Pigment Network Detection in Dermoscopy Images using Deep Learning

Pedro Filipe Coimbra de Sousa

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Dr. Ana Catarina Fidalgo Barata
Prof. Jorge dos Santos Salvador Marques

Examination Committee

Chairperson: Prof.
Advisor: Dr. Ana Catarina Fidalgo Barata
Members of the Committee: Prof. Maria Margarida Campos da Silveira

October 2018

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First and foremost, I would like to thank my supervisors Prof. Jorge Salvador Marques and Dr. Ana Catarina Barata for their guidance, constant support, availability and constructive suggestions, which were determinant for the accomplishment of the work presented in this thesis. I was extremely lucky to have supervisors who cared so much about my work and who answered my questions and doubts so promptly.

I would also like to thank to Eduardo Félix, Francisco Lourenço, Samuel Arcadinho, Miguel Silva, Pedro Pereira, Rui Santos and Loïc Foulon for the friendship, companionship, for providing such a pleasurable working atmosphere and for all the stories that I will always remember throughout these 5 years of Instituto Superior Técnico.

To my closest friends, I express my gratitude for their unconditional friendship, support and enjoyable moments throughout these years.

Above all, I would like to thank my family for their unconditional support and love throughout my life, especially my parents, who have always encouraged and supported me, providing me all the conditions to study and complete my education and my brother, who was always there for me.

Last but not least, I would like to address a special thanks to Inês, for her continuous support, encouragement, patience, and affection that she showed during my most stressed moments and for being by my side at all times.

This work was partially supported by FCT under project TDC/EEIPRO/0426/2014.

Abstract

Melanoma is considered to be the most dangerous form of skin cancer. However, if the melanoma is diagnosed in its early stages, it can be easily cured. Some medical techniques have been proposed to improve the performance of early melanoma diagnosis, including dermoscopy, which combines illumination techniques with magnification to obtain a better visualization of the skin lesion. Based on this technique, several medical procedures, such as the ABCD rule and the 7-point checklist, were developed to simplify not only the distinction between the different types of lesions but also the detection of malignant melanomas. These procedures rely on the detection of dermoscopic features and colors in dermoscopy images of the lesion. One of the most relevant dermoscopic structures detected by these procedures is pigment network. Some works were published addressing the automatic detection of this structure, but the majority of them only focus on detecting it and not localizing it, which would be of great importance for medical experts. Thus, this work proposes a system for the automatic detection and segmentation of pigment network, using a deep learning approach. The developed system was based on a well-known convolutional neural network (CNN) architecture called U-Net, which was designed for biomedical image segmentation tasks. This system receives as input a dermoscopy image and generates a binary mask where the presence of pigment network is highlighted. This method was tested against a dataset of 600 images belonging to the ISIC database, achieving a sensitivity of 90.5% and a specificity of 88.9%, which proves the reliability of the proposed system.

Keywords: Medical Image Analysis, Dermoscopy, Pigment Network Detection, Deep Learning, Convolutional Neural Networks, Image Segmentation.

Resumo

O melanoma é considerado umas das formas mais perigosas de cancro de pele. Contudo, se o melanoma for diagnosticado precocemente pode ser facilmente curado. Algumas técnicas de aquisição de imagem médica foram propostas de modo a melhorar o desempenho do diagnóstico precoce de melanomas, incluindo a dermoscopia, que combina técnicas de iluminação com magnificação de modo a obter uma melhor visualização da lesão cutânea. Vários procedimentos de diagnóstico médico baseados na dermoscopia, como a regra ABCD e a lista de verificação dos 7 pontos, foram desenvolvidos de modo a facilitar não só a distinção entre os diferentes tipos de lesão, mas também a deteção de melanomas malignos. Estes procedimentos baseiam-se na deteção de estruturas dermoscópicas e cores em imagens de dermoscopia da lesão. Uma das estruturas dermoscópicas mais relevantes detectada recorrendo a estes procedimentos é a rede pigmentar. Alguns trabalhos publicados abordam a deteção automática desta estrutura, mas a grande maioria foca-se apenas na deteção e não na localização, o que ajudaria bastante os dermatologistas. Assim, este trabalho propõe um sistema para a detecção automática e segmentação de rede pigmentar, recorrendo a uma abordagem de aprendizagem profunda. O sistema desenvolvido é baseado numa conhecida arquitetura de redes neuronais convolucionais chamada U-Net, que foi projetada para a segmentação de imagens biomédicas. O sistema recebe como entrada uma imagem dermoscópica e gera uma máscara binária onde a presença de rede pigmentar é destacada. Este método foi testado usando um conjunto de 600 imagens pertencentes à base de dados ISIC, obtendo uma sensibilidade de 90.5% e uma especificidade de 88.9%, o que comprova a confiabilidade do sistema proposto.

Palavras-Chave: Análise de Imagens Médicas, Dermoscopia, Detecção de Rede Pigmentar, Aprendizagem Profunda, Redes Neuronais Convolucionais, Segmentação de Imagens.

Contents

List of Tables xi

List of Figures xiii

Acronyms xvii

1 Introduction 1

1.1 Context and Motivation . . . 1

1.2 Objectives and Thesis Structure . . . 2

1.3 Contributions . . . 2

2 Dermoscopy Image Analysis 3

2.1 Types of Skin Lesions . . . 3

2.2 Dermoscopy . . . 5

2.3 Medical Diagnosis . . . 5

2.3.1 ABCD rule . . . 6

2.3.2 7-point checklist . . . 7

2.4 Pigment Network . . . 8

2.5 Detection of Pigment Network . . . 9

3 Deep Learning 13

3.1 Early years . . . 13

3.2 Recent tendencies . . . 14

3.3 Background . . . 14

3.3.1 Computational model of neuron . . . 15

3.3.2 Network Architecture . . . 16

3.3.3 Loss Function . . . 17

3.3.4 Training process . . . 17

3.4 Convolutional Neural Networks . . . 18

3.4.1 Motivation . . . 18

3.4.2 Convolutional Layers . . . 20

3.4.3 Pooling . . . 22

3.4.4 Fully connected layers . . . 23

3.4.5 Deconvolution . . . 24

3.5 CNN architectures for image segmentation . . . 26

4 Proposed Methodology **29**
4.1 Overall architecture 29
4.2 Pre-Processing 29
4.3 CNN Architecture 30
4.4 Training method of the network 30
4.4.1 Definition of hyperparameters 31
4.4.2 Model Evaluation 33

5 Implementation and Results **35**
5.1 Dataset 35
5.2 Performance metrics 36
5.3 Implementation aspects 37
5.4 System optimization 37
5.5 Results 39

6 Conclusion **43**

Bibliography **45**

List of Tables

2.1 Melanocytic algorithm [1] 6
2.2 ABCD rule of dermoscopy [5] 7
2.3 Criteria and scores of the 7-point checklist method (adapted from [6]). 8

3.1 Summary of winning networks in the ImageNet classification challenge (adapted from [44] [45]) 14

5.1 ISIC 2017 Dataset [20] distribution 36
5.2 Confusion matrix 36
5.3 System performance in the validation set 39
5.4 System performance in the test set 39
5.5 Results over training for Experiment 4 40

List of Figures

2.1 Skin lesions classification tree (adapted from [1]). 3
2.2 Examples of melanocytic (1st row) and non-melanocytic (2nd row) lesions [1]. 4
2.3 Melanoma mimicking clark nevus [1]. 5
2.4 Examples of the ABCD rule [1]. 7
2.5 Examples of the 7-point checklist [1]. 8
2.6 Examples of pigment network [1]. 9

3.1 (a) Biological neuron scheme; (b) Artificial neuron scheme (adapted from [50]). 15
3.2 Non-linear activation functions used in neural networks. (a) Sigmoid; (b) TanH; (c) ReLU. 16
3.3 Example of a MLP architecture (adapted from [52]); the activation function is denoted by σ. 17
3.4 Example of a CNN architecture (adapted from [57]), consisting of convolutional, pooling and fully connected layers. 19
3.5 Example of an architecture for semantic image segmentation (adapted from [59]); the number below the boxes represent the number of filters of each layer. 19
3.6 Example of CNN architecture for semantic image segmentation (adapted from [60]). . . . 20
3.7 (a) Input image; (b) Activation maps obtained after convolution with 7 different filters (adapted from [62]). 21
3.8 Convolution between a filter (kernel) and a multi-channel input image (adapted from [63]). 21
3.9 Pictorial representation of ReLU functionality (adapted from [61]). 22
3.10 Features visualization example (adapted from [65]). 23
3.11 Pictorial representation of max-pooling functionality (adapted from [50]). 23
3.12 Example of a CNN architecture for a handwritten digit recognition task (adapted from [66]). 24
3.13 The transpose of convolving a 3×3 kernel over a 5×5 input using 2×2 strides (i.e., N = 5, K = 3, S = 2 and P = 0) (adapted from [67]). The input is represented in green, while the output is represented in blue. 24
3.14 (Dilated convolution) Convolving a 3×3 kernel over a 7×7 input with a dilation factor of 2 (i.e., N = 7, K = 3, d = 2, S = 1 and P = 0) (adapted from [67]). 25
3.15 Example of the application of the unpooling operation (adapted from [70]). 25
3.16 FCN architecture (adapted from [59]). 26
3.17 U-Net architecture (adapted from [77]). 27

4.1 Block diagram of the system's architecture. 29
4.2 Training methodology. The small loop corresponds to each mini-batch training (full line). The bigger loop is performed once per epoch (loop including dashed line), after all mini-batches have been feedforwarded once in the current epoch. 31
4.3 Model evaluation procedure. 33

5.1 (a) Original image with pigment network; (b) Binary segmentation mask (extracted from the ISIC 2017 dataset [20]). 35
5.2 Balanced error over epochs for all the tested configurations using validation data. 38
5.3 Satisfactory segmentation results: original image (left), automatic segmentation (center) and groundtruth segmentation (right). 41
5.4 Poor segmentation results: original image (left), automatic segmentation (center) and groundtruth segmentation (right). 42

Acronyms

ACC Accuracy. 36

AI Artificial Intelligence. 14

ANN Artificial Neural Network. 13, 15–17, 19, 22, 23

BA Balanced Accuracy. 36

BCC Basal Cell Carcinoma. 4, 5

BE Balanced Error. 33

CE Cross-Entropy. 32, 39, 40

CNN Convolutional Neural Network. 13, 14, 18–20, 22–26, 29–31, 33, 43, 44, 48

CPU Central Processing Unit. 14, 37

DNN Deep Neural Network. 13

ELM Epiluminescence Microscopy. 5

FCN Fully Convolutional Network. 20, 26

FFT Fast Fourier Transform. 10

FN False Negatives. 36

FP False Positives. 36, 40

GPU Graphics Processing Unit. 14, 37

ILCA Iterative Loop Counting Algorithm. 10

ILSVRC ImageNet Large-Scale Visual Recognition Challenge. 14

ISIC International Skin Imaging Collaboration. 35

IST Instituto Superior Técnico. 37

LDA Linear Discriminant Analysis. 10

LoG Laplacian of Gaussian. 10

MAP Maximum a posteriori Probability. 10

MLP Multi-Layer Perceptron. 16

MSE Mean Squared Error. 17

NGLDM Neighboring Gray-Level Dependence Matrix. 9

NN Neural Network. 13, 17–19

ReLU Rectified Linear Unit. 16, 22, 27

RNN Recurrent Neural Network. 14

SE Sensitivity. 36

SGD Stochastic Gradient Descent. 18

SP Specificity. 36

SVM Support Vector Machine. 14

TanH Hyperbolic Tangent. 16

TDS Total Dermoscopy Score. 6–8

TN True Negatives. 36

TP True Positives. 36, 40

WCE Weighted Cross-Entropy. 32, 39, 40, 43

Chapter 1

Introduction

This chapter intends to emphasize the importance of detecting pigment network when diagnosing a skin lesion as melanoma or not. It also details the objectives of this dissertation as well as its structure. At the end of this chapter, the main contributions of this thesis are outlined.

1.1 Context and Motivation

Skin cancer is one of the most common forms of cancer and may take the form of a benign or malignant lesion. Melanoma is considered to be the deadliest of malignant skin cancers, mainly due to its ability to metastasize [1]. Although it comprises only 4% of all skin cancers, it is responsible for 80% of skin cancer-related deaths [2]. However, if the melanoma is diagnosed in its early stages, it is easily curable by performing a simple excision of the lesion [1]. Thus, it is crucial to develop reliable automatic systems for melanoma detection.

The dermoscopy technique was developed to improve the performance of early melanoma diagnosis, combining special illumination techniques with magnification to obtain a better visualization of the skin lesion [3]. Previous studies showed that comparing to naked-eye analysis, dermoscopy improves the melanoma diagnostic accuracy by 10-27%. However, this technique can only improve the diagnostic performance if the dermatologists are trained formally [4], otherwise, some lesions can be underdiagnosed, leading to dangerous misclassifications. Based on this technique, some medical procedures were developed to simplify the classification between the different types of skin lesions and detect malignant melanomas. These diagnostic algorithms, such as the ABCD rule [5] and the 7-point checklist [6], rely on the detection of dermoscopic features and colors that are observed in a dermoscopy image of the lesion. One of the most relevant dermoscopic structures analysed by these algorithms is the pigment network [1].

The presence of pigment network is accounted when distinguishing between a melanoma and other skin lesions, being moreover a feature present in all medical algorithms for the diagnosis of melanomas. Furthermore, when an atypical network is present, it commonly results in the lesion being classified as melanoma. Thus, it is essential to develop methods to detect it, since it has a great importance in melanoma diagnosis [7].

Despite its automatic detection being a very complex problem, several works have been published addressing the detection of this structure [8–14], however, only a few focus not only on detecting it but also localizing it in dermoscopy images, which is of major interest for the dermatologists when diagnosing a lesion. Thus, the aim of this thesis is to carry out the automatic detection and segmentation of pigment network using a deep learning approach.

The advent of deep learning has had a significant impact on many areas in machine learning [15], dramatically improving the state-of-the-art in different tasks such as object detection, speech recognition, language translation [16] and also dermoscopy image analysis [17]. So far, very few investigations have considered deep learning techniques towards the automatic detection of pigment network, with the exception of some successful works such as the one presented by Kawahara et. al [18]. This is highly related to the fact that deep learning architectures require a large amount of data and only recently were provided databases with enough examples, such as the DermNet [19] and ISIC [20] datasets, that would make this area benefit from the power of deep learning.

1.2 Objectives and Thesis Structure

The main objective of this thesis is to develop a system capable of automatically detecting pigment network in dermoscopy images. To achieve this goal, this thesis will follow a deep-learning approach, based on a well-known convolutional neural network architecture specifically designed for biomedical image segmentation problems. Moreover, this work attempts to overcome the issue of imbalanced data present in the used dataset.

This work is organized as follows: Chapter 2 presents a review of the state-of-the-art related to the analysis of dermoscopy images, introducing the dermoscopy technique as well as medical algorithms and methods for the detection of pigment network. Chapter 3 introduces deep learning background, addressing its evolution over the years, basic concepts about neural networks and convolutional neural networks. Additionally, this chapter also presents some well-known architectures for image segmentation. Chapter 4 describes the methods used in the development of the system for automatic detection of pigment network. Chapter 5 presents the experimental results and some implementation aspects, including a quantitative evaluation of the proposed system using a database of 600 images of the test set belonging to the ISIC dataset. Chapter 6 addresses the main conclusions and considerations towards future work.

1.3 Contributions

The following contributions are presented in this thesis:

- One of the first systems for the automatic detection of pigment network using a deep learning approach.
- A method to handle imbalanced data.

Chapter 2

Dermoscopy Image Analysis

This chapter addresses basic concepts about Dermoscopy Image Analysis, starting by introducing the types of skin lesions and medical diagnostic techniques. Then, a more detailed approach to pigment network and its detection is provided.

2.1 Types of Skin Lesions

There is a large variety of skin lesions, which can be evaluated in different ways. It is common to organize these lesions following a hierarchical structure, as illustrated in Figure 2.1. The first step of any dermoscopy analysis is the classification of the lesion as melanocytic or non-melanocytic [1]. This classification is based on the type of skin cells responsible for the lesion's origin. Melanocytic lesions, like melanoma, arise from the proliferation of melanocytes, which produce a protein pigment called melanin. Non-melanocytic lesions develop from other types of skin cells, namely squamous cells and basal cells. In clinical routine, the distinction between both types of lesion is performed by visually inspecting the presence of a set of dermoscopic features, such as the pigment network, which is considered one of the most important structures in dermoscopy [7]. The second step is to classify the lesion as benign or malignant. This classification is based on a set of dermoscopic structures and criteria. Figure 2.2 depicts some examples of both melanocytic and non-melanocytic lesions.

