table 44 and Table 45), the results obtained are shown. In the first column is the name of the image tested. In the second one is the asymmetry score calculated according to the formula 12. The next column is the border irregularity, which is based in the standard deviation, that is calculating with the formula 13. The next one is the colour score, which is calculated comparing the Euclidian distance between the reference colours of the image with a threshold value. The last parameter is shown in the next column. The diameter score is estimated according to the formula 15. The next is the TDV score, which is the “Total Dermoscopy Value” and it is calculated following the formula 16. Finally, the classification result is presented, in green are the ones which are detected correctly and in yellow the ones that are detected as a suspicious lesion.

In the table 44, the analysis of the benign lesions is done while in the table 45, the melanoma lesions are tested.

Table 44. Results of the benign lesions.

NAME	ASYMMETRY	BORDER	COLOUR	DIAMETER	TDV	RESULT
ISIC_0024335	0,6010	0	4	2	3,5312	Benign
ISIC_0024533	0,0357	0	5	2	3,5464	Benign
ISIC_0024597	0,0349	0	5	2	3,5454	Benign
ISIC_0024598	0,0605	0	4	2	2,8287	Benign
ISIC_0024837	0,0200	0	4	2	2,7760	Benign
ISIC_0025428	0,0451	2	5	2	3,7587	Benign
ISIC_0024839	0,0161	0	4	3	3,2710	Benign
ISIC_0025763	0,0560	2	4	2	3,2728	Benign
ISIC_0026210	0,0103	0	4	4	4,0134	Benign
ISIC_0026726	0,0552	0	5	2	3,5718	Benign
ISIC_0024402	0,2183	0	5	2	3,3919	Benign
ISIC_0024447	0,0510	1	5	3	3,9162	Benign
ISIC_0024524	1,0230	1	4	2	4,1799	Benign
ISIC_0024933	0,6381	3	4	2	3,8795	Benign
ISIC_0025033	0,0868	1	5	2	3,4628	Benign
ISIC_0025057	0,0604	0	4	3	3,3286	Benign
ISIC_0025172	0,1427	0	5	1	3,1855	Benign
ISIC_0025300	0,0295	0	5	2	3,5384	Benign
ISIC_0025332	0,0244	0	5	1	3,0317	Benign
ISIC_0025395	0,0527	0	4	2	2,8185	Benign
ISIC_0025388	0,5228	8	4	3	4,7296	Benign
ISIC_0025393	0,0265	0	4	3	3,5345	Benign
ISIC_0025407	0,0844	7	4	2	3,5597	Benign
ISIC_0025400	0,5510	2	5	2	4,4163	Benign
ISIC_0025445	0,0424	2	4	2	3,2551	Benign
ISIC_0025517	0,0568	7	5	2	4,0239	Benign
ISIC_0025594	0,5117	2	5	3	4,8652	Suspicious
ISIC_0025670	0,5513	0	5	4	4,9667	Suspicious
ISIC_0025653	0,0621	5	5	1	3,5807	Benign
ISIC_0025680	0,0575	0	5	2	3,5748	Benign

Table 45. Results of the melanoma lesions.

NAME	ASYMMETRY	BORDER	COLOUR	DIAMETER	TDV	RESULT
AUGmented_0_2649	1,0465	3	5	4	5,9304	Malignant
AUGmented_0_2071	1,0684	8	4	5	6,4390	Malignant
AUGmented_0_602	0,8503	8	4	4	5,9054	Malignant
ISIC_0027552	1,0335	3	4	4	5,6636	Malignant
ISIC_0033670	1,0456	8	4	5	6,6592	Malignant
AUGmented_0_3990	0,5263	8	4	5	5,7342	Malignant
ISIC_0028965	0,5098	4	4	5	5,5627	Malignant
AUGmented_0_9073	1,0272	5	4	4	5,5853	Malignant
ISIC_0024732	0,5606	8	4	4	5,2788	Suspicious
ISIC_0029740	1,0374	3	4	5	5,8653	Malignant
AUGmented_0_418	1,0432	8	4	5	6,6562	Malignant
AUGmented_0_1610	1,1903	3	4	3	5,3675	Suspicious
AUGmented_0_9139	1,0188	8	4	5	6,3744	Malignant
AUGmented_0_2471	1,0639	6	4	3	5,4830	Malignant
AUGmented_0_1962	1,3317	8	4	5	7,0312	Malignant
AUGmented_0_3236	1,0673	7	4	3	5,5875	Malignant
ISIC_0027964	1,1317	8	4	4	6,0212	Malignant
AUGmented_0_9816	1,0765	0	4	5	5,8995	Malignant
ISIC_0030824	1,0374	8	4	4	5,8986	Malignant
ISIC_0034316	1,0913	8	4	4	6,2187	Malignant
ISIC_0024459	1,1072	5	4	5	6,2227	Malignant
ISIC_0033834	1,0291	8	4	5	6,6378	Malignant
ISIC_0033819	1,0500	8	4	4	6,1650	Malignant
ISIC_0033562	1,0345	7	4	5	6,2615	Malignant
ISIC_0033526	0,5307	8	4	4	5,4899	Malignant
ISIC_0033324	1,1061	5	4	3	5,4379	Suspicious
ISIC_0033037	1,0164	4	4	3	5,2657	Suspicious
ISIC_0032372	1,1426	8	4	3	5,7854	Malignant
ISIC_0031550	1,0408	6	4	3	5,4930	Malignant
AUGmented_0_7568	0,5603	6	4	5	5,5784	Malignant

