UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

DEPARTAMENTO DE COMUNICACIONES

FINAL DEGREE PROJECT:

AUTOMATIC DETECTION OF METASTATIC TISSUE IN LYMPH NODE SECTIONS OF BREAST CANCER PATIENTS USING CONVOLUTIONAL NEURAL NETWORKS

Thesis submitted for the Bachelor's Degree in Telecommunication Technologies and Services Engineering in the Universitat Politècnica de València.
Academic year 2019-2020

Author: Javier Abad Martínez

Supervised by:

Prof. Dr. Valery Naranjo Ornedo
and
Dr. Adrián Colomer Granero

Valencia, July 2020

Abstract

Breast cancer is currently the leading cause of death due to cancer in women, after lung cancer. For its diagnosis and staging, detection of metastatic tissue in axillary lymph nodes is occasionally used, since lymphatic spread is the main prognostic factor, especially in early stages. However, the pathologist's work in this diagnosis is considerably complex and tedious, so the need to automate this process arises. In the present work, we propose models based on Convolutional Neural Networks for the identification of metastasis in sections of axillary lymph nodes stained in H&E. Models developed from scratch and models based on fine-tuning of pre-trained networks are exposed, some of them with performances good enough for their application in the clinical reality of hospitals. In addition, these state-of-the-art techniques are compared to others based on traditional machine learning feature extractors and classifiers, identifying the advantages and disadvantages of each approach. Finally, a Grad-CAM study is carried out that serves as a tool for the comprehension of deep learning, since it allows the pathologist to visualize the activations of the proposed networks.

Keywords: machine learning, deep learning, convolutional neural network, breast cancer diagnosis, axillary lymph node, medical imaging, computer vision, fine-tuning, Grad-CAM

Resumen

El cáncer de mama se impone en la actualidad como la principal causa de muerte por cáncer en mujeres, después del cáncer de pulmón. Para su diagnóstico y estadificación, en ocasiones se recurre a la detección de tejido metastásico en ganglios linfáticos axilares, ya que la diseminación linfática es el principal factor pronóstico, sobre todo en estadíos iniciales. Sin embargo, la labor del patólogo en este diagnóstico es considerablemente compleja y tediosa, por lo que nace la necesidad de automatizar este proceso. En el presente trabajo, se proponen modelos basados en Redes Neuronales Convolucionales para la identificación de metástasis en secciones de ganglios linfáticos axilares tintados en H&E. Se exponen modelos de redes diseñadas específicamente para este objetivo y modelos basados en ajuste fino de redes pre-entrenadas, algunos de ellos con rendimientos suficientemente buenos para su aplicación en la realidad clínica de los hospitales. Además, se comparan estas técnicas de vanguardia con otras basadas en extractores de características y clasificadores tradicionales de aprendizaje automático, identificando las ventajas y desventajas de cada planteamiento. Finalmente, se realiza un estudio Grad-CAM que sirve como herramienta para la comprensión del aprendizaje profundo, ya que permite al patólogo la visualización de las activaciones de las redes propuestas.

Palabras clave: aprendizaje automático, aprendizaje profundo, red neuronal convolucional, diagnóstico de cáncer de mama, ganglio linfático axilar, imagen médica, visión artificial, ajuste fino, Grad-CAM

Resum

El càncer de mama s'imposa en l'actualitat com la principal causa de mort per càncer en dones, després del càncer de pulmó. Per al seu diagnòstic i estadificació, a vegades es recorre a la detecció de teixit metastàtic en ganglis limfàtics axil·lars, ja que la disseminació limfàtica és el principal factor pronòstic, sobretot en estadis inicials. No obstant això, la labor del patòleg en aquest diagnòstic és considerablement complexa i tediosa, pel que naix la necessitat d'automatitzar aquest procés. En el present treball, es proposen models basats en Xarxes Neuronals Convolucionals per a la identificació de metàstasi en seccions de ganglis limfàtics axil·lars tintats en H&E. S'exposen models de xarxes dissenyades específicament per a aquest objectiu i models basats en ajust fi de xarxes pre-entrenades, alguns d'ells amb rendiments prou bons per a la seua aplicació en la realitat clínica dels hospitals. A més, es comparen estes tècniques d'avantguarda amb altres basades en extractors de característiques i classificadors tradicionals d'aprenentatge automàtic, identificant els avantatges i desavantatges de cada plantejament. Finalment, es realitza un estudi Grad-CAM que serveix com a eina per a la comprensió de l'aprenentatge profund, ja que permet al patòleg la visualització de les activacions de les xarxes proposades.