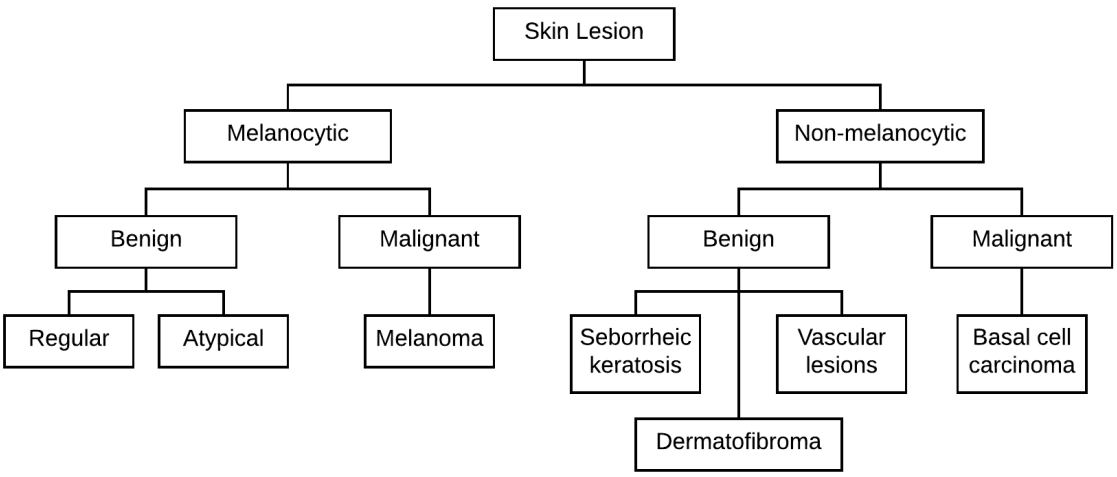
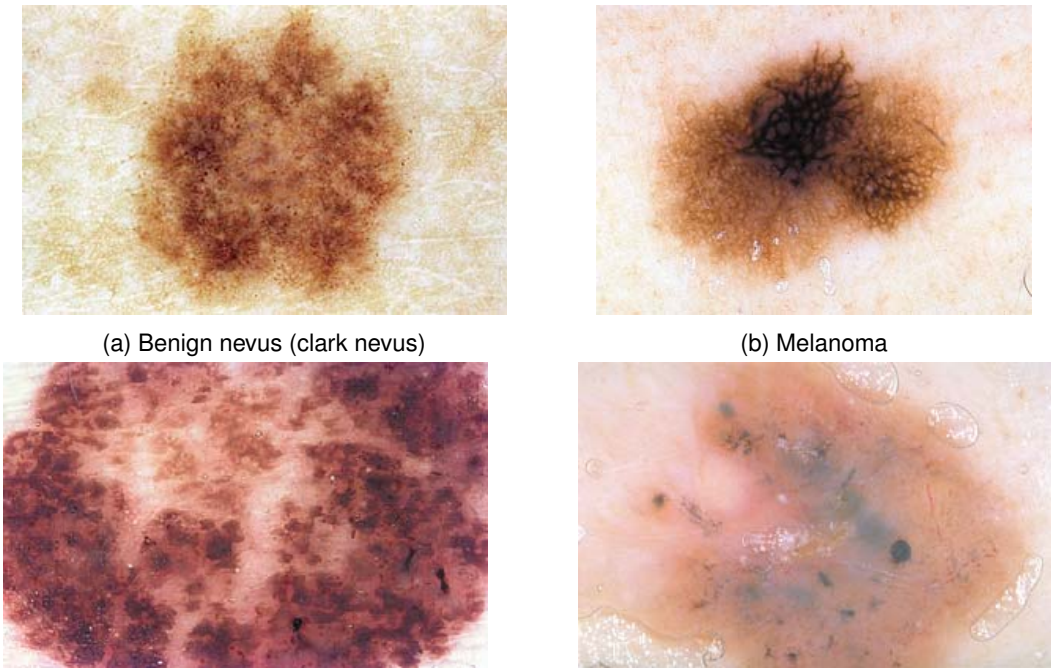


Figure 2.1: Skin lesions classification tree (adapted from [1]).



(a) Benign nevus (clark nevus)

(b) Melanoma

(c) Seborrheic keratosis

(d) BCC

Figure 2.2: Examples of melanocytic (1st row) and non-melanocytic (2nd row) lesions [1].

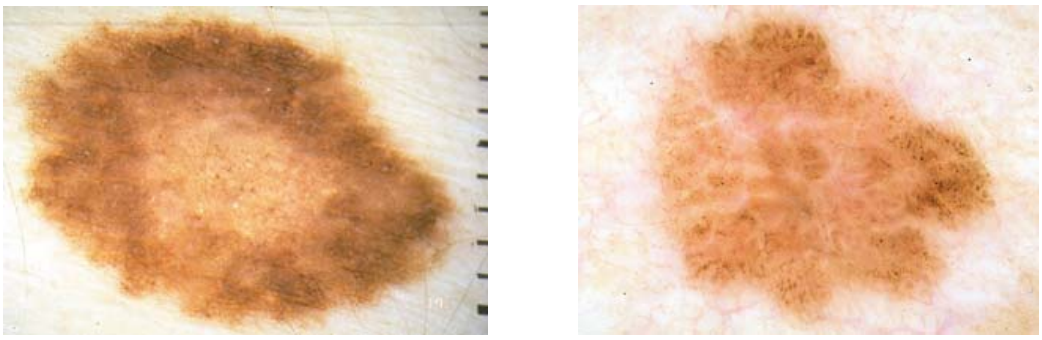
There are four main types of non-melanocytic lesions with each one of them having distinctive characteristics. As seen in Figure 2.1, three of the non-melanocytic lesions are benign neoplasms and their names are vascular lesions, seborrheic keratosis and dermatofibroma. The malignant neoplasm of the non-melanocytic lesions is the Basal Cell Carcinoma (BCC). This lesion is the most common type of skin cancer. Nonetheless, since its growth is exceedingly slow, most BCCs are innocuous. However, if appropriate treatment is not carried out, they can produce extensive tissue destruction, which may lead to death [1].

As stated above, the melanocytic lesions are also distinguished between the malignant (melanoma) and benign ones, which include regular and atypical lesions. Benign melanocytic lesions are usually called moles or nevi and can be classified as congenital when present at birth or acquired when developed during life. Another classification can be performed regarding the origin of the nevi within the skin. Nevi tend to begin as nests of nevus cells along the dermoepidermal junction (junctional nevus) and, over the time, nevus cells migrate downward into the dermis (intradermal nevus). When the nevus cells are present at both layers, it is called compound nevus. Depending on its origin, the lesion may present a specific coloration and dermoscopic structures [21]. The irregularly shaped melanocytic nevi are called atypical nevi (also, clark nevi) and those are considered an intermediate state between benign and malignant neoplasms. Clark nevi are the most common nevi, being regarded as the most relevant precursor of melanoma [1].

Melanoma is the malignant form of melanocytic lesions. Although it is less common than the BCC, melanoma grows faster and shows a high potential to metastasize to other organs, which makes it crucial to detect melanomas in an early stage. Melanoma *in situ* refers to this stage, where the neoplasm is still situated within the epidermis [1], not having contact with deeper layers of the skin. In this case, the melanoma has not yet metastasized and a simple excision of the neoplasm is enough to entirely remove it.

The distinction between melanomas and other melanocytic lesions is one of the major challenges of dermoscopy. Sometimes, a melanoma is wrongly diagnosed by dermatologists as being a different type of lesion due to the similarities that melanomas share with nevi. Figure 2.3 shows an example of a

melanoma mimicking a clark nevus.



(a) Clark nevus (b) Melanoma

Figure 2.3: Melanoma mimicking clark nevus [1].

The existence of this kind of errors constitutes the motivation to develop methods to help dermatologists to successfully diagnose melanomas.

2.2 Dermoscopy

Dermoscopy, also called Epiluminescence Microscopy (ELM) or dermatoscopy, is a non-invasive diagnostic technique that allows the observation of a variety of clinical patterns and features of pigmented skin lesions which are invisible to the naked-eye [1]. It is mainly used to distinguish malignant skin lesions, such as melanoma and pigmented BCC, from benign melanocytic nevi and seborrheic keratoses.

This diagnostic tool combines special illumination techniques with magnification, resulting in the visualization of an improved image of the skin lesion [3]. The first step of this technique consists in placing mineral oil or spraying alcohol or water on the surface of the skin lesion. This preparation stage allows to eliminate part of surface reflection and makes the lesion more transparent to light, avoiding the light to be reflected or dispersed [22]. This initial step is followed by the inspection of the lesion using a dermatoscope, a stereomicroscope, a camera or a digital imaging system. The dermatoscope is the most widely used as it combines a relatively high magnification with a rapid and simple use. More recently, the use of digital imaging systems has become popular as it allows easy storage, retrieval and follow-up of pigmented skin lesions [23]. Depending on the instrument used, the magnification of the lesion ranges from 6x to 40x and even up to 100x [1]. With this technique, dermatologists are able to analyse the region of interest of the skin lesion and its dermoscopic structures in detail, which enhances their possibilities to reach a successful diagnosis.

Previous studies showed that dermatologists only diagnose 65-80% of the melanomas by clinical naked-eye examination, highly depending on the dermatologist's expertise [4]. Comparing to naked-eye analysis, dermoscopy improves the diagnostic accuracy of melanoma detection by 10-27% [24]. Nevertheless, dermoscopy can only improve dermatologists' diagnostic performance if trained formally [1], as some lesions can be underdiagnosed leading to probably dangerous misclassifications.

2.3 Medical Diagnosis

Dermatologists use certain criteria based on dermoscopy image analysis to distinguish the different types of skin lesions and detect melanomas, starting by classifying the lesion as melanocytic or non-melanocytic and, afterwards, diagnose it as benign or malignant (see Figure 2.1), using diagnostic algorithms. These criteria are commonly related with the dermoscopic features and colors that are

observed in a dermoscopy image of the lesion. The dermoscopic features can be divided into global features and local features [1]. The global features are a set of patterns which are mostly distributed throughout the region of the pigmented skin lesion. This first group includes several different patterns, including reticular, globular, cobblestone, parallel, among others. The local features are located in specific regions of the lesion and are known as the letters of the dermoscopic alphabet since they allow a final detailed diagnosis of the lesion [1], i.e. melanoma or not. Some of the local patterns considered in dermoscopy analysis are pigment network, dots, globules, streaks and pigmentation.

The most classic dermoscopic approach for diagnosing pigmented skin lesions is called pattern analysis, which was proposed by Pehamberger et al. in 1987 [25]. This method depends on the description of a set of patterns (global features), where each pattern is composed of one or more dermoscopic structures (local features) that are present in the lesion. Pattern analysis can be used to identify and diagnose both melanocytic and non-melanocytic lesions. The final diagnosis of the lesion as benign or malignant relies on the number of dermoscopic structures present in the lesion and on the shape of the structures that compose the pattern.

Despite the increase in the accuracy rate of correctly diagnosed lesions by dermatologists using this method, there is subjectivity inherent to the assessment of the local dermoscopic structures and pattern characterization, which raises some problems related to the reliability and reproducibility of this process [23]. To solve this issue, new algorithms were developed, including several scoring systems such as the ABCD rule [5] and the 7-point checklist [6], which require a previous step to classify the pigmented lesion as melanocytic or non-melanocytic. For this initial classification is used the melanocytic algorithm, which is shown in table 2.1. Only after the diagnosis of the lesion as non-melanocytic is ruled out and a melanocytic lesion is diagnosed, can one of the aforementioned algorithms be applied.

Step	Diagnostic criteria	Diagnosis
I	Pigment network Brown to black dots/globules Streaks Homogeneous blue pigmentation Parallel pattern (on palms and soles)	Melanocytic lesion
II	Milia-like cysts Comedo-like openings (irregular crypts)	Seborrheic keratosis
III	Leaf-like areas Arborizing vessels Irregular gray-blue globules and blotches	Basal cell carcinoma
IV	Red lacunas Red-bluish to red-black homogeneous areas	Vascular lesion
V	Central white patch (surrounded by delicate pigment network)	Dermatofibroma
VI	None of the above criteria	Melanocytic lesion

Table 2.1: Melanocytic algorithm [1]

2.3.1 ABCD rule

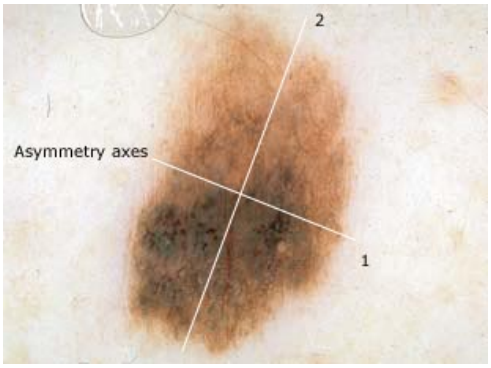
The ABCD rule of dermoscopy [5] is a semiquantitative method which allows to classify a skin lesion as melanoma or non-melanoma. As the name suggests, this procedure assesses four different characteristics of the skin lesion: Asymmetry (A), Border (B), Color (C) and Differential Structure (D). To each one of these characteristics is assigned a score based on specific criteria. Table 2.2 describes a summary of the criteria used to assign each score. Afterwards, these scores are weighted according to the values presented in the table 2.2, allowing the computation of a Total Dermoscopy Score (TDS) according to the equation (2.1).

Criterion	Description	Score	Weight Factor
Asymmetry	In 0, 1, or 2 axes; assess not only contour, but also colors and structures	0-2	1.3
Border	Abrupt ending of pigment pattern at the periphery in 0-8 segments	0-8	0.1
Color	Presence of up to six colors 1-6 (white, red, light-brown, dark-brown, blue-gray, black)	1-6	0.5
Differential Structures	Presence of network, structureless or homogeneous areas, streaks, dots, and globule	1-5	0.5

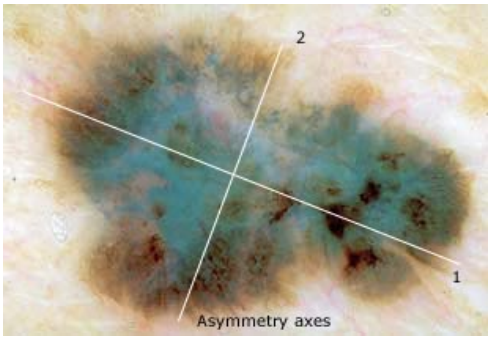
Table 2.2: ABCD rule of dermoscopy [5]

$$TDS = 1.3A_{score} + 0.1B_{score} + 0.5C_{score} + 0.5D_{score} \tag{2.1}$$

The value of the TDS determines the final diagnosis of the melanocytic lesion. A lesion is diagnosed as a melanoma if the TDS value is higher than 5.85. For TDS values lower than 4.75 the melanocytic lesion is diagnosed as benign. If the TDS value is between the two thresholds, the lesion is considered suspicious and a close follow-up is recommended. Figure 2.4 illustrates how the ABCD rule is applied to two different lesions, with the first one diagnosed as a benign melanocytic lesion and the second one diagnosed as a melanoma.



(a)
A = 1 (x 1.3) = 1.3;
B = 4 (x 0.1) = 0.4;
C = 3 [light/dark-brown, black] (x 0.5) = 1.5;
D = 2 [network, dots] (x 0.5) = 1;
TDS = 4.2 (Benign Lesion)



(b)
A = 2 (x 1.3) = 2.6;
B = 5 (x 0.1) = 0.5;
C = 4 [light/dark-brown, blue-gray, black, white] (x 0.5) = 2;
D = 4 [homogeneous areas, streaks, dots, globules] (x 0.5) = 2;
TDS = 7.1 (Melanoma)

Figure 2.4: Examples of the ABCD rule [1].

2.3.2 7-point checklist

The 7-point checklist [6] is an alternative method to the ABCD rule which is also widely used by the dermatologists to diagnose melanocytic skin lesions. This diagnostic procedure requires the identification of only 7 dermoscopic criteria, allowing less experienced clinicians to be able to successfully detect melanomas [1]. These dermoscopic structures can be divided into two groups: major and minor criteria. If any of the criteria is present in the lesion, it will receive a score, as shown in table 2.3. Afterwards, the individual scores assigned to each one of the criteria are summed up and it is computed a total score associated to the skin lesion. If this value is above 3, the lesion is diagnosed as melanoma. Thus, for a melanoma to be diagnosed, it is only required the identification of at least 2 melanoma-specific

dermoscopic criteria (1 major plus 1 minor or 3 minor criteria) [1]. Figure 2.5 exemplifies the application of the 7-point checklist method to two skin lesions.

Major criteria (2 points)	**Minor Criteria** (1 point)
Atypical pigment network	Irregular streaks
Blue-whitish veil	Irregular pigmentation
Atypical vascular pattern	Irregular dots/globules
	Regression structures

Table 2.3: Criteria and scores of the 7-point checklist method (adapted from [6]).

(a) Benign lesion (b) Melanoma

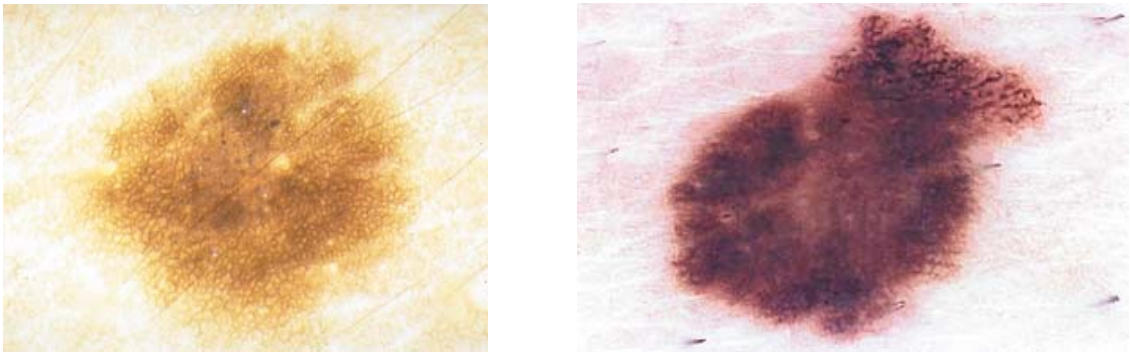
Figure 2.5: Examples of the 7-point checklist [1].

2.4 Pigment Network

As stated in Subsection 2.1, pigment network is one of the most important structures in dermoscopy. It is a grid-like network consisting of pigmented lines over a diffuse light-brown background, having a pattern very similar to a honeycomb [26]. The presence of this structure is accounted when diagnosing a lesion as melanocytic or non-melanocytic or in the identification of a melanoma [7], which makes it crucial to detect it in a lesion.

This dermoscopic structure represents the hallmark of melanocytic lesions although it can be found in other types of skins lesions. The assessment of its alterations in a lesion allows the differentiation between benign or malignant proliferation, especially when confined to the epidermis and superficial dermis [1]. Pigment network has its anatomic basis either in melanin pigment in keratinocytes, or in melanocytes along the dermoepidermal junction [27] and can be typical or atypical. A network is considered typical when it is light to dark-brown pigmented, regularly meshed, narrowly spaced and more or less distributed throughout the lesion, with its lines getting gradually thinner and fainter in pigmentation at its periphery. Oh the other hand, an atypical network is irregularly meshed with broader and darker pigmented lines, being distributed more or less irregularly throughout the lesion and usually ending abruptly at the lesion's periphery. Figure 2.6 exemplifies both typical and atypical networks.

As studied in Section 2.3, pigment network is assessed in several medical procedures. On the pattern analysis procedure, the reticular pattern, the most common global feature in melanocytic lesions, is characterized by pigment network covering almost the entire lesion [25]. In the ABCD rule, the presence of pigment network is accounted for the differential structures criterion when computing the TDS [5]. Finally, in the 7-point checklist procedure, pigment network and its shape are considered. If the network present in a given lesion is atypical, it receives one of the major scores [6].



(a) Typical network (b) Atypical network

Figure 2.6: Examples of pigment network [1].

Thus, the distribution of melanocytes on the epidermis makes from this dermoscopic feature a key structure when diagnosing a melanoma. For this reason, it is essential to develop methods to detect pigment network, which can help the automatic classification of a given lesion as melanocytic or non-melanocytic, as well as the shape of the network. As seen, the presence of atypical network commonly results in the lesion being identified as a melanoma.