To analyse the results, the sensitivity, specificity, and the accuracy will be measure:

Formula[17]. Calculation of the Sensitivity, Specificity and Accuracy

$$\text{sensitivity (\%)} = \frac{\text{true detected melanoma cases}}{\text{all melanoma cases}} * 100$$

$$\text{specificity (\%)} = \frac{\text{true detected non – melanoma cases}}{\text{all non melanoma cases}} * 100$$

$$\text{accuracy (\%)} = \frac{\text{true detected cases}}{\text{all cases}} * 100$$

Sensitivity = 86.6%

Specificity = 93.3%

Accuracy = 90.0%





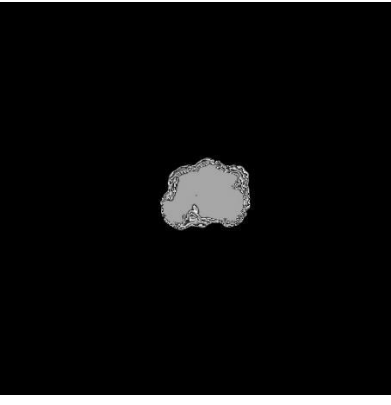


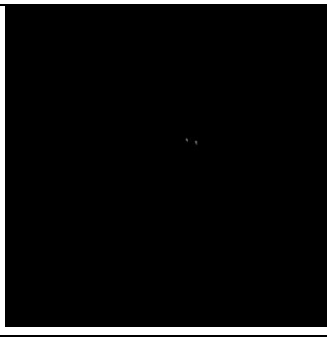


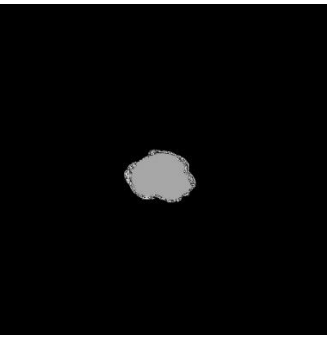

The final accuracy is 90% but have to be in account that there is not any lesion which is detected wrongly, just in case of the benign lesion two and in case of the melanoma lesions four are detected as a suspicious.

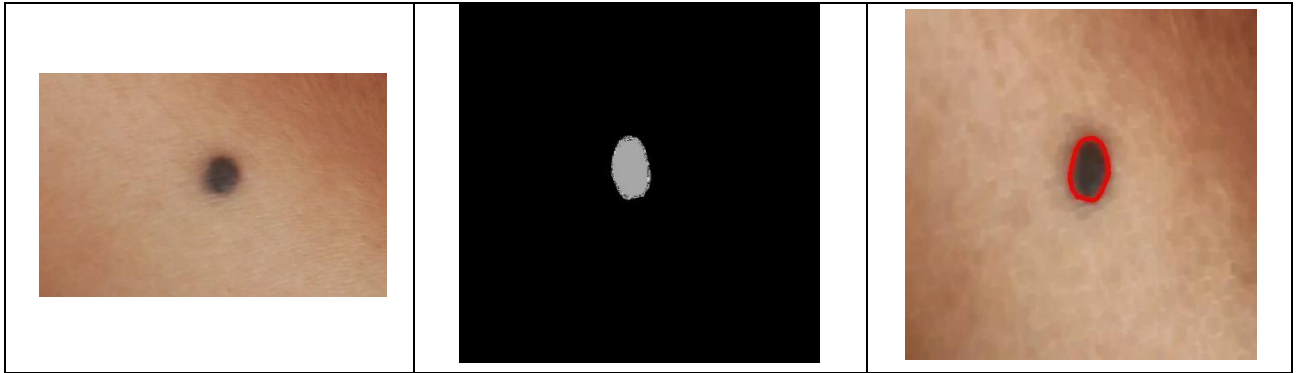
As a mentioned before, all the images used were dermoscopy images, but now, the program will be tested using images taken with a phone.