Paraules clau: aprenentatge automàtic, aprenentatge profund, xarxa neuronal convolucional, diagnòstic de càncer de mama, gangli limfàtic axil·lar, imatge mèdica, visió artificial, ajust fi, Grad-CAM

Contents

Abstract iii

Resumen v

Resum vii

1 Introduction 1

1.1 Motivation . . . 2
1.2 Summary of the methodology followed . . . 2
1.3 Structure of the thesis . . . 4
1.4 Objectives . . . 5
1.4.1 General objectives . . . 5
1.4.2 Sustainable development objectives . . . 5
1.5 State-of-the-art review . . . 6
1.6 Biological background . . . 7
1.6.1 Breast cancer . . . 7
1.6.2 Lymphatic dissemination in breast cancer . . . 8
1.6.3 The haematoxylin and eosin (H&E) stain . . . 8
1.6.4 Automatic detection of metastasis in sentinel axillary lymph nodes (SLN) . . . 9

2 Theoretical framework 11

2.1 Introduction to Artificial Neural Networks . . . 11
2.1.1 The multilayer perceptron . . . 12
2.1.2 Dropout . . . 15
2.1.3 Batch normalization . . . 15
2.1.4 Data Augmentation . . . 16
2.1.5 Training, validation and test . . . 17
2.2 Convolutional Neural Networks . . . 18
2.2.1 Convolutional layer . . . 18
2.2.2 Pooling layer . . . 20
2.2.3 Fully-connected layer . . . 20

2.2.4 Residual blocks . . . 21
2.2.5 Inception blocks . . . 23
2.3 Transfer learning and fine-tuning with CNN pre-trained architectures . . . 25
2.3.1 VGG . . . 26
2.3.2 GoogLeNet . . . 26
2.3.3 ResNet . . . 26
2.3.4 Xception . . . 27
2.3.5 DenseNet . . . 27
2.3.6 MobileNet . . . 28

3 Methodology 29
3.1 Materials, apparatus and procedures . . . 29
3.1.1 Equipment and programming environment . . . 29
3.1.2 Characterization of the database . . . 30
3.1.3 Description of the metrics used . . . 32
3.2 Description of the proposed predictive models . . . 34
3.2.1 Convolutional Neural Networks from scratch . . . 34
3.2.2 CNNs based on fine-tuning of pre-trained architectures . . . 40

4 Results 43
4.1 Convolutional Neural Networks from scratch . . . 43
4.2 CNNs based on fine-tuning of pre-trained architectures . . . 46
4.3 Visualization of the relevant histopathological patterns with Grad-CAM . . . 49

5 Discussion 51
5.1 General discussion . . . 51
5.2 Comparison with other traditional machine learning techniques . . . 54
5.3 Limitations . . . 57

6 Conclusions and future work 59
6.1 Conclusions . . . 59
6.2 Future work . . . 61

Appendices 63

A The gradient descent algorithm 65
A.1 Mathematical background . . . 65
A.2 Gradient descent optimization algorithms . . . 66

B The back-propagation mechanism 69
B.1 Mathematical background . . . 69
B.2 Popular loss functions . . . 70

Bibliography 73

List of Figures

1.1 Histological images of axillary lymph nodes stained with H&E. 8

2.1 Functioning and parameters of a neuron. 12

2.2 Example of a multilayer perceptron with a single hidden layer of four neurons. The input layer is formed by a vector of three components, expressed with three neurons. The output layer has only one neuron, so the network will be used for binary problems. 13

2.3 (a) Standard Neural Network and (b) After applying dropout. 15

2.4 On the right a standard network, on the left a network to which batch normalization has been integrated. β and γ are incorporated as new hyperparameters of the system. Retrieved from *How does Batch Normalization Help Optimization?*, by A. Ilyas et. al, 2018, `https://gradientscience.org/batchnorm/`. Copyright by gradient science. 16

2.5 Double partition of the dataset: training and validation, and test. 17

2.6 Example of convolution with a RGB image. The convolution is computed with a 3x3 kernel with the image section in its three channels (Red, Green, and Blue), and then added together with the bias. The result of the operation will have only one channel. Retrieved from *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*, by S. Saha, 2018, `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53`. Copyright by Towards Data Science. 19