2.5 Detection of Pigment Network

Pigment network is one of the key structures in dermoscopy image analysis. However, its automatic detection is a very complex problem for different reasons. In some lesions, there is a low contrast between the network and the background. Furthermore, the size of the network holes may be considerably different for some images, and even in the same one could exist big irregularities in shape and size [14]. This is the reason why the detection of pigment network was ignored for a long time. Nonetheless, this problem has been investigated more recently, leading to the publication of several works.

In [8], Fleming et al. proposed one of the first methods to detect pigment network. Their work consisted of extracting and measuring the pigment network characteristics. They investigated the thickness and the variability of the thickness of network lines and the size and variability of network holes. Their algorithm applies the first and second derivative to the dermoscopy images to detect pixels that belong to the curvilinear lines. The lines of the pigment network are considered ridges and a pixel is considered active if the first derivative is close to zero and the second derivative has a high absolute value. The final network is obtained by linking the pixels, since the second derivative provides information about the orientation and proximity between pixels. However, their results were purely qualitative since there was no outcome concerning the behaviour of the system in the differentiation between *pigment network* and *no pigment network*. Grana et al. [28] proposed a similar method for pigment network detection. However, they used a different approach to link the pixels, using a set of morphological masks, instead of applying the second derivative.

Anantha et al. [9] proposed and compared two different texture analysis algorithms for pigment network detection. The first method uses filtering with Laws' energy masks and the second one involves statistics over Neighboring Gray-Level Dependence Matrix (NGLDM), with the first one obtaining better results. The system was tested over a total number of 155 images, achieving an accuracy of correctly classified images of 80%.

Betta et al. [10] conducted the detection of atypical pigment network by combining a structural technique with a spectral one. The structural technique is used to detect simple shape structures such as lines and points. This procedure starts by taking the difference of an image and its response to the

median filter. Then, this difference image is thresholded to create a binary mask which undergoes a morphological closing operation to remove potential isolated points. The spectral technique creates a mask using a sequence of the Fast Fourier Transform (FFT), high-pass filters, inverse FFT and thresholding to exclude any slowly modulating frequencies. In the end, both masks are combined to provide one network image where the holes of pigment network are highlighted. This method achieved a sensitivity of 50% and specificity of 100% in a dataset of 30 images. Di Leo et al. [29], from the same research group, improved on the previous study. They include a stage where the areas of the final network image are classified as atypical or non-atypical (the absence of network and the presence of typical network belong to the same class), relying on color and spatial features to characterize the holes of the pigment network. They achieve results greater than 85% for both specificity and sensitivity (no exact values are set).

In [30], Shrestha et al. use 10 different texture measures for the analysis of atypical pigment network. This procedure achieved an accuracy of 95.4% for the detection of atypical pigment network, but it does not deal with the problem of differentiation between *pigment network* and *no pigment network*.

Sadeghi et al. [11] carry out the detection of pigment network structures in dermoscopy images using in the first place, the Laplacian of Gaussian (LoG) filter to properly capture sudden transitions in intensity, allowing to enhance meshes or round shaped structures which are characteristics of a pigment network region. Then, the resulting filtered image is converted to a graph, using eight-connected components analysis. Afterwards, cyclic subgraphs are searched using the Iterative Loop Counting Algorithm (ILCA) to find cyclic texture features that may indicate the presence of pigment network. Finally, the subgraphs are filtered and noise or wrongly detected structures, such as globules and dots, are removed and a graph of pigment network is obtained. The resulting image is then classified according to the density ratio of the graph. They reported an accuracy of 94.3%. In [31], the same research group improved their algorithm and extended the previous work, presenting a new method for classification between *absent*, *typical* and *atypical*. To do so, they propose an algorithm based on the previous one, which detects the net structure and extracts structural, geometric, chromatic and texture features, generating the classification model with the LogitBoost algorithm. They obtain an accuracy of 82% discriminating between the three aforementioned classes and an accuracy of 93% discriminating between two classes: *absent* or *present*.

Wighton et al. [12] propose an algorithm for the detection of pigment network based on supervised machine learning techniques. Their method was divided into three main steps. First, in a feature extraction stage, they used colour features and spectral features, filtering each color channel with a series of Gaussian and Laplacian of Gaussian filters at various scales. Secondly, they used Linear Discriminant Analysis (LDA) to reduce the dimensionality. In the end, the Maximum a posteriori Probability (MAP) Bayesian method was used for the generation of the model. The aim of this work was to differentiate between absent and present in pigment network. The masks of the skin lesions were obtained to classify the images' pixels as background (pixels outside the lesion), absent (pixels of the skin lesion with no pigment network) and present (pixels of the skin lesion with pigment network). The test was carried out over 734 images, without reported results in pigment network detection.

Barata et al. [13] present a work focused on the detection of pigment network using a bank of directional filters and morphological operations. The proposed method follows a set of three sequential steps. In the pre-processing step, a given RGB image is converted to grayscale and two types of artifacts are removed: hair and reflections. In the second step, regions with pigment network are detected using intensity and spatial organization properties. In this stage, a bank of filters is applied to perform an enhancement of the network, where the lines associated with pigment network are sharpened in relation to the background. Then, a binary mask of the network is obtained by performing morphological operations based on the eight-connectivity criterion. In the final stage, the mask generated is used to

assign a binary label to each image: with or without pigment network. For this purpose, features were extracted from the mask and they are used to train the AdaBoost classifier. The algorithm was tested using a dataset containing 200 images, with groundtruths segmented by experts, achieving a sensitivity of 91.1% and a specificity of 82.1%.

In [14], Arroyo et al. present an algorithm for detection of pigment network in dermoscopic images, with its design consisting of two blocks. In the first block, a supervised machine learning process is carried out, generating a set of rules that, when applied over the image, provide a mask with the pixels that are candidates to be part of the pigment network. In the second block, the generated mask is processed by performing a structural analysis to find the structures that correspond to the pigment network. Then, the diagnosis is made, determining whether it has pigment network or not and to generate the mask corresponding to that structure, if any. The method was tested using a data set of 220 images, achieving a sensitivity of 86% and a specificity of 81.7%.

Kawahara et al. [18] propose a fully convolutional neural network to detect pigment network from dermoscopy skin lesion images. Their neural network architecture uses interpolated feature maps from several intermediate network layers and addresses the imbalanced labels by minimizing a negative multi-label Dice-F_1 score, where the score is computed across the mini-batch for each label. This approach ranked first in the *2017 ISIC/ISBI Part 2: Dermoscopic Feature Classification Task* challenge over both the validation and test datasets, achieving a sensitivity of 80.3% and a specificity of 95.6%. It should be noted that these scores are based on a superpixel classification, and not on a pixel-wise classification, due to aspects related to the challenge where the authors participated.

This thesis presents an approach for pigment network detection using deep learning techniques, following the trend where the work of Kawahara et al. [18] is comprehended.

Chapter 3

Deep Learning

The goal of this chapter is to introduce deep learning background and concepts that will be used throughout this thesis. This chapter starts by addressing the early years of this field and its recent evolution, then some basic background on Neural Networks and, finally, Convolutional Neural Networks, which are the type of Neural Networks that has been having an increasing success in computer vision tasks, attracting interest across a huge variety of domains, such as visual object recognition, object detection and many other domains [16], including dermoscopy image analysis [17].

3.1 Early years

Despite the recent increasing interest in Neural Networks (NNs), research in this field started many decades ago [32]. The first step towards Artificial Neural Networks (ANNs) came in 1943, when McCulloch and Pitts [33] introduced a computational model for neural networks based on mathematics and algorithms called threshold logic. They showed that even a simple type of neural network could, in principle, compute any arithmetic or logical function. In 1949, Hebb [34] introduced the first ideas about unsupervised learning, by creating a learning hypothesis based on the mechanism of neural plasticity, which became known as Hebbian learning. In 1958, another major breakthrough was made by Rosenblatt [35]. Using the McCulloch-Pitts neuron and the findings of Hebb, he developed the first perceptron that could learn in the Hebbian sense, through the weighting of inputs. This work was instrumental in the later formation of neural networks.

Ivakhnenko [36] revolutionized the neural networks field by presenting the first functional deep feedforward multilayer perceptron using statistical methods at each layer to find the best features and forward them through the system. In 1982, Fukushima [37] created a multilayer ANN capable of learning how to recognize visual patterns, which inspired the well-known Convolutional Neural Networks (CNNs). These works established the starting point of Deep Neural Networks (DNNs).

Nevertheless, it was still to be developed a powerful learning algorithm that could efficiently update the network's weights during the training phase. In the mid-80s, Rumelhart et al. [38] showed experimentally that backpropagation could generate useful internal representations of input data in the hidden layers of neural networks, by propagating the error values back using the chain rule to compute the gradient of the error with respect to all the weights, minimizing a loss function. In 1989, LeCun et al. [39] applied backpropagation to Fukushima's weight-sharing convolution neural networks [37] to recognize handwritten zip code digits. This combination became the essential ingredient of many feedforward neural networks.

Nonetheless, Hochreiter [40] discovered that the NNs suffered from the vanishing or exploding gra-

dient problem, which becomes a key issue the deeper a network gets. The Long Short-Term Memory cell [41] was robust to this problem, while outperforming Recurrent Neural Networks (RNNs) on tasks related to sequential data.

After the success of backpropagation, neural networks gained a huge popularity in the 1990s, before falling into disuse when other machine learning techniques became more popular [42].

3.2 Recent tendencies

The recent advancements of deep learning were, in part, highly motivated by the failure of traditional algorithms to generalize well in Artificial Intelligence (AI) tasks such as speech recognition and object recognition [42].

However, an important part of these late improvements can be attributed to two major factors. First, neural networks have become deeper, mainly, due to high-performance computing systems, with the availability of faster Central Processing Units (CPUs) in the beginning and the arrival of Graphics Processing Units (GPUs) afterwards. Additionally, there was a better software infrastructure for distributed computing [42]. Second, the increasingly larger datasets enabled a more powerful statistical generalization, allowing to successfully train deeper networks. These two key aspects led the scientific community to concentrate on developing algorithms that scale up to large datasets.

The current intensity of interest in deep learning began when, in 2012, Krizhevsky et al. [43] (SuperVision group) won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) by developing a deep convolutional neural network, which was later named *AlexNet*. ImageNet is an image database with over 15 million labeled images belonging to roughly 22,000 different categories. ILSVRC uses a subset of ImageNet with roughly 1.2 million images with 1000 different categories according to the WordNet hierarchy and has 732 to 1300 images per class, being one of the most important datasets available for computer vision tasks. This challenge consisted of one or more of the following tasks: image classification, single-object localization and object detection [44].

Table 3.1 shows the evolution of the winners' results for the image classification challenge from 2011 to 2015 [44] [45]. Before 2012, the top results were achieved relying on Support Vector Machines (SVMs). In 2012, the SuperVision group decreased the Top-5 Error from 25.8% to 16.4%, outperforming the commonly used models at the time. Since then, deep CNNs have systematically won the ImageNet competition, making them the state-of-the-art algorithms for object classification.

Year	Top-5 Error	No. of layers	Name	Reference
2011	25.8	-	XRCE	[46]
2012	16.4	8	AlexNet	[43]
2013	11.7	8	ZFNet	[47]
2014	6.7	22	GoogLeNet	[48]
2015	3.6	152	ResNet	[45]

Table 3.1: Summary of winning networks in the ImageNet classification challenge (adapted from [44] [45])

3.3 Background

Before reviewing the deep learning methods frequently used in image classification tasks, the theoretical foundations of neural networks are introduced.

3.3.1 Computational model of neuron

Neural networks are computational models inspired by the way biological neural networks process information in the human brain. The densely interconnected set of simple units in an ANN can be compared, on an oversimplified scale, with the complex webs of interconnected neurons in a brain [49].

The neuron is the basic operation unit of the nervous system. Essentially, the neuron consists of three major functional components: dendrites, cell body and axon (see Figure 3.1a).

A neuron receives input signals (impulses) from other neurons through its dendrites. Thereafter, the neuron's cell body generates output signals which are transmitted along the axon. The axon eventually branches out into strands and sub-strands. At the terminals of these strands are the synapses, which are elementary structures and functional units between two neurons. The synapse allows the axon to connect to the dendrites of neighbouring neurons. These connections between neurons highly depend on the synaptic strength at that synapse, which, ultimately, evaluates the 'importance' that the neuron should give to the received signal. Since a neuron has a large number of dendrites, it may receive many signals simultaneously from others neurons and those are all processed at the cell body. If the electric potential in the cell body reaches a given threshold, the neuron fires (electrical activity in the form of short pulses is generated) and the resulting electric signal is sent down the axon [50] [51].

The analogy between the artificial neuron (also called node) and the biological neuron is that the connection between nodes represents the axon and the dendrites, the connection weights represent the synaptic strength and the threshold approximates the activity in the cell's body. This idea is visually represented in Figure 3.1.

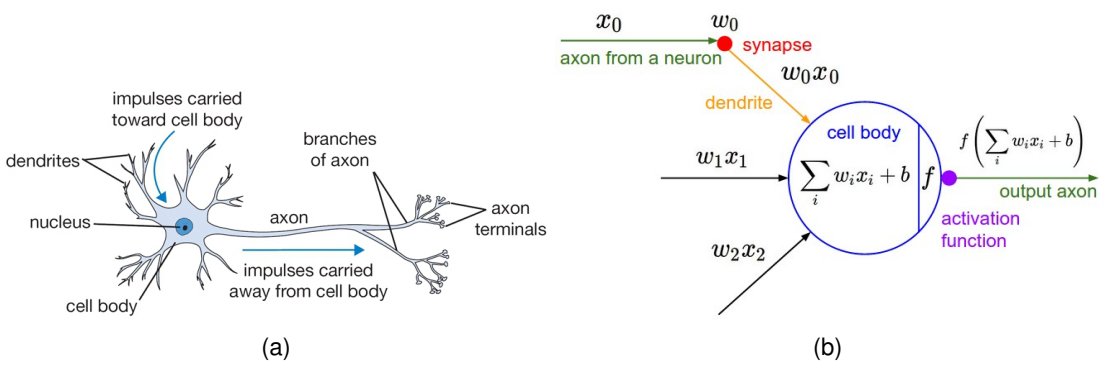


Figure 3.1: (a) Biological neuron scheme; (b) Artificial neuron scheme (adapted from [50]).

As referred above for the biological neural network, an artificial neuron may be connected to several other neurons creating a network. Each neuron's input has an associated weight, which is assigned taking into account its 'relative importance' compared to other inputs. Thus, these weights are learnable parameters that control the influence of one neuron on another [50]. The activity in the cell's body of the artificial neuron is mathematically modelled according to

$$y = f\left(\sum_{i=0}^{N-1} w_i x_i + b\right), \tag{3.1}$$

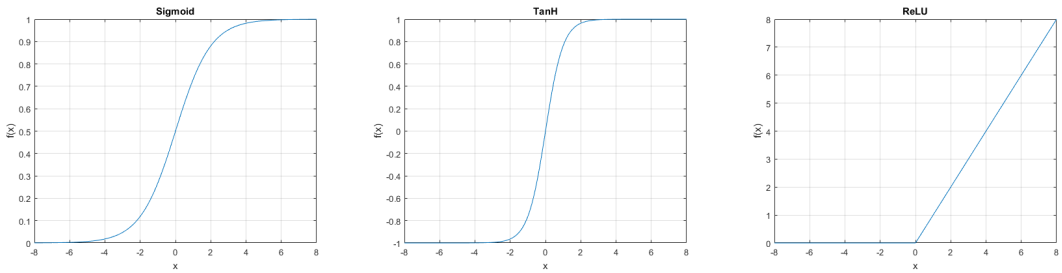
where x_i denotes the input signal received from the connection i, w_i is the weight assigned to the connection i, N is the number of input signals, b refers to an additional input 1 with assigned weight b - commonly called *bias* -, which basically adds a constant value to the output, enhancing the flexibility of the node. The output y is a function of the parameters described above, corresponding to the signal transmitted to the neighbouring neurons. The function f is called the activation function, which introduces a non-linearity to the output of the neuron by applying a non-linear function over its input.

The most commonly used activation functions are the Sigmoid (equation (3.2)), the Hyperbolic Tangent (TanH) (equation (3.3)) and the Rectified Linear Unit (ReLU) (equation (3.4)), which are illustrated in Figure 3.2.

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{3.2}$$

$$f(x) = \frac{2}{1 + e^{-2x} - 1}. \tag{3.3}$$

$$f(x) = \begin{cases} 0, \ if \ x < 0 \\ x, \ if \ x \geq 0 \end{cases} . \tag{3.4}$$



(a) (b) (c)

Figure 3.2: Non-linear activation functions used in neural networks. (a) Sigmoid; (b) TanH; (c) ReLU.

The main purpose of these activation functions is to introduce non-linearity to the network. Most part of the data is non-linear, so the neurons must learn non-linear representations [16].

3.3.2 Network Architecture

Artificial neural networks can be seen as weighted directed graphs, where the artificial neurons are nodes and the directed edges with assigned weights are connections between neurons' outputs and neurons' inputs.

Considering the connection pattern, i.e. the architecture, ANNs can be divided into feedforward networks and recurrent networks [51]. In a broad sense, in a feedforward network, the graph has no loops, while in a recurrent network (also called feedback network), loops occur due to feedback connections.

The most common feedforward network is the Multi-Layer Perceptron (MLP), which is composed of multiple neurons organized into layers that have unidirectional connections between them [51], as illustrated in Figure 3.3

In general, the architecture of an ANN can be divided into the input layer, one or more hidden layer(s) and an output layer.

The input layer is composed of input nodes that receive information from the external environment and pass on the information to the hidden nodes of the following layer. These inputs are commonly normalized within the limit values produced by activation functions, resulting in better numerical precision for the mathematical operations performed by the network [53].