There were 12 images tested and 11 were good detected as a benign and the other image was not detected because the segmentation process did not work, as can be seen in the Table 46, third row. In the Table 46, some of the examples can be observed. In the first column, all the original images are shown. In the next, the masks obtained using the segmentation process and in the last one the segmented images are located.

As can be observed, the program is working for almost all the images and the one which is not working is because of the segmentation, probably because the pre-processing step is not able to remove all the hairs. But is important the fact that even if the quality of the images is not such a good quality as the one of the dermoscopy images, the program is working. So, it concludes that the implementation of a real application using it is possible in future investigations.

Table 46. Results of the images taken with a phone.

Original Images	Masks	Segmented Images
		
		
		
		



As future work I would like to try to improve my method to do it more robust in order to improve the accuracy and in an investigation in the future I would like to create a web page or a mobile app to provide with this useful tool to anyone who need it.

Conclusions and Results

- 1.** The principal goal of the project is achieved due to with an input of a dermoscopy lesion the program is able, after different steps, to sort out the image into benign lesion, melanoma lesion or suspicious lesion.
- 2.** After read and study several articles about how the melanoma classification has been done by other authors, the decision of use the ABCD rule was taken to do it has been used in many previous investigations with better results than the ones which used CNN.
- 3.** The research about the topic, referred in the chapter: “3. General Theoretical Concepts”, was indispensable for the understanding of all the mathematical and more theoretical concepts used in the method development.
- 4.** The used of Colab has helped to improve the speed of the program, reducing the times needed to train the U-Net model.
- 5.** The method developed shows 90% accuracy in the predicting test. Although it is not the best accuracy score obtained, comparing with other investigations, results are promising for further investigation.
- 6.** According to the main goal of the project, the development of melanoma classification tool to improve the early detection, a future investigation will allow the creation of the web page or phone application using the method developed.

List of references

- [1] Zghal, N. S., & Derbel, N. (2020). Melanoma Skin Cancer Detection based on Image Processing. *Current medical imaging reviews*, 16(1), 50–58. <https://doi.org/10.2174/1573405614666180911120546>
- [2] Leiter, U., Eigentler, T., & Garbe, C. (2014). Epidemiology of skin cancer. *Advances in experimental medicine and biology*, 810, 120–140. https://doi.org/10.1007/978-1-4939-0437-2_7
- [3] Abbas, Q., Emre Celebi, M., Garcia, I. F., & Ahmad, W. (2013). Melanoma recognition framework based on expert definition of ABCD for dermoscopic images. *Skin research and technology: official journal of International Society for Bioengineering and the Skin (ISBS) [and] International Society for Digital Imaging of Skin (ISDIS) [and] International Society for Skin Imaging (ISSI)*, 19(1), e93–e102. <https://doi.org/10.1111/j.1600-0846.2012.00614.x>
- [4] European Comision. (s. f.). *ECIS - European Cancer Information System*. ECIS. [https://ecis.jrc.ec.europa.eu/explorer.php?\\$0-1\\$1-LT,ES\\$2-130,183\\$4-1,2\\$3-27\\$6-0,85\\$5-1990,2012\\$7-2\\$CAgeSpecificRates\\$X0_14-\\$X0_12-\\$X0_13-Y\\$X0_16-N\\$CTrendsByAge\\$X1_14-Y\\$X1_12-\\$X1_18-6\\$X1_16-N\\$CTrendsByPeriod\\$X2_14-Y\\$X2_12-\\$X2_10-ASR_EU_NEW\\$X2_16-N\\$CTrendsByCohort\\$X3_17-ByPeriod\\$X3_16-N](https://ecis.jrc.ec.europa.eu/explorer.php?$0-1$1-LT,ES$2-130,183$4-1,2$3-27$6-0,85$5-1990,2012$7-2$CAgeSpecificRates$X0_14-$X0_12-$X0_13-Y$X0_16-N$CTrendsByAge$X1_14-Y$X1_12-$X1_18-6$X1_16-N$CTrendsByPeriod$X2_14-Y$X2_12-$X2_10-ASR_EU_NEW$X2_16-N$CTrendsByCohort$X3_17-ByPeriod$X3_16-N)
- [5] Guy, G. P., Jr, Thomas, C. C., Thompson, T., Watson, M., Massetti, G. M., Richardson, L. C., & Centers for Disease Control and Prevention (CDC) (2015). Vital signs: melanoma incidence and mortality trends and projections - United States, 1982-2030. *MMWR. Morbidity and mortality weekly report*, 64(21), 591–596.
- [6] World Health Organization. (s. f.). Cancer Today. International Agency for Research on Cancer. <https://gco.iarc.fr/today/home>
- [7] Markovic, S. N., Erickson, L. A., Rao, R. D., Weenig, R. H., Pockaj, B. A., Bardia, A., Vachon, C. M., Schild, S. E., McWilliams, R. R., Hand, J. L., Laman, S. D., Kottschade, L. A., Maples, W. J., Pittelkow, M. R., Pulido, J. S., Cameron, J. D., Creagan, E. T., & Melanoma Study Group of the Mayo Clinic Cancer Center (2007). Malignant melanoma in the 21st century, part 1: epidemiology, risk factors, screening, prevention, and diagnosis. *Mayo Clinic proceedings*, 82(3), 364–380. <https://doi.org/10.4065/82.3.364>
- [8] A. Rosenfeld, "Computer vision: basic principles," in Proceedings of the IEEE, vol. 76, no. 8, pp. 863-868, Aug. 1988, doi: 10.1109/5.5961. Abstract: The author provides a general introduction to computer vision. He discusses basic techniques and computer implementations, and also indicates areas in which further research is needed. He focuses on two-dimensional object recognition, i.e. recognition of an object whose spatial orientation, relative to the viewing direction is known. URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5961&isnumber=291>
- [9] Alegre, E. A., Pajares, G. P., & Escalera, A. E. (Eds.). (2016). Conceptos y métodos en visión por computador. Grupo de Vision del Comité Español de Automática (CEA). <http://intranet.ceautomatica.es/sites/default/files/upload/8/files/ConceptosyMetodosenVxC.pdf>