2.7 Exemplification of the operations of max pooling and average pooling. 20

2.8 Example of a Convolutional Neural Network. Two of the convolutional blocks are shown, consisting of a convolution layer, ReLU activation and pooling. A classifier consisting of a flatten, a fully-connect layer and softmax activation for the classification is added. The network classifies the input among various means of transport. Retrieved from *Basics of the Classic CNN*, by C. Churh Chatterjee ,2019 ,`https://towardsdatascience.com/basics-of-the-classic-cnn-a3dce1225add`. Copyright by Towards Data Science. 21

2.9 Residual learning: a building block. From *Deep Residual Learning for Image Recognition* [30], by K. He et. al, 2015. 22

2.10 Identity block. 22

2.11 Convolutional block. 22

2.12 Images of a French bulldog in different positions: (a) lying down (b) standing and (c) in profile. 23

2.13 Inception module, naïve version. From *Going deeper with convolutions* [90], by C. Szegedy et. al, 2015. 24

2.14 Inception module with dimension reductions. From *Going deeper with convolutions* [90], by C. Szegedy et. al, 2015. 24

3.1 Picture of Google's Cloud TPU v3 Pods. Retrieved from *Google Cloud. Cloud TPU*, by Google, `https://cloud.google.com/tpu?hl=es-419`. 30

3.2 Six randomly chosen images from the training set with their respective labels. Class "1" refers to tissue that metastasizes, while Class "0" refers to healthy tissue. Retrieved from *Histopathologic Cancer Detection Dataset*, by B. Veeling, `https://www.kaggle.com/c/histopathologic-cancer-detection/data`. Copyright by Kaggle 31

3.3 Distribution of the classes (cancer and non-cancer) in the training and validation set and the test set. 32

3.4 Confusion matrix with the possible solutions in our problem. 33

3.5 Model from scratch #1. 36

3.6 (a) Model from scratch #2 and (b) Convolutional_block [F1,F2,F3] 37

3.7 Inception_block with F filters. Structure of the inception blocks used in Model from scratch #3. 38

3.8 (a) Model from scratch #3 and (b) ResNet-Inception_block [F1,F2,F3] 39

4.1 Learning curves for (a) Model from scratch #1 (Initial model) (b) Model from scratch #2 (Model with residual blocks) and (c) Model from scratch #3 (ResNet-Inception). 44

4.2 Confusion matrices for the models (a) Model from scratch #1 (Initial model) (b) Model from scratch #2 (Model with residual blocks) and (c) Model from scratch #3 (ResNet-Inception). 45

4.3 ROC curves for the models (a) Model from scratch #1 (Initial model) (b) Model from scratch #2 (Model with residual blocks) and (c) Model from scratch #3 (ResNet-Inception). 46

4.4 Learning curves for the pre-trained models (a) VGG19 (b) Inceptionv3 (c) ResNet50 (d) Xception (e) DenseNet201 (f) MobileNet. 47

4.5 Confusion matrices for the pre-trained models (a) VGG19 (b) Inceptionv3 (c) ResNet50 (d) Xception (e) DenseNet201 (f) MobileNet. 47

4.6 ROC curves for the pre-trained models (a) VGG19 (b) Inceptionv3 (c) ResNet50 (d) Xception (e) DenseNet201 (f) MobileNet. 48

4.7 Grad-CAM of four randomly selected images in our dataset, all correctly classified as cancerous. The classifier model used is the one based on VGG19. The yellow areas are those in which the network identifies patterns to classify the image, therefore, they are supposed to be the sections of the lymph node where the metastatic tissue is located. The same exercise could be done with sections without cancer, observing the histological features identified by the network to rule out the presence of metastases. 49

5.1 Comparison between traditional models and CNNs based on their accuracy and training times (in milliseconds). Circles represent the former, while triangles represent the latter. 56

5.2 Comparison between traditional models and CNNs based on their accuracy and test times (in milliseconds). Circles represent the former, while triangles represent the latter. 56

A.1 Weights update is done by gradient descent, taking steps in the opposite direction to the gradient of the loss function. The optimization process is similar to the exploration of a surface, looking for its minimum. From *Pattern recognition and machine learning* [7], by C. M. Bishop, 2006 65

List of Tables

3.1 Summary of the models based on fine-tuning. The optimizer, the frozen and total layers, the trainable and total parameters and the computational cost in training are indicated.. 42

4.1 Figures of merit for the models from scratch. 45

4.2 Figures of merit for the pre-trained models. 48

5.1 Accuracy scores for all the proposed models. 53

5.2 Parameters – both total and trainable – and computational cost – in hours – of all the proposed models. 54

5.3 Training times, test times and accuracy scores for the five best CNNs and the three best models based on traditional machine learning, obtained in [1] 55

Chapter 1

Introduction

Human intelligence is our ability to reason, solve problems, understand complex ideas, and abstract concepts and learn from our experiences. Intelligence is not limited to an encyclopedic knowledge but includes a deep capacity to understand the environment and to learn from it.