The hidden layers are responsible for extracting patterns associated with the data being analysed. These layers perform most of the internal processing from the network and the information is transferred from the previous layer to the following layer. The number of hidden layers and nodes must be carefully considered [54]. First, one problem may arise related to the increased training time of the network that can make the network impossible to adequately train. Furthermore, using too few neurons and/or layers

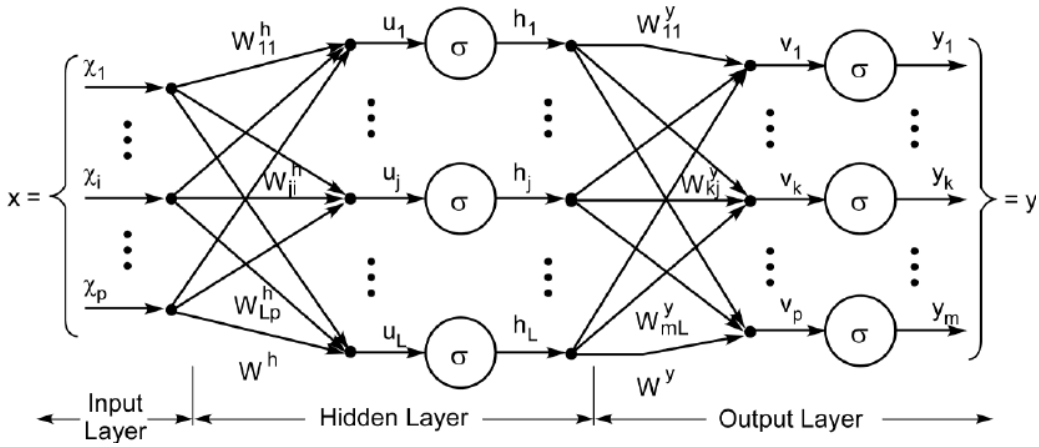


Figure 3.3: Example of a MLP architecture (adapted from [52]); the activation function is denoted by σ.

may result in underfitting, which occurs when the model cannot adequately represent the data. If the opposite occurs, overfitting may occur. In this case, the NN is so flexible that the limited amount of information contained in the training set is not sufficient to train all the neurons in the hidden layers. Thus, the network's parameters should be set up taking into account that the network must have the ability to generalize well, i.e., the trained ANN must be able to classify new data well.

In the output layer, the output nodes receive the processed data from the hidden layers and the information is transferred to the outside of the network. Generally, this layer is taken to represent class scores, e.g. for classification tasks, where each output node corresponds to a class. In such problems, the node with the highest outputted value expresses the class predicted by the NN as the most probable one.

3.3.3 Loss Function

Before explaining how the learning process works, it is important to define the concept of a loss function.

Loss function, also called cost function, is an important part in ANNs, which is used to measure the inconsistency between a predicted value $\hat{y}$ and a true label y. In a broad sense, this function indicates how good the neural network is for a certain task. One of the most used loss functions is the Mean Squared Error (MSE) loss function, which is indicated in equation 3.5.

$$\mathcal{L}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 \tag{3.5}$$

To compute the loss function, it is necessary to go over each training example in a dataset, compute y for that example, and then compute the function defined above. If the predictions are accurate, the loss function will output a low number. Otherwise, it will output a high number and the model should be improved.

3.3.4 Training process

At a high level of abstraction, a neural network can be seen as a *'black box'* with two operation modes: learn and predict. During the learning process, the NN receives pairs of inputs and desired outputs and updates its internal parameters accordingly, so that the computed output get as similar as possible to the desired output. In the prediction process, the NN takes the input and generates an output,

according to the internal parameters obtained during the learning process.

The learning process of a feedforward neural network requires that the NN receives as input a set of training data that includes a dataset of examples $\mathcal{X}$ and the corresponding desired outputs $\mathcal{Y}$, that are commonly referred as labels. The goal of this learning process is to assign the correct weights to the connections of the neural network.

The training process starts by (randomly) assigning weights to the connections. For each input in the training set, the neural network is activated, i.e. the input is feedforward throughout the network, and the output is observed. After that, the output is compared to the desired output and the error between both is measured using a loss function and its derivative is computed. This error is backpropagated through the entire network and the weights are adjusted accordingly using an iterative optimization method called Stochastic Gradient Descent (SGD). This method performs a parameter update for each training example x_i and label y_i, using the following equation:

$$\theta_t = \theta_{t-1} - \eta \nabla_{\theta_t} \mathcal{L}(\theta_{t-1}, x_i, y_i), \tag{3.6}$$

where θ is the weight vector, η is the learning rate and $\nabla_{\theta_t} \mathcal{L}(\theta_{t-1}, x_i, y_i)$ is the gradient of the loss function with respect to the weights. For the computation of the gradient, it is used the backpropagation algorithm which uses the error values. The process is repeated until a stopping criterion is satisfied. Once this criterion is fulfilled, the neural network is considered to have learned from the training data. It should be mentioned that this technique can only be applied to networks with differentiable activation functions. Summarily, the training process is performed by minimizing a loss function using an optimization algorithm.

3.4 Convolutional Neural Networks

This section focuses on Convolutional Neural Networks, introducing their relevance to image recognition tasks. In addition, it is performed a more detailed description of architectures proposed for image segmentation.

3.4.1 Motivation

Using standard feedforward neural networks, as the multi-layer perceptron, is impractical in image classification tasks, since the number of weights is very high. For instance, an input image of size 200×200 would require the neural network to have 40,000 input nodes, which corresponds to one neuron per pixel. If by any chance the input layer is followed by a hidden layer of 20,000 nodes, then this would imply that there would be 800 million weights (40,000×20,000) between the neurons [55]. This huge number of connections is only for the first layer and does not account for additional layers.

To overcome this problem, convolutional layers have been proposed. These layers replace a large number of connections by a set of convolutional filters [55]. Regarding the previous example, it is much simpler to learn a set of convolutional filters (e.g. each one of size 11×11), than a large matrix with 800 million entries. In such layers, the weights are shared by several neurons. The input image is filtered using different convolutional masks and to each value of the output matrix is applied a non-linear activation function. The resulting matrix after the application of a filter and a non-linearity is called an activation map. Besides convolutions layers, pooling layers are also incorporated into this architecture not only to reduce the computation cost but also to reduce overfitting [50].

Compared to traditional feedforward NNs with layers of similar size, CNNs have much fewer connections, and therefore, fewer parameters to estimate, which makes them easier to train [43].

Nevertheless, CNNs are quite similar to regular NNs in many aspects. As traditional NNs, CNNs are comprised by an input layer, one or more hidden layers and an output layer, which consist of neurons that have learnable weights and biases. The feedforwarding process and the training algorithms are also kept in such networks. Despite that, the hidden layers of a CNN have, typically, a smaller number of parameters but are deeper when compared to regular ANNs [50].

For image classification tasks, the CNN architecture tends to follow a structure composed of convolution layers, pooling (sub-sampling) layers and fully connected layers, as depicted in Figure 3.4. Such an architecture can be divided into two main functional parts: the first part, composed by the convolutional layers and the pooling layers, is responsible for feature extraction, while the second part, composed by the fully connected layers, is responsible for the classification task, which is based on the features that were extracted in the previous part [56].

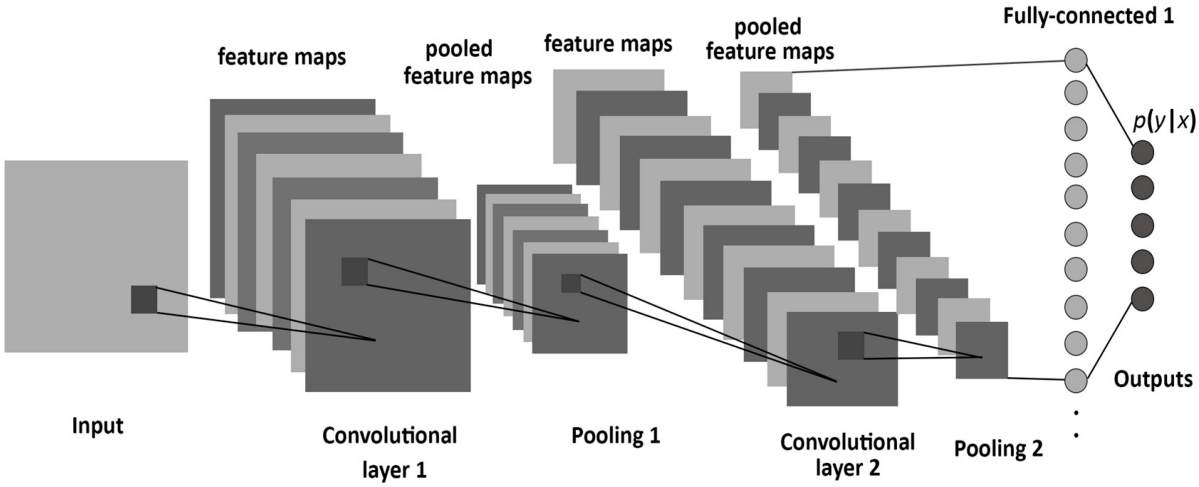


Figure 3.4: Example of a CNN architecture (adapted from [57]), consisting of convolutional, pooling and fully connected layers.

Although CNNs are widely used in image classification problems, there are many visual tasks, especially in medical image analysis, where the desired output should include localization, providing not only the classes but also additional information regarding the spatial localization of those classes. Thus, it is required that a class label is assigned to each pixel, which corresponds to the main idea behind semantic image segmentation using CNNs [58]. Figure 3.5 depicts an example of a network for image segmentation and the actual segmentation output for an input image.

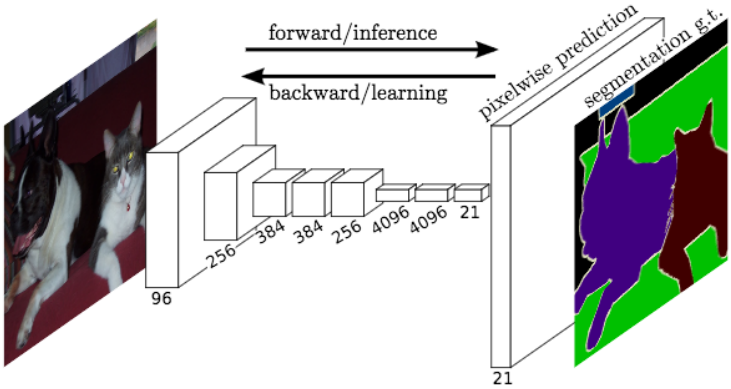


Figure 3.5: Example of an architecture for semantic image segmentation (adapted from [59]); the number below the boxes represent the number of filters of each layer.

Recent semantic segmentation algorithms convert the regular CNN architecture developed for classification tasks to Fully Convolutional Networks (FCNs) [59], which are built only from locally connected layers, such as convolution, pooling, deconvolution and unpooling. There is no dense layer (fully connected layer), which results in reducing the number of parameters and computation time. In order to obtain a segmentation map, such architectures, usually have two parts:

1. Downsampling path: captures the semantic/contextual information.
2. Upsampling path: recovers spatial information.

The downsampling path is used to extract and interpret the context (*what*), while the upsampling path enables precise localization (*where*), allowing to obtain a coarse label map from the network by classifying every local pixel in the input image. Additionally, the upsampling path introduces the idea of deconvolution network that generates a dense pixel-wise class probability map by consecutive operations of unpooling and deconvolution [60] (see Figure 3.6).

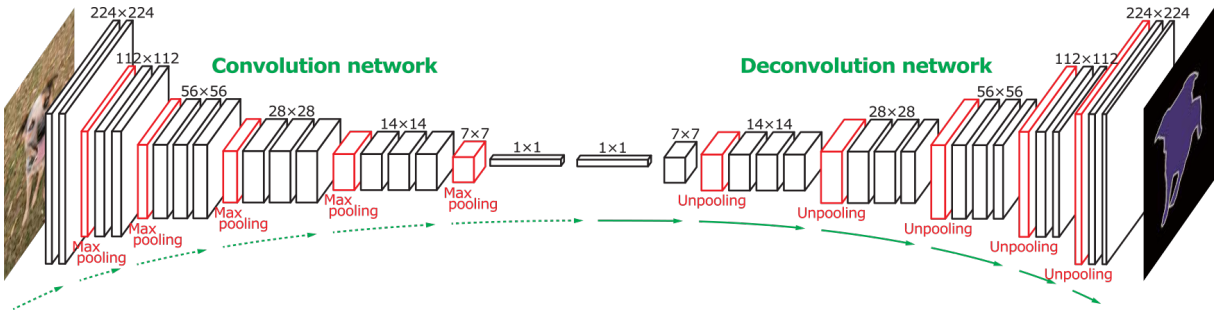


Figure 3.6: Example of CNN architecture for semantic image segmentation (adapted from [60]).

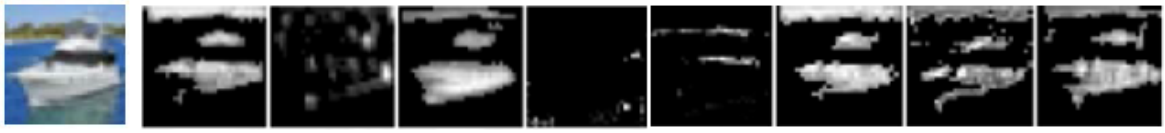
As depicted in Figure 3.6, the downsampling path is done by a convolution network, where convolutional and pooling layers are used. Regarding the upsampling path, it is done using a deconvolution network, where deconvolution and unpooling layers are used.

For the purpose of this thesis, these types of layers that compose the CNN architecture are explained in the following subsections.

3.4.2 Convolutional Layers

The convolutional layer is the core block of a CNN. Each layer of this type is composed by a set of filters (also known as kernels), which detect what features are present in an input image, through a convolution operation [50]. The input image may have a depth greater than one, i.e., RGB images have 3 channels, which imply a depth of 3, while greyscale images only have 1 channel, and, therefore, a depth of 1. Thus, the filter must have the same number of channels as the input image. The filter's values are called weights and have to be learned to detect the aforementioned features.

During the convolution process, a filter slides over the original image by a predetermined number of pixels, which is called stride. For each position, an element-wise multiplication is performed between the input matrix and the filter, and then the multiplication outputs are added to get the final outcome [61]. This summation is followed by the application of a non-linear activation function, obtaining an activation value. Each one of these activation values represents one single element in the output matrix, commonly called activation map, which aggregates the extracted features by the filter (see Figure 3.7 for examples). These filters allow to perform different operations such as edge detection, sharpen and blur, by changing the numerical values within the matrix. A higher number of filters leads to more image features getting extracted and the network is better at recognizing patterns [50].



(a) (b)

Figure 3.7: (a) Input image; (b) Activation maps obtained after convolution with 7 different filters (adapted from [62]).

Each convolution layer has an associated set of hyperparameters which determines the size of the activation map. The hyperparameters that have to be decided upon are the number of filters, *K*, the filter size, *F*, the stride, *S* and the amount of zero-padding, *P*, which refers to the added number of rows/columns with zero values around the input matrix, allowing to control the spatial size of the output volumes. The number of filters is the most variable parameter. Using more filters results in a more powerful model, but the overfitting risk increases due to the increased parameter count.

In Figure 3.8, it is illustrated a convolution between a kernel of size $3 \times 3 \times 3$ and an input image of size $N \times N \times 3$ (RGB image), using a stride $S = 1$ and a padding $P = 1$, resulting in a $(N-1) \times (N-1) \times 1$ activation map.

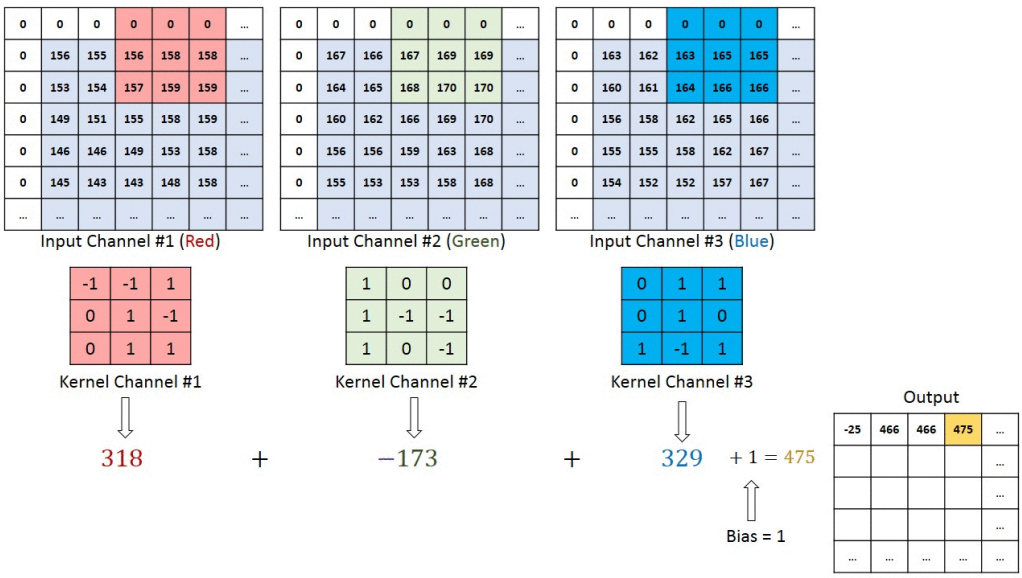


Figure 3.8: Convolution between a filter (kernel) and a multi-channel input image (adapted from [63]).

In a more detailed manner, a convolution layer works as follows [50]:

1. The (i)-th convolution layer receives as input a matrix with dimensions $W_i \times H_i \times D_i$, where W_i refers to the width, H_i to the height and D_i to the depth of the matrix. For the first convolutional layer, the input is the original image, while for the following ones, the input corresponds to the activation maps generated in the previous layer.

2. In the following step, each one of the filters of size $F_i \times F_i \times D_i$ convolves with the input matrix, producing a matrix of size $W_{i+1} \times H_{i+1} \times 1$ for each filter (see Figure 3.8 for a simple example). This computed matrix corresponds to the above-mentioned activation map, where W_{i+1} and H_{i+1} can be computed as demonstrated in equations (3.7) and (3.8), respectively.

$$W_{i+1} = \frac{W_i - F + 2P}{S} + 1 \tag{3.7}$$

$$H_{i+1} = \frac{H_i - F + 2P}{S} + 1 \tag{3.8}$$

3. After all filters have convolved with the input images, an activation volume of size $W_{i+1} \times H_{i+1} \times D_{i+1}$ is produced, where $D_{i+1} = K_i$ is the number of filters. It is this generated activation volume that is forwarded to the next layer of the network.

After each convolutional layer, a non-linear activation function is applied to each value of the generated activation maps. In tasks where CNNs are used, it was found that ReLU is the activation function that guarantees a better performance in most situations, allowing the network to train several times faster than the other known activation functions [43], which has a great influence when training deep CNNs with large datasets. Additionally, it does not suffer from the vanishing or exploding gradient problem in most situations. However, if the neural network becomes too deep, this problem may still happen [45]. Essentially, ReLU is an element-wise operation where negative pixel values in activation maps are replaced by zero, as depicted in Figure 3.9.