- [10] Levin, G. L., & Dorsey, B. D. ofBook - Image Processing and Computer Vision. Openframeworks. https://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html
- [11] GeeksforGeeks. (2018, 27 June). MATLAB | RGB image representation. <https://www.geeksforgeeks.org/matlab-rgb-image-representation/>
- [12] Palus H. (1998) Representations of colour images in different colour spaces. In: Sangwine S.J., Horne R.E.N. (eds) The Colour Image Processing Handbook. Springer, Boston, MA. https://doi.org/10.1007/978-1-4615-5779-1_4
- [13] Bankman, I. N. (2000). Handbook of Medical Imaging. En 1 - Fundamental Enhancement Techniques (pp. 3–18). Amsterdam University Press. <https://doi.org/10.1016/B978-012077790-7/50004-7>.
- [14] Ihab S. Mohamed (2017). Detection and Tracking of Pallets using a Laser Rangefinder and Machine Learning Techniques - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/An-example-of-convolution-operation-in-2D-2_fig3_324165524
- [15] Goyal, M. (2011). Morphological image processing. IJCST, 2(4). <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.219.4602&rep=rep1&type=pdf>
- [16] Open Source Computer Vision. (2021). OpenCV: OpenCV modules. <https://docs.opencv.org/master/>
- [17] Raid, A., Khedr, W., El-dosuky, M., & Aoud, M. (2014). Image Restoration Based on Morphological Operations. International Journal of Computer Science, Engineering and Information Technology, 4(3), 9–21. <https://doi.org/10.5121/ijcseit.2014.4302>
- [18] The SciPy community. (2021). NumPy Reference — NumPy v1.20 Manual. NumPy. <https://numpy.org/doc/stable/reference/index.html>
- [19] S. M. Pizer, R. E. Johnston, J. P. Ericksen, B. C. Yankaskas and K. E. Muller, "Contrast-limited adaptive histogram equalization: speed and effectiveness," [1990] Proceedings of the First Conference on Visualization in Biomedical Computing, 1990, pp. 337-345, doi: <https://doi.org/10.1109/VBC.1990.109340>
- [20] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, 234–241. doi: https://doi.org/10.1007/978-3-319-24574-4_28
- [21] Alonso, R. (2020, 8 abril). IA, Machine Learning y Deep Learning, ¿cuál es la diferencia? HardZone. <https://hardzone.es/tutoriales/rendimiento/diferencias-ia-deep-machine-learning/>