The ambition to create machines capable of emulating human intelligence has been a constant from the mid-twentieth century until today. However, there is a name with which most of the scientific community agrees when it comes to finding the origin: Alan Turing (1912-1954). His well-known publication *Computer Machinery and intelligence* [91] begins the revolution that **artificial intelligence** represents today.

Throughout the 1950s **machine learning** was born as an artificial intelligence application. The principle of it is to design algorithms capable of learning from experience, capable of automating tasks and, in fact, learning from their mistakes and perfecting themselves. Proof of this are the pioneering works of Frank Rosenblatt in the design of the Perceptron in 1958 [70] or the sophisticated models present today, some used in this work.

It is not surprising, therefore, that the applications of machine learning have reached something as sensitive as the medical image. The potential of a machine learning algorithm to learn and to rival with expert pathologists is of particular interest to the scientific community. The medical image is part of a machine learning application known as **Computer Vision**, which aims to design algorithms capable of emulating the behavior of a human eye, being able to classify images based on criteria such as color, shapes or textures. Furthermore, medical image classification is usually a **supervised learning** problem, in which the algorithm is taught with previously classified images, so that it is capable, once it has learned, of classifying unknown (unclassified) images on its own.

Today the most widespread technique in medical imaging are **Convolutional Neural Networks** (CNNs), introduced in 1998 by LeCun et al. in their revolutionary paper *Gradient-based learning applied to document recognition* [53]. CNNs fall within a machine learning discipline known as **deep learning**. They have benefited greatly from the advancement in image acquisition devices and the storage of information in large databases, as well as the increase in

computational resources and the GPU-acceleration. In short, they have superseded the traditional machine learning techniques, such as k-Nearest Neighbors or Support Vector Machines, whose main problem resided in the extraction of relevant features, since they usually required extensive knowledge of the problem and sophisticated feature engineering. CNNs are a promising alternative to all machine learning algorithms developed so far, with potential application to virtually any Computer Vision problem, achieving state-of-the-art results.

1.1 Motivation

This thesis was presented by Javier Abad Martínez as a final project of the degree in Telecommunication Technologies and Services Engineering in July 2020 and was linked to a grant from the Spanish Ministry of Education and Professional Training that same year.

This document develops a CNN-based model to detect metastatic tissue in sections of axillary lymph nodes removed from patients with breast cancer. Breast cancer is the most common cancer in women and the main cause of death due to cancer after lung cancer. A popular technique to diagnose breast cancer and its stage is by detecting metastases in the axillary lymph nodes, since lymphatic spread is the main prognostic factor.

In this context, this thesis finds its motivation in looking for a deep learning model capable of helping pathologists in the diagnosis of breast cancer, specifically in its staging, and being able to define a treatment accordingly.

Furthermore, this document was developed in parallel with another thesis by the same author entitled *A comparison of machine learning models for the detection of metastatic tissue in axillary lymph nodes* [1] in which a benchmark of models based on hand-crafted learning techniques, known for several decades, is developed. Furthermore, [1] aims to demonstrate that some of these techniques may be more convenient than cutting-edge ones, such as CNNs. Therefore, this document has a second motivation: to compare the proposals of traditional models used in [1] with solutions based on Convolutional Neural Networks. The conclusions can be interesting to start a line of research on which methods are better depending on the circumstances, reaching maximum importance in something as sensitive as the diagnosis of breast cancer.

1.2 Summary of the methodology followed

For the preparation of this work, a methodology was followed that occupied most of the academic year 2019-2020, that is, from September 2019 to July 2020.