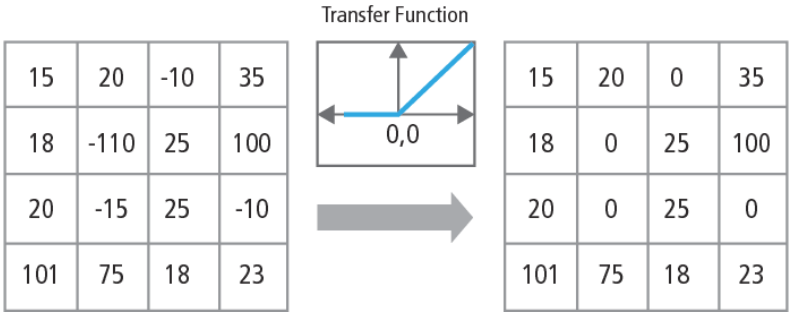


Figure 3.9: Pictorial representation of ReLU functionality (adapted from [61]).

As previously mentioned, the entries of a filter's matrix correspond to the weights of a neuron, which means that the training of a CNNs is similar to the training of a regular ANN. During the training process, the weights are updated and tend to converge. This allows the filters to be able to extract relevant features from the images. Nonetheless, not all types of features can be extracted using the same convolution layer. As mentioned before, the convolution layers are applied one after the other, which means that features extracted by a filter in one layer are built upon the features extracted in previous layers. Hence, the extracted features at each layer tend to have an increasingly higher level of abstraction, as illustrated in Figure 3.10. The first layers of a network capture low level features such as lines, edges or shapes, and deeper levels provide more complex and abstract features [64].

3.4.3 Pooling

The goal of pooling is to reduce the dimension of each activation map [61], keeping the most important information. By applying pooling, not only the input images get smaller and *manageable*, but also the number of parameters and computation performed in the network are reduced, which reduces overfitting, allowing the model to generalize better for new inputs [50]. Additionally, the network gets invariant to small transformations, distortions and translations in the input image [57].

There are different types of pooling, such as the average pooling, the max pooling and the sum pooling. However, the most regularly used pooling operation is the max-pooling [50]. This operation can be seen as a *max filter*, where each $R \times R$ region of the activation map is replaced by its maximum value. This operation is illustrated in Figure 3.11, using a filter with size 2×2 and a stride of 2.

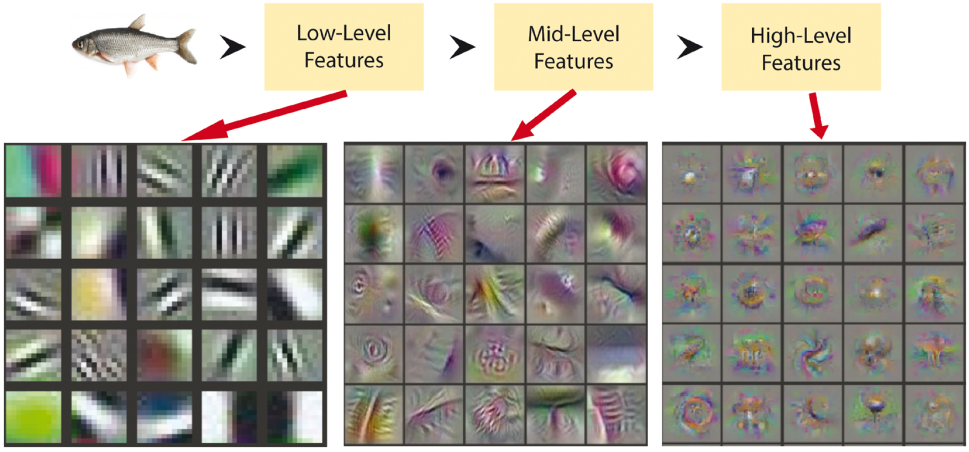


Figure 3.10: Features visualization example (adapted from [65]).

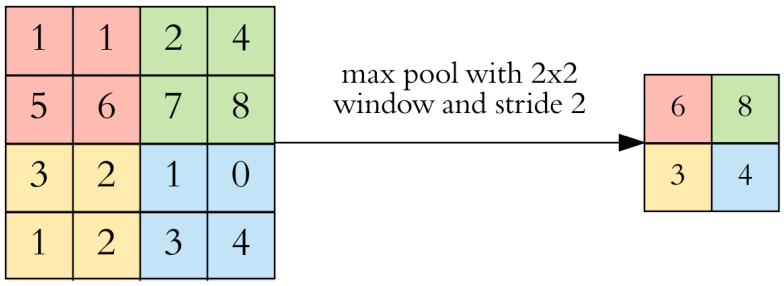


Figure 3.11: Pictorial representation of max-pooling functionality (adapted from [50]).

If, by chance, there is a small distortion in the input, the output after pooling will not change, due to the maximum value being taken in that local neighbourhood.

3.4.4 Fully connected layers

As mentioned in Section 3.4.1, CNN architectures designed for image segmentation do not use fully connected layers. However, for a better comprehension about this theme, they are addressed in this section. After the convolution and pooling layers in CNN architectures for image classification tasks, the classification part consists of few fully connected layers. In a fully connected layer, as the name suggests, each neuron belonging to a certain layer is connected to all neurons from previous and following layers. This architecture is similar to that of the regular ANNs discussed in Subsection 3.3.2. As described in Subsection 3.4.2, the deeper convolutional layers are able to provide high-level features from the input image, such as a dog face, resulting in several activation maps. The idea behind the fully connected layers is to use the extracted features from the previous stages in order to classify the input image into a certain class, e.g., dog, horse, fish, boat, etc. Thus, the high-level extracted features are fed to the first fully connected layer, where each value of the activation maps is connected to all neurons of this layer. After that, the information is combined, propagating forward the successive outcomes from each layer. The output layer is a fully connected layer with as many nodes as the number of output classes, where the activation function used is, commonly, a softmax classifier, that essentially receives an array containing different activation values where each value corresponds to a class [50]. This classifier estimates the posterior probability of each class label over K classes [57], according to

$$y_i = \frac{exp(-z_i)}{\sum_{k=1}^{K} exp(z_j)}, \tag{3.9}$$

where y_i refers to the probability of the input image belonging to class i and z_j corresponds to the activation value j. The sum of the output probabilities is 1, which is assured by the softmax function.

For a visual perception of how the layers are connected, Figure 3.12 shows an example of a CNN architecture for a handwritten digit recognition task, where it is illustrated, in an abstract way, how the neurons in the fully connected layer are connected to all the neurons of the previous and following layers. In the output layer, it can be seen that the layer has 10 neurons, one for each output class (digits from 0 to 9).

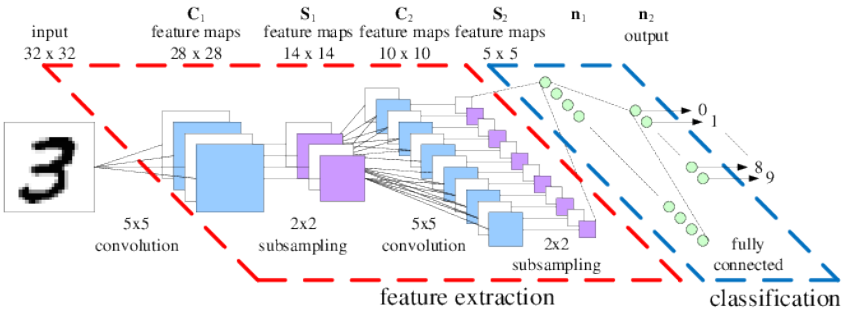


Figure 3.12: Example of a CNN architecture for a handwritten digit recognition task (adapted from [66]).

3.4.5 Deconvolution

The use of deconvolution [1] arises from the need to use a transformation going in the opposite direction of a normal convolution, i.e., from something that has the shape of the output of some convolution to something that has the shape of its input while maintaining a compatible connectivity pattern. For instance, this transformation can be used as the decoding layer of a convolutional autoencoder or to project feature maps to a higher-dimensional space [67].

Transposed convolution - also called deconvolution - can be seen as the backward pass of a corresponding traditional convolution [68]. Contrary to the traditional convolution that connects multiple input activations to a single activation, it associates a single activation with multiple output activations. Figure 3.13 shows a deconvolution operation of a 3×3 kernel over a 5×5 input using a stride of 2 and no zero-padding. This operation is actually equivalent to convolving a 3×3 kernel over a 2×2 input, with 1 zero inserted between inputs, padded with a 2×2 border of zero using a stride of 1. However, the stride of deconvolution gives the dilation factor for the input feature map. Firstly, the deconvolution upsamples the input by a factor of the stride value with padding. Then, it performs the convolution operation on the upsampled input.

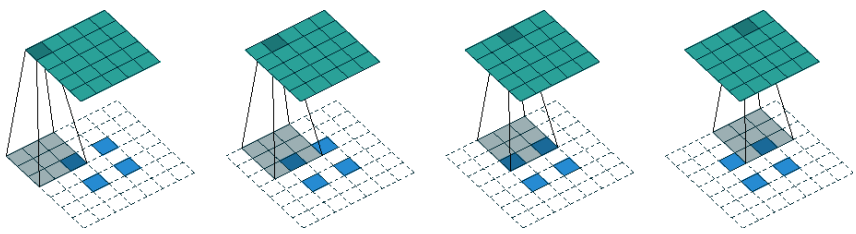


Figure 3.13: The transpose of convolving a 3×3 kernel over a 5×5 input using 2×2 strides (i.e., N = 5, K = 3, S = 2 and P = 0) (adapted from [67]). The input is represented in green, while the output is represented in blue.

[1] Deconvolution is an unfortunate name because this is no different than convolution itself (and deconvolution has a slightly different meaning in signal processing). It is usually called transposed convolution [67].

Recently, an alternative strategy was proposed to increase the amount of relevant information that a network can cover. The dilated convolution, also called atrous convolution, increases the size of the kernel by inserting zeros between the kernel elements, without loss of resolution or coverage [69]. The dilation rate is controlled by an additional hyperparameter d.

Dilated convolutions are used to increase the receptive field of output units without increasing the kernel size, which is especially effective when multiple dilated convolutions are stacked one after another [67]. Figure 3.14 provides an example of a dilated convolution, convolving a 3×3 kernel over a 7×7 input with a dilation factor of 2.

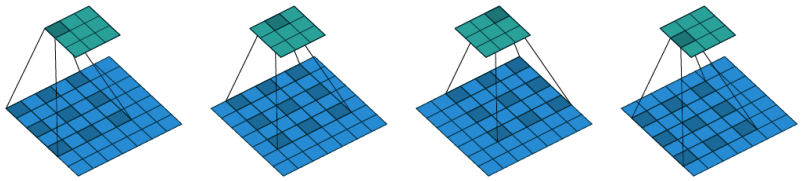


Figure 3.14: (Dilated convolution) Convolving a 3×3 kernel over a 7×7 input with a dilation factor of 2 (i.e., N = 7, K = 3, d = 2, S = 1 and P = 0) (adapted from [67]).

Another technique for upsampling feature maps to a higher resolution is unpooling. In CNNs, the max pooling operation is non-invertible. However, it is possible to obtain an approximate inverse by recording the locations of the maxima within each pooling region in a set of switch variables. In the deconvolution network, the unpooling uses these switches to place the reconstructions from the layer above into appropriate locations, preserving the structure of the stimulus [47]. Figure 3.15 shows an example of the application of the unpooling operation.

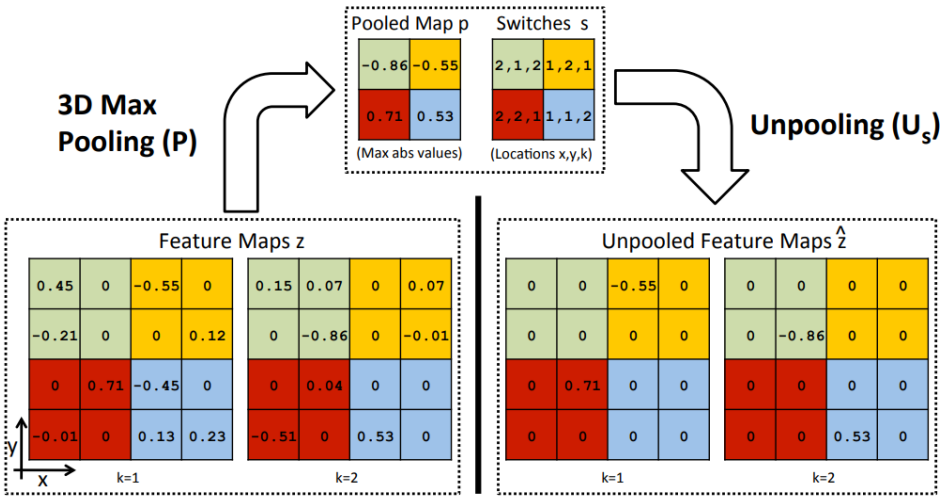


Figure 3.15: Example of the application of the unpooling operation (adapted from [70]).

The pooling operation P is applied to the feature maps z, yielding pooled maps p and switches s that record the location of the maximum. Given the pooled maps and switches, the unpooling operation U_s can be performed, which inserts the pooled values in the appropriate locations in the features maps, setting to zero the remaining elements.

3.5 CNN architectures for image segmentation

One major issue of CNNs is that the spatial information of the image is lost when the convolution features are fed into the final fully connected layers of the network. Nonetheless, spatial information is especially important for semantic segmentation tasks. Hence, in 2015, Long et al. [59] proposed the FCN to overcome this limitation. The idea behind their approach was to take advantage of existing CNNs as powerful visual models that are able to learn hierarchies of features. They transformed those existing and well-known classification models, such as AlexNet [43], GoogLeNet [48] and ResNet [45], into convolutional ones, by replacing the final fully connected layers with transposed convolutional ones to output spatial maps instead of classification scores. By applying this operation, the original spatial dimensions of the input image can be recovered while performing semantic segmentation at the same time. This network has made it feasible to train models for pixel-wise semantic segmentation in an end-to-end fashion. Figure 3.16 depicts the architecture of the FCN.

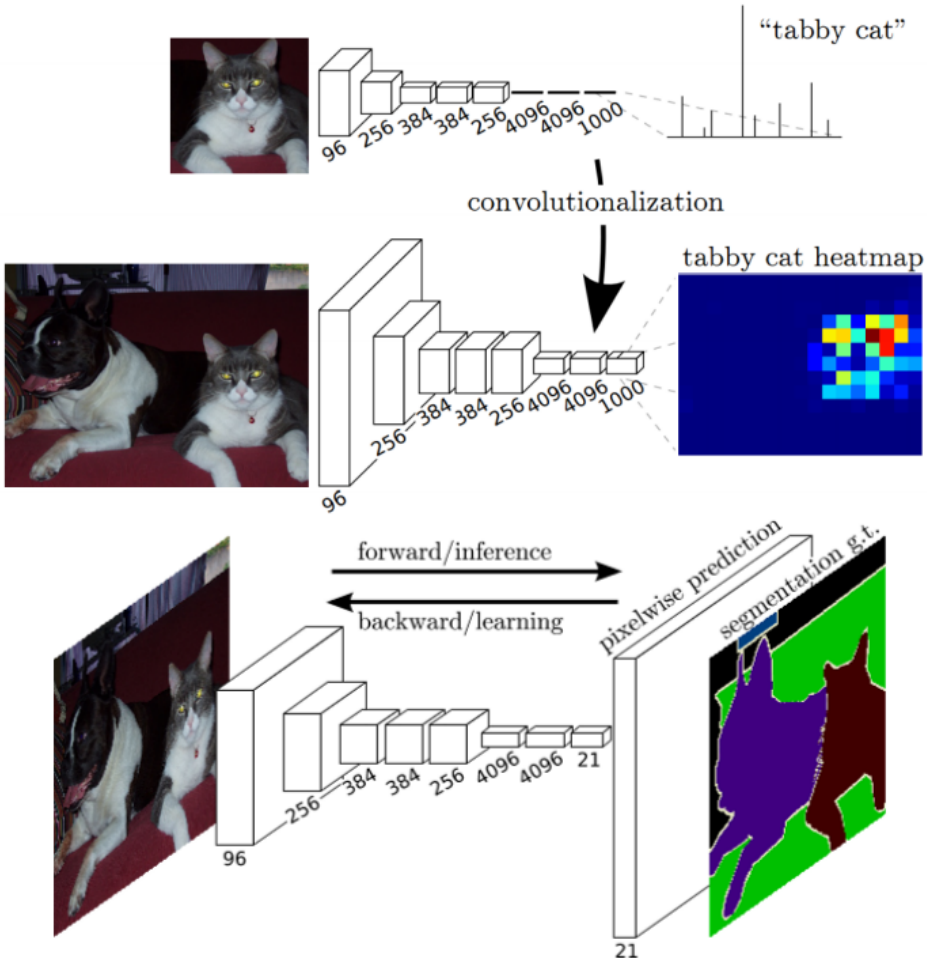


Figure 3.16: FCN architecture (adapted from [59]).

Since Long et al. popularized the CNN architecture for dense predictions without any fully connected layers, a large number of FCN-based [60] [71–76] methods have been proposed, promoting the application of deep learning strategies to image semantic segmentation. Recently, U-Net [77] became one of the most popular networks for biomedical image segmentation problems. Essentially, U-Net is a deep convolutional network that learns to segment images in an end-to-end setting, which means that it receives as input an image in its rawest form and produces an output segmentation map. The architecture of this network consists of a contracting path where the image's context is captured and a symmetric

expanding path that enables precise localization, which is illustrated in Figure 3.17.

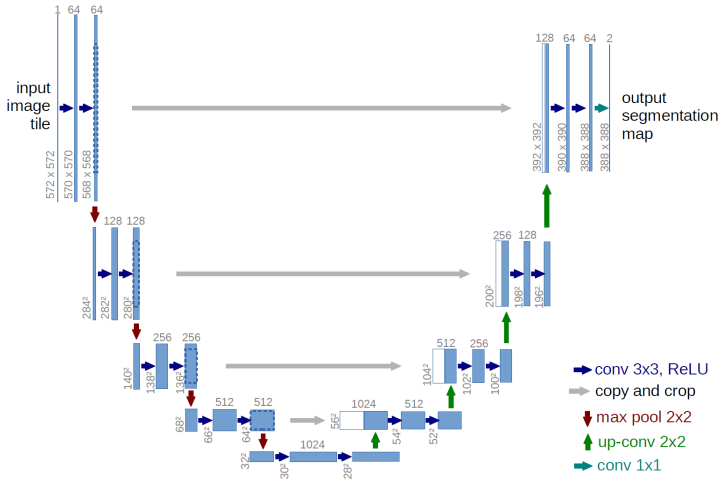


Figure 3.17: U-Net architecture (adapted from [77]).

In the figure above, each blue box represents a multi-channel feature map, while each white box represents copied feature map. On top of the boxes is indicated the number of channels and the size of the image is given at the lower left edge of each box. The arrows show the different operations.