- [22] Rotemberg, V., Kurtansky, N., Betz-Stablein, B., Caffery, L., Chousakos, E., Codella, N., Combalia, M., Dusza, S., Guitera, P., Gutman, D., Halpern, A., Helba, B., Kittler, H., Kose, K., Langer, S., Lioprys, K., Malvehy, J., Musthaq, S., Nanda, J., Reiter, O., Shih, G., Stratigos, A., Tschandl, P., Weber, J. & Soyer, P. A patient-centric dataset of images and metadata for identifying melanomas using clinical context. *Sci Data* 8, 34 (2021). <https://doi.org/10.1038/s41597-021-00815-z>
- [23] Gallant, S. I., & Gallant, S. I. (1993). *Neural network learning and expert systems*. MIT press.
- [24] Wang, S.-C. (2003). Artificial Neural Network. *Interdisciplinary Computing in Java Programming*, 81–100. doi: https://doi.org/10.1007/978-1-4615-0377-4_5
- [25] Andrej Krenker, Janez Bešter and Andrej Kos (April 11th 2011). Introduction to the Artificial Neural Networks, *Artificial Neural Networks - Methodological Advances and Biomedical Applications*, Kenji Suzuki, IntechOpen, DOI: 10.5772/15751. Available from: <https://www.intechopen.com/books/artificial-neural-networks-methodological-advances-and-biomedical-applications/introduction-to-the-artificial-neural-networks>
- [26] Tch, A. (2021, 16 february). The mostly complete chart of Neural Networks, explained. Medium. <https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464>
- [27] O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458. Available from: <https://arxiv.org/pdf/1511.08458.pdf>
- [28] Reconstruction of porous media from extremely limited information using conditional generative adversarial networks - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Commonly-used-activation-functions-a-Sigmoid-b-Tanh-c-ReLU-and-d-LReLU_fig3_335845675
- [29] Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. 2017 International Conference on Engineering and Technology (ICET). doi: <https://doi.org/10.1109/icengtechnol.2017.8308186>
- [30] E. Nasr-Esfahani et al., "Melanoma detection by analysis of clinical images using convolutional neural network," 2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2016, pp. 1373-1376, doi: <https://doi.org/10.1109/EMBC.2016.7590963>
- [31] Yu C, Yang S, Kim W, Jung J, Chung KY, et al. (2018) Correction: Acral melanoma detection using a convolutional neural network for dermoscopy images. *PLOS ONE* 13(4): e0196621. <https://doi.org/10.1371/journal.pone.0196621>
- [32] Li, Katherine. M., & Li, Evelyn. C. (2018). *Skin Lesion Analysis Towards Melanoma Detection via End-to-end Deep Learning of Convolutional Neural Networks*. Published. <https://arxiv.org/pdf/1807.08332.pdf>

- [33] Abbas, Q., Emre Celebi, M., Garcia, I. F., & Ahmad, W. (2013). Melanoma recognition framework based on expert definition of ABCD for dermoscopic images. Skin research and technology : official journal of International Society for Bioengineering and the Skin (ISBS) [and] International Society for Digital Imaging of Skin (ISDIS) [and] International Society for Skin Imaging (ISSI), 19(1), e93–e102. <https://doi.org/10.1111/j.1600-0846.2012.00614.x>
- [34] Kasmi, R., & Mokrani, K. (2016). Classification of malignant melanoma and benign skin lesions: implementation of automatic ABCD rule. IET Image Processing, 10(6), 448–455. <https://doi.org/10.1049/iet-ipr.2015.0385>
- [35] The Python Tutorial — Python 3.9.5 documentation. (s. f.). Python. <https://docs.python.org/3/tutorial/index.html>
- [36] What is NumPy? — NumPy v1.20 Manual. (s. f.). NumPy. <https://numpy.org/doc/stable/user/whatisnumpy.html>
- [37] OpenCV. (2020, 4 November). About. <https://opencv.org/about/>
- [38] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.
- [39] math — Mathematical functions — Python 3.9.5 documentation. (s. f.). Python. <https://docs.python.org/3/library/math.html>
- [40] TensorFlow. (s. f.). TensorFlow. <https://www.tensorflow.org/>
- [41] Team, K. (s. f.). Keras: The Python deep learning API. Keras. <https://keras.io/>
- [42] Neshovski, R., D., D., & D. (s. f.). Home. United Nations Sustainable Development Goals. <https://www.un.org/sustainabledevelopment/>