As explained in Section 3.4.1, this type of network merges a convolutional network architecture (contracting path on the left side) with a deconvolutional network architecture (expanding path one the right side).

The contracting path is composed of a repetitive pattern of two 3×3 convolutions, each followed by a ReLU and a 2×2 max pooling operation with a stride of 2 for downsampling. At each downsampling step, the number of feature channels is doubled.

Regarding the expansive path, every step includes an upsampling operation of the feature map obtained in the contracting path, followed by sequences of 2x2 transposed convolutions, that halves the number of feature map channels. At each step, a concatenation of the resulting feature map with the correspondingly feature map obtained from the contracting path is performed, followed by two 3×3 convolutions and a ReLU layer after each convolution.

The entire network has 23 convolutional layers, where the last layer is used to map each component feature vector related to the desired number of output classes.

Chapter 4

Proposed Methodology

This section presents the techniques that were used in the development of a system to detect pigment network in dermoscopy images. It was used a deep learning approach where a CNN model was developed and trained using skin lesion images from a publicly available dataset [20].

4.1 Overall architecture

The architecture of the proposed system is shown in Figure 4.1. The main components of the system are two identical pre-processing modules, a CNN architecture module and a training module. The system receives as input a dermoscopy image which is pre-processed before being fed to a trained CNN model, generating as output a binary mask, where white indicates the presence of pigment network and black corresponds to the absence of pigment network. The CNN model is trained using a set of dermoscopy images and their corresponding groundtruth segmentations of pigment network, which are also pre-processed.

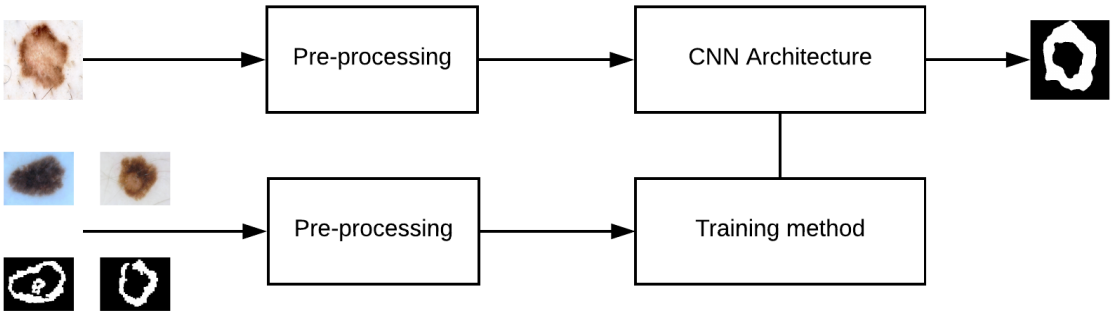


Figure 4.1: Block diagram of the system's architecture.

In the next sections, it will be detailed all the procedures taken in each one of the system's modules.

4.2 Pre-Processing

Before feeding the dermoscopy images into the CNN architecture, the raw input images are subjected to a set of pre-processing transformations. In this thesis, the pre-processing procedures applied to the input images are as follows:

- **Image resizing**. The input images were all of different sizes, varying from 576×768 to 6748×4499. Thus, to reduce the computational cost and the complexity of the problem, all images of the dataset were resized to a constant value of 256×256.

- **Image channels reduction**. For some experiments, the input RGB images were converted to grayscale images, reducing the images depth from 3 to 1. The grayscale conversion was performed using the ITU-R 601-2 luma transform [78], which is given by

$$T = 0.299R + 0.587G + 0.114B, \tag{4.1}$$

  where R denotes the red color channel, G is the green color channel and B is the blue color channel. This transform was chosen mainly due to its implementation easiness.

- **Image normalization** By dividing each input image by 255, each color channel was normalized from a range between 0 and 255 pixel values to a range between 0 and 1 normalized values. This technique is commonly used in image processing tasks to avoid contrast issues, namely distortions caused by lights and shadows [79].

4.3 CNN Architecture

Since the goal is to segment the pigment network regions in dermoscopy, the CNN architecture that was chosen is the U-Net [77], which was specifically designed by O. Ronneberger et al. to deal with biomedical image segmentation problems. The architecture of this network is described in Section 3.5 and it is illustrated in Figure 3.17. However, a few changes were made to the architecture of the network.

First, the depth of the channels at each stage of the network, including both convolution and deconvolution parts, was reduced by a quarter, i.e. instead of using 64, 128, 256, 512 and 1024 channels, 16, 32, 64, 128 and 256 channels were used. The reason behind this adjustment is related to limitations of hardware, mainly memory issues (see Section 5.3 for the implementation details).

The other alteration that has been done is that the final layer was reduced from two layers to only one layer. The pixel-wise classification is binary, i.e. each pixel is classified as foreground (white) or background (black), so it is possible to use only one output layer with a sigmoid activation function (see equation (3.2)) and reduce the number of parameters of the network. Thus, instead of the output pixel $p(x, y)$ belonging to the class of the node with the highest probability p_{ij}, it is considered foreground if its probability is above a threshold λ and background if it is below that given threshold λ, as shown in equation (4.2).

$$p(x, y) = \begin{cases} 1, \; if \; p_{ij} \geq \lambda \\ 0, \; otherwise \end{cases} \tag{4.2}$$

4.4 Training method of the network

The training methodology follows the block diagram depicted in Figure 4.2. Since the training is done from scratch, the weights of the model are randomly initialized. The training data is split into mini-batches, wherein each iteration (smaller loop in the figure), one mini-batch is feedforwarded through the CNN architecture. Then, the generated binary mask is compared with the desired binary output using a loss function (see next section for details of the used loss functions). After that, the derivative of the loss function is computed and the error is backpropagated through the network from the end to the

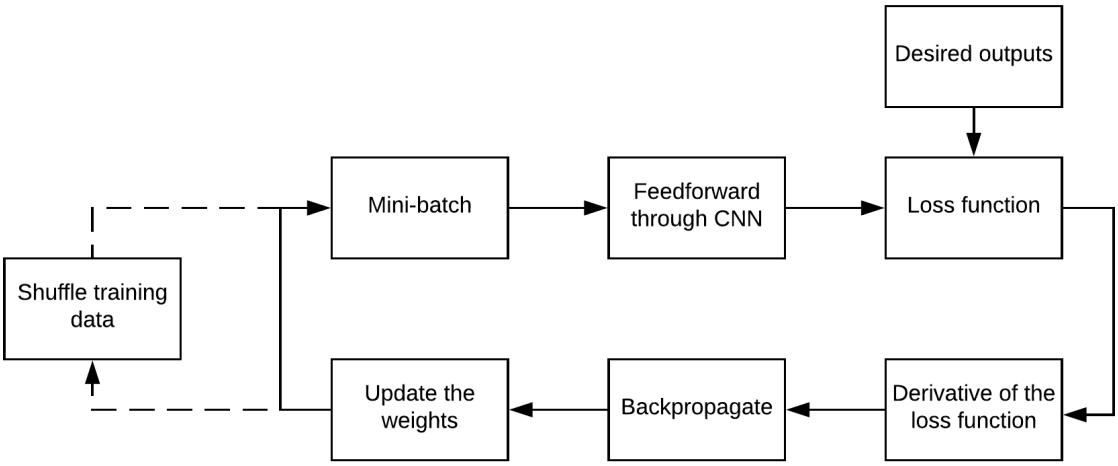


Figure 4.2: Training methodology. The small loop corresponds to each mini-batch training (full line). The bigger loop is performed once per epoch (loop including dashed line), after all mini-batches have been feedforwarded once in the current epoch.

start. Once all derivatives are computed, the weights are updated using a gradient-based optimization algorithm called Adam [80] and another mini-batch is feedforwarded through the CNN and the procedure repeats. After all the mini-batches have been feedforwarded once through the CNN (what is called an epoch), the training set is shuffled (bigger loop in the figure which contains the dashed line) and the aforementioned mini-batch training is repeated. The described procedure is performed for a chosen number of epochs. The training data is shuffled at each epoch to make sure that the model remains general and overfit less. By shuffling the data after each epoch, the risk of creating batches that are not representative of the overall dataset decreases and the estimate of the gradient will be better; it has been observed that if the order in which the mini-batches are visited is changed for each epoch, a faster convergence is obtained [81].

4.4.1 Definition of hyperparameters

For the learning process, some parameters must be carefully considered in order to achieve the best possible performance. Thus, the parameters to be defined before the training algorithm are the following:

- **Batch size**. The batch size is the number of samples fed to the network in one training iteration, in order to make one update to the model parameters. Since the entire dataset cannot be propagated into the neural network at once for memory limitations, it is divided into batches, which makes the overall training procedure require less memory and become faster. It should be highlighted that the higher the batch size is, the more memory will be needed and the slower is the training procedure.

- **Epochs**. The number of epochs denotes how many times the entire dataset has passed forward and backward through the neural network, i.e., one epoch is when every image has been seen once during training. Nevertheless, this concept should not be confused with iterations. The number of iterations corresponds to the total number of forward and backward passes, with each pass using a batch and depends on the the batch size, the number of epochs and number of training images. It is computed as follows:

$$\#iterations = \frac{\#epochs \times \#training\ images}{batch\ size} \tag{4.3}$$

- **Loss function**. As introduced in section 3.3.3, the loss function evaluates the inconsistency between the predicted value $\hat{y}$ and the groundtruth label y in every batch.

  For the purpose of this thesis, two loss functions were tested, namely the Cross-Entropy (CE) and the Weighted Cross-Entropy (WCE). The cross-entropy loss function for binary classification is given by

$$\mathcal{L}_{CE}(y, \hat{y}) = -\sum_{i=1}^{N} \left[y_i \, log\left(\frac{exp(\hat{y})}{1 + exp(\hat{y})} \right) + (1 - y_i) \, log\left(\frac{1}{1 + exp(\hat{y})} \right) \right], \tag{4.4}$$

  where N denotes the total number of training images.

  Since the training dataset is imbalanced (see Section 5.4), i.e., the number of foreground pixels (pixels where pigment network is present) is much smaller than the number of background pixels (pixels where pigment network is absent), it is necessary to find a way to overcome this issue. To tackle this problem, it is introduced a weight as a multiplicative coefficient for the positive class, i.e. foreground pixels, in the loss function. Thus, the weighted cross-entropy loss function for binary classification is as follows:

$$\mathcal{L}_{WCE}(y, \hat{y}) = -\sum_{i=1}^{N} \left[y_i \, log\left(\frac{exp(\hat{y})}{1 + exp(\hat{y})} \right) w + (1 - y_i) \, log\left(\frac{1}{1 + exp(\hat{y})} \right) \right], \tag{4.5}$$

  where $w = \frac{\#total\,pixels}{\#foreground\,pixels}$.

- **Optimizer**. The optimizer used in this work was the Adam [80], which is a gradient-based optimization algorithm that computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients; the name Adam is derived from adaptive moment estimation. This optimizer was chosen since it has little memory requirements and the hyperparameters β_1 and β_2 have intuitive interpretations, requiring little or no tuning.

  Besides storing an exponentially decaying average of past squared gradients v_t like RMSprop, Adam also keep an exponentially decaying average of past gradients m_t, similar to momentum [82], according to:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{4.6}$$

$$v_t = \beta_2 m_{t_1} + (1 - \beta_2) g_t^2, \tag{4.7}$$

  where m_t and v_t are the estimates of the first moment and the second moment of the gradients respectively, $\beta_1, \beta_2 \in [0, 1]$ are hyperparameters that control the exponential decay rate of these moving averages and $g_t = \nabla_{\theta_t} f_t(\theta)$ denotes the gradient, i.e. the vector of partial derivatives of the objective function f_t with respect to the parameters θ (weights and biases). After that, the learning rate η_t is updated using the external learning rate η, according to

$$\eta_t = \eta \frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t} \tag{4.8}$$

  and the parameters θ_t are adjusted using the Adam update rule given by

$$\theta_t = \theta_{t-1} - \eta_t \frac{m_t}{\sqrt{v_t} + \hat{\epsilon}}, \tag{4.9}$$

where $\hat{\epsilon}$ corresponds to a small constant for numerical stability.

- **Learning rate**. The learning rate parameter controls the step size for which the weights of a model are updated regarding the loss gradient. The lower its value is, the slower the convergence is but it is ensured that it is not missed any local minimum.

The hyperparameters values chosen are defined in Section 5.4.

4.4.2 Model Evaluation

As referred in section 4.4, the CNN was trained for a fixed number of epochs. However, during training, the model was validated at every 10 epochs, in order to select the best model configuration. Figure 4.3 illustrates how each model was evaluated to obtain the best possible one.

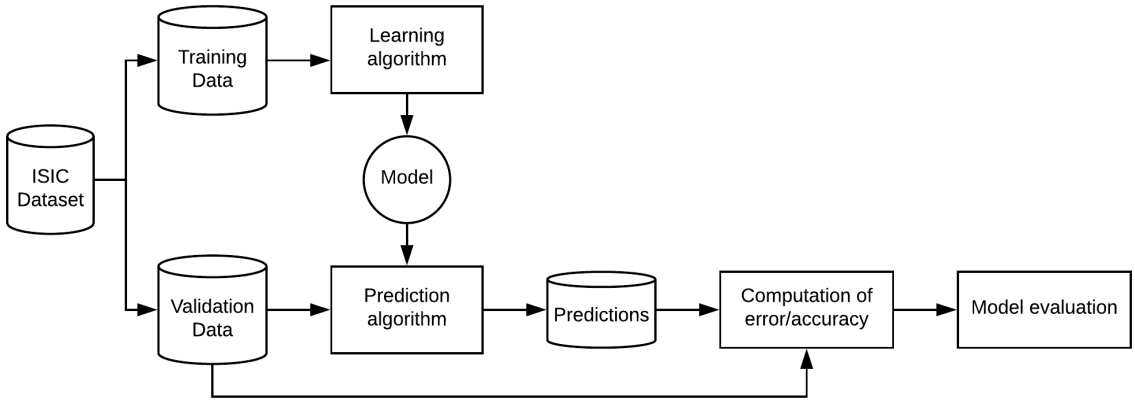


Figure 4.3: Model evaluation procedure.

As depicted in Figure 4.3, the model is learned from the training data through a learning algorithm. To evaluate how well the model can perform, a predicted algorithm, using the learned weights of the model, generates predictions from a new set of data (validation data) that was not used during the training procedure. Essentially, the new set of data allows the validation of the model's performance in order to understand if the model is able to generalize to new inputs. After that, the model's performance is evaluated by using a metric that compares the predictions with the desired outputs of the validation data. The metric chosen was the Balanced Error (BE) which is given by

$$BE = 1 - \frac{SE + SP}{2}, \tag{4.10}$$

where SE is the sensitivity (percentage of correctly classified foreground pixels) and SP is the specificity (percentage of correctly classified background pixels).

In the next chapter, the results of this metric are presented using the validation data as the input of the system. After obtaining for each experiment the model that achieves the lowest BE, these models are used to detect pigment network in the test data.

Chapter 5

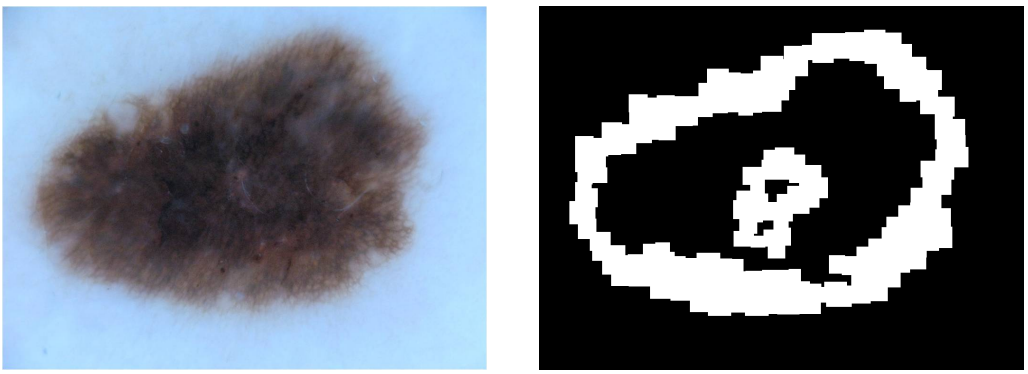
Implementation and Results

This section presents the results obtained for the proposed system as well as some implementation aspects.

5.1 Dataset

The International Skin Imaging Collaboration (ISIC) 2017 Challenge [83] dataset [20] for Skin Lesion Towards Melanoma Detection was used throughout this work .

This dataset of dermoscopy images was very recently made publicly available and it contains 2750 RGB images which are pre-partitioned into 2000 training images, 150 validation images and 600 test images. Furthermore, it is also provided corresponding superpixel masks and superpixel-mapped expert annotation of the presence or absence of pigment network. These superpixel masks were converted to binary segmentation masks, where white denotes the pixels where pigment network is present (foreground) and black corresponds to the background. Figure 5.1 depicts an example where pigment network is present and its corresponding manual segmentation in a binary mask.



(a) (b)

Figure 5.1: (a) Original image with pigment network; (b) Binary segmentation mask (extracted from the ISIC 2017 dataset [20]).

The dataset is composed by malignant and benign lesions, where pigment network may or may not be present. Table 5.1 shows the number of dermoscopy images that contain pigment network for each subset.

It should be noted that only recently this database was made publicly available, providing a large number of dermoscopy images with superpixel-level annotations performed by expert dermatologists,

Subset	Pigment Network	Non-pigment Network	Total images
Training	1131	869	2000
Validation	64	86	150
Test	333	267	600

Table 5.1: ISIC 2017 Dataset [20] distribution

which is why this thesis could only be developed in the present days.

5.2 Performance metrics

This section addresses the procedures used to quantitatively evaluate the performance of the proposed system for detection of pigment network.

The proposed network detection system generates a binary segmented image that is compared with the groundtruth segmentation using pixel-based statistics [1]. The pixels $p(x, y)$ can be classified as True Positives (TP), False Positives (FP), True Negatives (TN) or False Negatives (FN), where

- **#TP**: Number of correctly detected as pigment network pixels.
- **#FP**: Number of wrongly detected as pigment network pixels.
- **#TN**: Number of correctly undetected pixels.
- **#FN**: Number of wrongly undetected pixels.

The explanation of these measures can be organized using the concept of confusion matrix, which illustrates the number of correct predictions in relation to the expected label. In these matrices, each column denotes the label of a predicted class, while each row represents the actual class diagnosed by the medical expert, as shown in Table 5.2.

		Predicted Class	
		Non-pigment Network (0)	Pigment Network (1)
Actual Class	Non-pigment Network (0)	TN	FP
	Pigment Network (1)	FN	TP

Table 5.2: Confusion matrix

The performance of the proposed system is assessed by obtaining the number of TP, TN, FN and FP in test images and by computing the Sensitivity (SE), Specificity (SP), Dice Score , Accuracy (ACC) and Balanced Accuracy (BA) as follows:

- **Sensitivity**, also known as recall, measures the proportion of actual positives that are correctly identified.

$$SE = \frac{\#TP}{\#TP + \#FN} \tag{5.1}$$

- **Specificity** measures the proportion of actual negatives that are correctly identified.

$$SP = \frac{\#TN}{\#TN + \#FP} \tag{5.2}$$

[1] It could be possible to assess the performance of the detection algorithm using the superpixel-wise classification instead of the pixel-wise classification. However, this is not the most natural way of evaluating the algorithm since the CNN architecture classifies all the image pixels in an independent way.

- **Dice coefficient** compares the pixel-wise agreement between the groundtruth and its corresponding predicted segmentation mask, measuring how similar these objects are.

$$Dice = \frac{2\,\#TP}{2\,\#TP + \#FP + \#FN}. \tag{5.3}$$

- **Accuracy** measures the proportion of correct predictions.

$$ACC = \frac{\#TP + \#TN}{\#TP + \#TN + \#FP + \#FN} \tag{5.4}$$

- **Balanced accuracy** is computed as the average of the sensitivity and specificity.

$$BA = \frac{SE + SP}{2} \tag{5.5}$$

During the training procedure, the performance metric used (balanced error) was computed using the validation set to obtain the best model configuration. The test set was only used to make the final assessment of the proposed system after the best model configuration was obtained.

5.3 Implementation aspects

The proposed method was implemented in Python 3.6.4 based on Tensorflow 1.8.0. Since it involves many complex operations with some of which computationally expensive either in processing time or memory requirements, a GPU was required to train and evaluate the models, using CUDA libraries developed by NVIDIA to compile and perform the parallel computations on the GPU. For this purpose, it was used a personal computer with an Intel Core i7 processor with 8GB of RAM and an NVIDIA GeForce GT 740M with 2GB of memory. Even though it has a compute capability of only 3.5, it highly reduces the training time compared to only using a CPU. Instituto Superior Técnico (IST) provided a computer with Intel Core i7 processor with 8GB of RAM (without GPU) where some of the training procedures were undertaken. The training time of the proposed system using only CPU took, on average, 53 hours. When using the personal GPU, the training time took, on average, 14 hours (the training time for all experiments was between 13 and 15 hours). Thus, the hardware available has restricted the number of experiments performed and the number of configurations obtained.

5.4 System optimization

The model was trained with a batch size of 2, not only due to memory limitations but also, because a small batch size brings a higher generalization ability and a faster overall training procedure. Regarding the epochs, it was chosen a number of 150 epochs. This choice was based on the examination of the behaviour of the balanced error vs the number of epochs plots using the validation data as the input of the system. It was found that the minimum balanced error was reached before 150 epochs. The optimization process during training was made using a very small learning rate of $\eta = 10^{-4}$ to guarantee a reliable training procedure. Regarding the optimizer hyperparameters, it was used the default values of $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\hat{\epsilon} = 10^{-8}$.

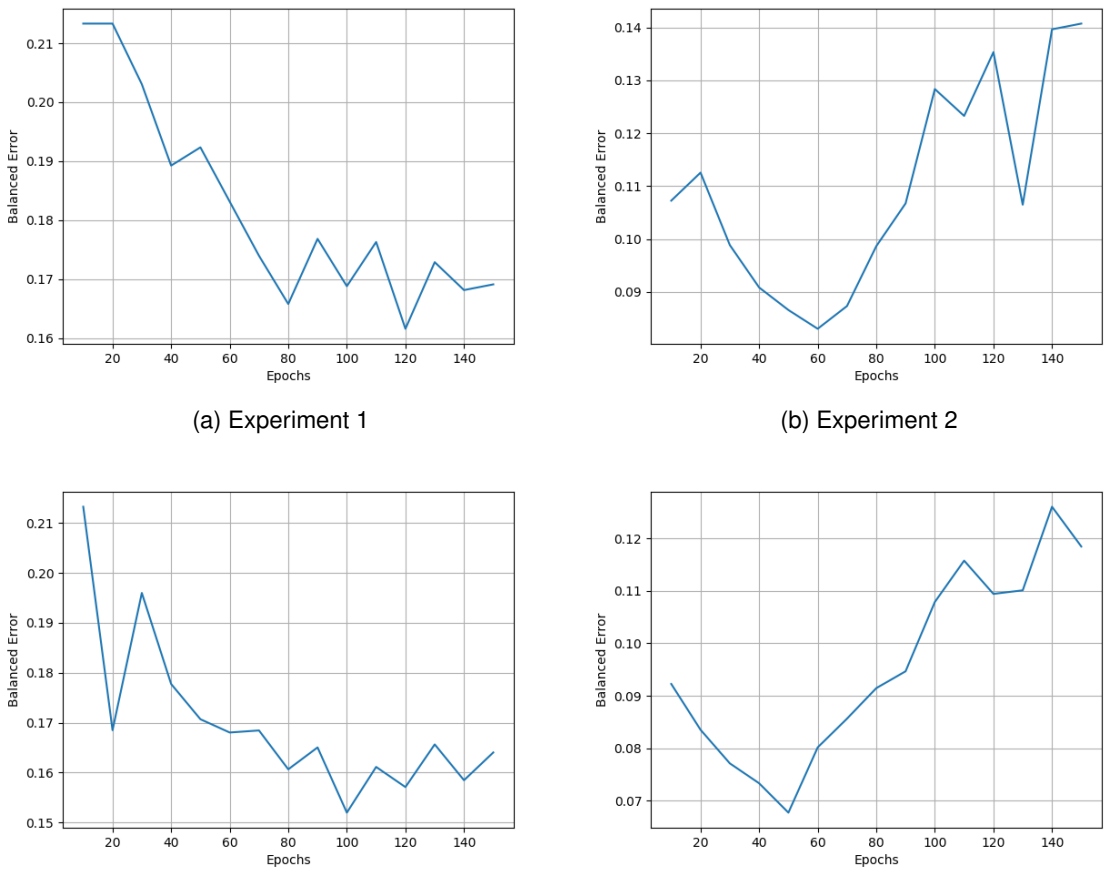
As explained in 4.4.1, to tackle the imbalanced data issue, it was used the weighted cross-entropy loss function that introduces a weight as a multiplicative coefficient for the positive class. The value of

this weight was obtained inverting the value of the foreground pixels/total pixels ratio using the training set images. Thus, it was found that only 4.5% of the total number of pixels (all images included) corresponded to foreground pixels, which led to a weight of 22.16.

After several preliminary experiments performed, the following ones were selected to be discussed:

- **Experiment 1**. The model was trained using greyscale dermoscopy images as input and cross-entropy as loss function.
- **Experiment 2**. The model was trained using greyscale dermoscopy images as input and weighted cross-entropy as loss function.
- **Experiment 3**. The model was trained using RGB dermoscopy images as input and cross-entropy as loss function.
- **Experiment 4**. The model was trained using RGB dermoscopy images as input and weighted cross-entropy as loss function.

In section 4.4.2, it was stated that the model parameters were saved every 10 epochs and evaluated using the validation set to obtain the best model configuration. Furthermore, it was introduced that the assessment of the best model would be made using the balanced error as the metric and the validation data as the input of the system. Figure 5.2 shows the results of this metric applied to each one of the models' outputs resulting from each one of the four experiments.



(a) Experiment 1

(b) Experiment 2

(c) Experiment 3

(d) Experiment 4

Figure 5.2: Balanced error over epochs for all the tested configurations using validation data.

The results obtained in the validation step are summed up in Table 5.3. The table specifies the minimum balanced error obtained for each one of the experiments and at which epoch it was reached.

Experiment	Color	Loss function	Minimum balanced error	Epoch
1	Greyscale	CE	16.2%	120
2	Greyscale	WCE	8.3%	60
3	RGB	CE	15.2%	100
4	RGB	WCE	**6.8%**	50

Table 5.3: System performance in the validation set

The results presented in Table 5.3 suggest that using RGB dermoscopy images instead of greyscale ones as input guarantees slightly better results. Furthermore, it also suggests that using the weighted cross-entropy loss function highly improves the results comparing to using the standard cross-entropy as loss function. However, the validation set is too small to be able to draw conclusions from these validation results. Thus, the model parameters obtained at the epoch where the minimum balanced error was achieved were used to assess the performance of each model using the dermoscopy images belonging to the test set as the input of the proposed system.

5.5 Results

The performance assessment of each one of the models was obtained by comparing every automatically generated binary mask, i.e. the output of the proposed system, with its corresponding groundtruth mask. For this purpose, it was used the test set as the input of the proposed system. The statistical results for the network detection system at a pixel level are presented in Table 5.4.

Exp.	Color	Loss function	SE	SP	Dice	ACC	BA
1	Greyscale	CE	60.6%	**97.3%**	26.4%	**95.1%**	79.0%
2	Greyscale	WCE	**93.9%**	84.0%	23.2%	84.5%	89.0%
3	RGB	CE	65.2%	96.9%	**31.6%**	**95.1%**	81.1%
4	RGB	WCE	90.5%	88.9%	28.8%	89.0%	**89.7%**

Table 5.4: System performance in the test set

Table 5.4 shows that, as seen in the validation step, the proposed system obtains slightly better results when training the model with RGB images. The balanced accuracy slightly increases, however, the dice coefficient increases $\sim 5\%$, which may indicate that the generated binary masks get more similar to the groundtruth by using RGB images as the input of the proposed system during training. Regarding the loss function, it can be seen that the results highly improve when using the weighted cross-entropy instead of the standard cross-entropy. Even though specificity decreases $\sim 10\%$, sensitivity increases more than 25%, which implies that more pixels with pigment network are detected. Using the WCE as loss function results in obtaining more pixels classified as *with pigment network* than the ones which actually contain it, but on the other hand, the proposed system gets much better in the task for which it was designed, i.e. detecting pigment network in dermoscopy images. The balanced accuracy increases by $\sim 10\%$ which proves the benefit of using this weighted loss function. Accuracy decreases when choosing the WCE as loss function over CE. However, accuracy is not a relevant metric when dealing with segmentation tasks, mainly because it is not sensitive to the fact that the classes are imbalanced. For instance, in Section 5.4 it was found that only 4.5% of the total number of pixels of the dermoscopy training images corresponded to foreground pixels. This means that if one used them as the input of the

proposed system and the output generated masks showed only background pixels, the accuracy of the model would still be 95.5%, when it did not detect pigment network as it was designed for.

To understand how the number of epochs affects the results, Table 5.5 shows the results obtained using the successive saved models during the learning procedure. For this purpose, it was used the model using RGB images as input and the WCE as loss function.

Epoch	Sensitivity	Specificity	Dice Coefficient	Accuracy
10	**96.1%**	81.5%	22.4%	82.3%
20	95.3%	83.3%	25.3%	84.0%
30	95.1%	83.9%	26.0%	84.5%
40	95.3%	84.2%	24.7%	84.9%
50	90.5%	88.9%	28.8%	89.0%
60	86.6%	91.0%	30.3%	90.8%
70	84.1%	91.4%	28.7%	91.0%
80	81.4%	92.2%	**33.6%**	91.6%
90	80.4%	92.7%	32.3%	92.0%
100	75.1%	94.1%	31.7%	93.0%
110	71.3%	**94.9%**	32.8%	**93.6%**
120	77.8%	93.3%	31.8%	92.5%
130	76.8%	93.6%	29.2%	92.7%
140	72.7%	94.1%	32.9%	93.1%
150	75.4%	93.4%	32.0%	92.6%

Table 5.5: Results over training for Experiment 4

By inspecting Table 5.5, the first noticeable thing is that using the balanced error in the validation step was a good metric to obtain the model that would guarantee the best results. Furthermore, by inspecting the results over the epochs, one may reckon that during training the model loses its capacity to detect pigment network, leading the results to approach the ones obtained when using the cross-entropy as loss function. This means that even though using the weight as a multiplicative coefficient of the minority class (foreground pixels) helps to detect more pixels actually containing pigment network, it does not emphasize the minority class enough compared to the majority class (background pixels), which may imply that the model is not learning equally from both classes and other ways of dealing with imbalanced data should be pursued. The Dice coefficient increases over the number of epochs mainly due to the decrease in the number of FP identified. Even though the number of TP decreases, the reduction in the number of FP represents a much higher number of pixels that stop being wrongly classified, which is why the increase of the Dice coefficient does not entail much relevance.

Nevertheless, the proposed system achieves an interesting performance on the task for which it was designed, achieving a $SE = 90.5\%$ and $SP = 88.9\%$.

Figures 5.3 and 5.4 show examples of satisfactory and poor segmentations performed by the proposed system in the detection of pigment network. In these figures, the left column corresponds to the original image, the middle column is the output of the proposed system and the right column corresponds to the groundtruth binary mask. Figure 5.3 shows that even when hair is present in the dermoscopic image, the proposed system can still perform a satisfactory detection of pigment network. Furthermore, it also explains the decrease of the specificity when using the WCE as loss function instead of the CE; the automatically generated masks show that the regions with pigment network are slightly wider than the manually segmented regions of pigment network performed by the experts. Figure 5.4 shows that when there is no sufficient contrast between the lesion area (where pigment network is located) and the background, the proposed system cannot perform a satisfactory segmentation of pigment network. However, it should be noted that the cases where the proposed system underperformed were rare.

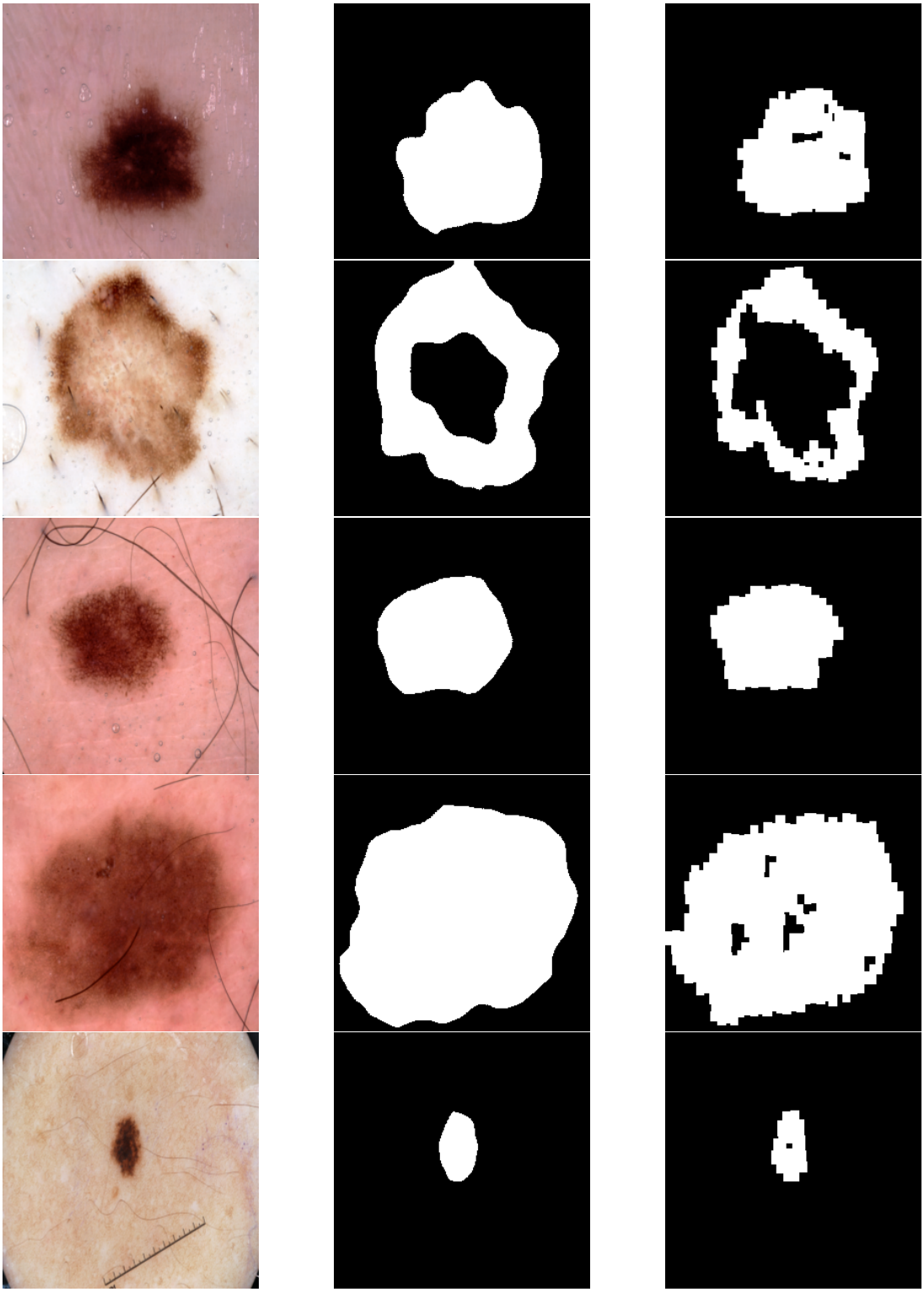


Figure 5.3: Satisfactory segmentation results: original image (left), automatic segmentation (center) and groundtruth segmentation (right).

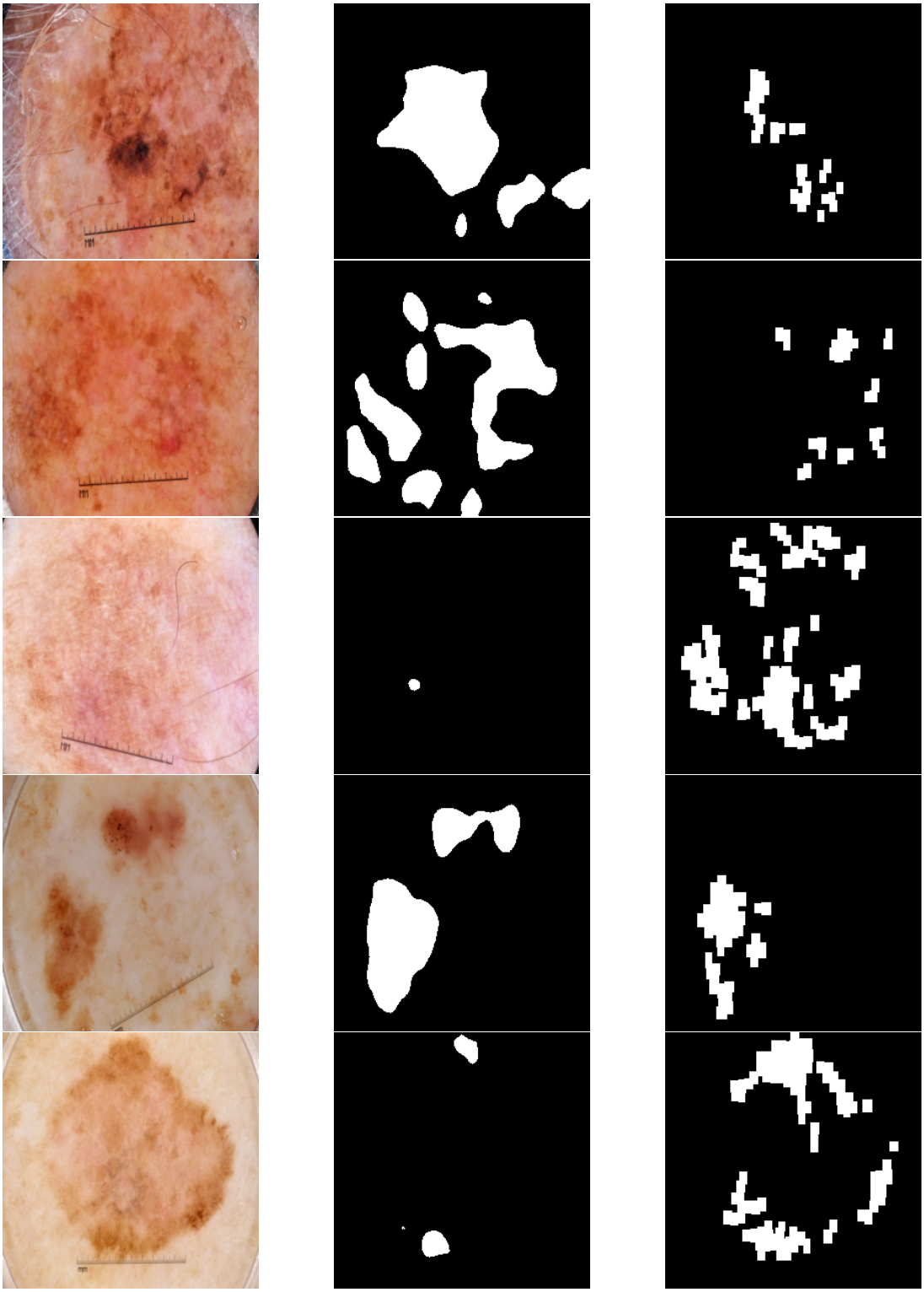


Figure 5.4: Poor segmentation results: original image (left), automatic segmentation (center) and groundtruth segmentation (right).

Chapter 6

Conclusion

The aim of this thesis was the development of an automatic system for the detection of pigment network in dermoscopy images, based on a deep learning approach. This work was only possible thanks to the publication of the ISIC 2017 dataset, which contained 2750 dermoscopy images with superpixel-mapped annotations performed by expert dermatologists.

The proposed system is comprised of two identical pre-processing modules, a CNN architecture module and a training module. The pre-processing modules are responsible for resizing and normalizing the input images as well as converting them to greyscale for some experiments. One of the modules is used to pre-process the training images before the training procedure, while the other one is used to pre-process the images before they are forwarded through the proposed system to obtain the corresponding segmentation of pigment network. The CNN architecture that was used is based on the U-Net, which was specifically designed for biomedical image segmentation tasks, in order to obtain automatically generated binary masks. The training of the model is performed from scratch, using the training images belonging to the ISIC dataset. During the training procedure, the dataset is divided into small mini-batches (2 images) as a matter of efficiency and memory limitations. Furthermore, it was used a weighted loss function (weighted cross-entropy) to deal with the imbalanced data, which introduces a multiplicative weight for the minority class (pixels with pigment network). To obtain the best model configuration, the model parameters were validated every 10 epochs using the balanced error in the validation step and the configuration that achieved the best score was chosen.

The quantitative evaluation of the proposed system was performed by comparing each generated binary mask with its corresponding groundtruth mask, using as input the test images of the ISIC dataset. In general, it was found that using RGB images as the input of the proposed system and during the training procedure slightly improves the results. Regarding the loss function, it was found that using the WCE helps to deal with the imbalanced data, guaranteeing highly better results, mainly on the detection of foreground pixels.

The results obtained were quite promising, proving that deep learning methods can help medical experts to make faster and more accurate diagnoses. The proposed system achieved a $SE = 90.5\%$ and a $SP = 88.9\%$ by considering RGB images as the input of the system as well as the WCE as the loss function. These results prove that the developed system is a useful tool for the automatic detection of pigment network in dermoscopy images.

Despite the encouraging results obtained for the developed system, the proposed solution may benefit from the following suggestions:

- **Original depth of the channels**. Using the full depth of the channels at each stage of the network as proposed by the authors of U-Net [77] would result in a higher number of features learned by the

network, which could improve the performance of the system. However, this should be combined with data augmentation, so that the model does not end up fitting the training data too well.

- **Data augmentation**. The performance of a deep CNN highly depends on the data that it has been trained with. The CNN architecture used has a very high number of parameters, which means that the model should be fed with a proportional amount of examples during training. For instance, by applying minor alterations to the existing data set, such as flips, translations or rotations, the model would be able detect foreground pixels even if they are placed in different orientations, which would make the model invariant to these alterations. This would also help to prevent overfitting, which is a common problem when the model is exposed to too few examples, learning patterns that do not generalize well to new data.

- **Transfer Learning**. Instead of training the entire CNN from scratch, it could be positive to try transfer learning by using pre-trained networks on very large datasets such as the Dermnet dataset [19] in order to initialize the weights safely.

- **Dropout**. This regularization technique could be used to reduce overfitting. At every iteration, it randomly selects some nodes, which are ignored during training. This means that their contribution is temporally removed. Thus, the other neurons have to handle the representation required to make the predictions for the missing nodes. The effect is that the network gets less sensitive to the specific weights of neurons, which makes the network capable of better generalization [84].

- **Detect other structures**. This CNN architecture could also be used to obtain models capable of detecting other dermoscopic structures, such as streaks, negative networks and milia-like cysts.

- **Distinguish between typical and atypical pigment network**. As mentioned in section 2.4, the presence of atypical network commonly results in the lesion being classified as a melanoma. Thus, it would be of great importance to distinguish between both types of pigment network, which would increase the value of the proposed system.

Bibliography

[1] "Dermoscopy tutorial." `http://www.dermoscopy.org/atlas/base.htm`. Accessed: 2018-09-11.

[2] A. J. Miller and M. C. Mihm Jr, "Melanoma," *New England Journal of Medicine*, vol. 355, no. 1, pp. 51–65, 2006.

[3] A. W. Kopf, M. Elbaum, and N. Provost, "The use of dermoscopy and digital imaging in the diagnosis of cutaneous malignant melanoma," *Skin Research and Technology*, vol. 3, no. 1, pp. 1–7, 1997.

[4] C. M. Grin, A. W. Kopf, B. Welkovich, R. S. Bart, and M. J. Levenstein, "Accuracy in the clinical diagnosis of malignant melanoma," *Archives of dermatology*, vol. 126, no. 6, pp. 763–766, 1990.

[5] W. Stolz, "ABCD rule of dermatoscopy: a new practical method for early recognition of malignant melanoma," *Eur. J. Dermatol.*, vol. 4, pp. 521–527, 1994.

[6] G. Argenziano, G. Fabbrocini, P. Carli, V. De Giorgi, E. Sammarco, and M. Delfino, "Epiluminescence microscopy for the diagnosis of doubtful melanocytic skin lesions: comparison of the abcd rule of dermatoscopy and a new 7-point checklist based on pattern analysis," *Archives of dermatology*, vol. 134, no. 12, pp. 1563–1570, 1998.

[7] G. Argenziano, H. P. Soyer, S. Chimenti, R. Talamini, R. Corona, F. Sera, M. Binder, L. Cerroni, G. De Rosa, G. Ferrara, *et al.*, "Dermoscopy of pigmented skin lesions: results of a consensus meeting via the internet," *Journal of the American Academy of Dermatology*, vol. 48, no. 5, pp. 679–693, 2003.

[8] M. G. Fleming, C. Steger, J. Zhang, J. Gao, A. B. Cognetta, C. R. Dyer, *et al.*, "Techniques for a structural analysis of dermatoscopic imagery," *Computerized medical imaging and graphics*, vol. 22, no. 5, pp. 375–389, 1998.

[9] M. Anantha, R. H. Moss, and W. V. Stoecker, "Detection of pigment network in dermatoscopy images using texture analysis," *Computerized Medical Imaging and Graphics*, vol. 28, no. 5, pp. 225–234, 2004.

[10] G. Betta, G. Di Leo, G. Fabbrocini, A. Paolillo, and P. Sommella, "Dermoscopic image-analysis system: estimation of atypical pigment network and atypical vascular pattern," in *Medical Measurement and Applications, 2006. MeMea 2006. IEEE International Workshop on*, pp. 63–67, IEEE, 2006.

[11] M. Sadeghi, M. Razmara, T. K. Lee, and M. S. Atkins, "A novel method for detection of pigment network in dermoscopic images using graphs," *Computerized Medical Imaging and Graphics*, vol. 35, no. 2, pp. 137–143, 2011.

[12] P. Wighton, T. K. Lee, H. Lui, D. I. McLean, and M. S. Atkins, "Generalizing common tasks in automated skin lesion diagnosis," *IEEE Transactions on Information Technology inBiomedicine*, vol. 15, no. 4, p. 622, 2011.

[13] C. Barata, J. S. Marques, and J. Rozeira, "A system for the detection of pigment network in dermoscopy images using directional filters," *IEEE transactions on biomedical engineering*, vol. 59, no. 10, pp. 2744–2754, 2012.

[14] J. L. G. Arroyo and B. G. Zapirain, "Detection of pigment network in dermoscopy images using supervised machine learning and structural analysis," *Computers in biology and medicine*, vol. 44, pp. 144–157, 2014.

[15] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.

[16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[17] A. C. F. Barata, E. M. Celebi, and J. Marques, "A survey of feature extraction in dermoscopy image analysis of skin cancer," *IEEE Journal of Biomedical and Health Informatics*, 2018.

[18] J. Kawahara and G. Hamarneh, "Fully convolutional neural networks to detect clinical dermoscopic features," *IEEE Journal of Biomedical and Health Informatics*, 2018.

[19] "Dermnet." `http://www.dermnet.com/`. Accessed: 2018-10-10.

[20] "Isic archive." `https://www.isic-archive.com/`.

[21] "Dermoscopedia." `https://dermoscopedia.org/Main_Page`. Accessed: 2018-09-18.

[22] "Dermnet nz." `https://www.dermnetnz.org/`. Accessed: 2018-09-18.

[23] R. P. Braun, H. S. Rabinovitz, M. Oliviero, A. W. Kopf, and J.-H. Saurat, "Dermoscopy of pigmented skin lesions," *Journal of the American Academy of Dermatology*, vol. 52, no. 1, pp. 109–121, 2005.

[24] H. Kittler, H. Pehamberger, K. Wolff, and M. Binder, "Diagnostic accuracy of dermoscopy," *The lancet oncology*, vol. 3, no. 3, pp. 159–165, 2002.

[25] H. Pehamberger, A. Steiner, and K. Wolff, "In vivo epiluminescence microscopy of pigmented skin lesions. i. pattern analysis of pigmented skin lesions," *Journal of the American Academy of Dermatology*, vol. 17, no. 4, pp. 571–583, 1987.

[26] S. W. Menzies, K. A. Crotty, C. Ingvar, and W. H. McCarthy, *An atlas of surface microscopy of pigmented skin lesions: dermoscopy*. McGraw-Hill Sydney, Australia, 2003.

[27] J. Krischer, A. Skaria, J. Guillod, E. Lemonnier, D. Salomon, R. Braun, and J.-H. Saurat, "Epiluminescent light microscopy of melanocytic lesions after dermoepidermal split," *Dermatology*, vol. 195, no. 2, pp. 108–111, 1997.

[28] C. Grana, R. Cucchiara, G. Pellacani, and S. Seidenari, "Line detection and texture characterization of network patterns," 2006.

[29] G. Di Leo, A. Paolillo, P. Sommella, and G. Fabbrocini, "Automatic diagnosis of melanoma: a software system based on the 7-point check-list," in *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pp. 1–10, IEEE, 2010.

[30] B. Shrestha, J. Bishop, K. Kam, X. Chen, R. H. Moss, W. V. Stoecker, S. Umbaugh, R. J. Stanley, M. E. Celebi, A. A. Marghoob, *et al.*, "Detection of atypical texture features in early malignant melanoma," *Skin Research and Technology*, vol. 16, no. 1, pp. 60–65, 2010.

[31] M. Sadeghi, M. Razmara, P. Wighton, T. K. Lee, and M. S. Atkins, "Modeling the dermoscopic structure pigment network using a clinically inspired feature set," in *International Workshop on Medical Imaging and Virtual Reality*, pp. 467–474, Springer, 2010.

[32] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[33] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[34] D. Hebbs, "The organization of behavior," *Wiely and Sons, New York, NY, USA*, 1949.

[35] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[36] A. G. Ivakhnenko, "Polynomial theory of complex systems," *IEEE transactions on Systems, Man, and Cybernetics*, no. 4, pp. 364–378, 1971.

[37] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*, pp. 267–285, Springer, 1982.

[38] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986.

[39] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[40] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen netzen," *Diploma, Technische Universität München*, vol. 91, no. 1, 1991.

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[44] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[46] J. Sánchez and F. Perronnin, "High-dimensional signature compression for large-scale image classification," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 1665–1672, IEEE, 2011.

[47] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014.

[48] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[49] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1 ed., 1997.

[50] "CS231n: Convolutional neural networks for visual recognition." Standford University [Online]. Available: `http://cs231n.github.io/`. Accessed: 2018-08-30.

[51] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[52] A. E. Faghfouri and M. B. Frish, "Robust discrimination of human footsteps using seismic signals," in *Unattended Ground, Sea, and Air Sensor Technologies and Applications XIII*, vol. 8046, p. 80460D, International Society for Optics and Photonics, 2011.

[53] I. N. Da Silva, D. H. Spatti, R. A. Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves, "Artificial neural networks," *Cham: Springer International Publishing*, 2017.

[54] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[55] S. Srinivas, R. K. Sarvadevabhatla, K. R. Mopuri, N. Prabhu, S. S. Kruthiventi, and R. V. Babu, "A taxonomy of deep convolutional neural nets for computer vision," *Frontiers in Robotics and AI*, vol. 2, p. 36, 2016.

[56] M. Peemen, B. Mesman, and H. Corporaal, "Efficiency optimization of trainable feature extractors for a consumer platform," in *International Conference on Advanced Concepts for Intelligent Vision Systems*, pp. 293–304, Springer, 2011.

[57] S. Albelwi and A. Mahmood, "A framework for designing the architectures of deep convolutional neural networks," *Entropy*, vol. 19, no. 6, p. 242, 2017.

[58] N. M. Zaitoun and M. J. Aqel, "Survey on image segmentation techniques," *Procedia Computer Science*, vol. 65, pp. 797–806, 2015.

[59] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[60] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, pp. 1520–1528, 2015.

[61] S. L. Hijazi, R. Kumar, and C. Rowen, "Using convolutional neural networks for image recognition," 2015.

[62] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial Intelligence and Statistics*, pp. 562–570, 2015.

[63] "Understanding convolutional layers in convolutional neural networks (CNNs)." Available: `http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html`. Accessed: 2018-09-03.

[64] S. Soleymani, A. Dabouei, H. Kazemi, J. Dawson, and N. M. Nasrabadi, “Multi-level feature abstraction from convolutional neural networks for multimodal biometric identification,” *arXiv preprint arXiv:1807.01332*, 2018.

[65] S. A. Siddiqui, A. Salman, M. I. Malik, F. Shafait, A. Mian, M. R. Shortis, E. S. Harvey, and H. editor: Howard Browman, “Automatic fish species classification in underwater videos: exploiting pre-trained deep neural network models to compensate for limited labelled data,” *ICES Journal of Marine Science*, vol. 75, no. 1, pp. 374–389, 2017.

[66] M. Peemen, B. Mesman, and H. Corporaal, “Efficiency optimization of trainable feature extractors for a consumer platform,” in *International Conference on Advanced Concepts for Intelligent Vision Systems*, pp. 293–304, Springer, 2011.

[67] V. Dumoulin and F. Visin, “A guide to convolution arithmetic for deep learning,” *arXiv preprint arXiv:1603.07285*, 2016.

[68] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, L. Wang, G. Wang, *et al.*, “Recent advances in convolutional neural networks,” *arXiv preprint arXiv:1512.07108*, 2015.

[69] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” *arXiv preprint arXiv:1511.07122*, 2015.

[70] M. D. Zeiler, G. W. Taylor, and R. Fergus, “Adaptive deconvolutional networks for mid and high level feature learning,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2018–2025, IEEE, 2011.

[71] D. Eigen and R. Fergus, “Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2650–2658, 2015.

[72] W. Liu, A. Rabinovich, and A. C. Berg, “Parsenet: Looking wider to see better,” *arXiv preprint arXiv:1506.04579*, 2015.

[73] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, “Conditional random fields as recurrent neural networks,” in *Proceedings of the IEEE international conference on computer vision*, pp. 1529–1537, 2015.

[74] G. Ghiasi and C. C. Fowlkes, “Laplacian pyramid reconstruction and refinement for semantic segmentation,” in *European Conference on Computer Vision*, pp. 519–534, Springer, 2016.

[75] G. Lin, C. Shen, A. Van Den Hengel, and I. Reid, “Exploring context with deep structured models for semantic segmentation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1352–1366, 2018.

[76] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, “Attention to scale: Scale-aware semantic image segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3640–3649, 2016.

[77] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[78] R. C. Gonzalez, R. E. Woods, *et al.*, “Digital Image Processing,” 2002.

[79] "Normalized RGB." `http://aishack.in/tutorials/normalized-rgb/`. Accessed: 2018-08-15.

[80] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[81] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*, pp. 437–478, Springer, 2012.

[82] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[83] N. C. Codella, D. Gutman, M. E. Celebi, B. Helba, M. A. Marchetti, S. W. Dusza, A. Kalloo, K. Liopyris, N. Mishra, H. Kittler, *et al.*, "Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic)," in *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*, pp. 168–172, IEEE, 2018.

[84] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.