The methodology followed can be summarized in four parts. First, an introduction to artificial intelligence, deep learning, and the Python programming language. Next, it was necessary to delve into the deep learning technique used – the Convolutional Neural Networks –, as well as to become familiar with the biological background and to do a bibliographic review on how similar problems had been approached. Once the problem and its possible approaches were known, in a third phase different solutions were proposed, some based on models designed from scratch and others on more specialized techniques such as fine-tuning. Finally, this document was written and revised.

In short, the four parts of the methodology followed are:

1. Preliminary work:

   - Introduction to artificial intelligence, machine learning and deep learning. Specialized bibliography was succinctly reviewed.
   - Introduction to the Python programming language. Through courses from the Coursera platform, the author became familiar with the handling of variables and structures in Python, for example, the generation of loops, conditionals or methods. This was decisive, since all the proposed models are based on this language. It has to be noted that Python programming was not given to the author during his degree.

2. Methodological development:

   - Study of the biological background of the problem. The author studied about breast cancer and its classification based on lymphatic spread. In addition, the importance of the axillary lymph node in this diagnosis was studied in depth.
   - In-depth bibliographic review of the state-of-the-art in the classification of medical images. Several models based on deep learning and, more specifically, on Convolutional Neural Networks were studied.
   - Completion of specialized courses in image classification using CNNs in Python. Specifically, a 40-hour classroom course was attended organized by the Computer Vision and Behavior Analysis Lab (CVBLab). CVBLab is a research group that belongs to the Universidad Politécnica de Valencia specialized in signal, image and video processing, as well as in data science and the creation of automatic prediction models.

3. Proposal and selection of models:

   - Characterization of the database. The origin of the database was studied and a set of non-useful images was discarded. Furthermore, its distribution and characteristics were analyzed in order to develop suitable models.
   - Design of the models from scratch. Over thirty different models were developed, incorporating increasingly sophisticated techniques and modules. Finally, of all the designs, three were chosen based on their classification capacity and for didactic reasons, selecting those that best reflected the fulfilment of the objectives of the thesis.
   - Design of the models based on fine-tuning. Based on the bibliographic review, those pre-trained architectures that were considered most appropriate for our problem were chosen, based on criteria of simplicity, popularity, suitability and classification capacity. Specifically, six models were selected.
   - Study of specialized metrics in machine learning and their use in comparing the proposed models.

4. Writing and revision of the thesis.

1.3 Structure of the thesis

This section summarizes the structure of the thesis. This is also reflected in the content index.

Chapter 1 is the introduction to the work. The first part (Sections 1.1, 1.2 and 1.3) serves as a preface, defining some of the concepts that will appear recurrently in the document, as well as a declaration of intent, exposing the motivation of the work, and some considerations, such as the methodology followed. Section 1.4 summarizes the objectives of the present work, separating the general ones from those related to sustainable development. Next, Section 1.5 is a review of the state-of-the-art in the classification of histopathological images. Supported by several references, the section succinctly comments on cutting-edge techniques in this discipline, which serve as an example for the present work. Finally, Section 1.6 studies the biological background of the problem. Therefore, it defines breast cancer, how it is classified and the importance of its diagnosis. Next, the importance of detection techniques for lymph node metastases to treat breast cancer is highlighted, as is the one in this study.

Chapter 2 develops in depth the theoretical framework of the proposed models. It introduces deep learning through Artificial Neural Networks, specifically with the well-known multilayer perceptron. The characteristics and functioning of the model used are then detailed: Convolutional Neural Networks. The chapter explains its composition and some techniques used to create more sophisticated networks that will ultimately result in better classifiers. Finally, fine-tuning is explained as a powerful technique to obtain excellent results in practically any Computer Vision problem.

Chapter 3 defines the methodology followed. Firstly, the database used, the materials and the metrics with which the models are evaluated are explained. All the proposed models are explained in detail below.

Chapter 4 summarizes the results obtained. It separates those obtained with the models from scratch and those that use fine-tuning. Furthermore, Grad-CAM is presented as a technique to validate the correct functioning of the proposed models, visualizing the activations in the classifications of some of the histopathological images from our database.

Chapter 5 covers the discussion of the results. Therefore, it analyzes the values obtained in Chapter 4. In addition, there is a specific section where it compares the performance of the system proposed in this document with that proposed in [1] based on traditional machine learning algorithms. Finally, Section 5.3 explains the limitations of the work, that is, the biases, assumptions, and simplifications.

Finally, Chapter 6 summarizes the conclusions and describes the potential future work to improve the present model.

1.4 Objectives

1.4.1 General objectives

This section presents the general objectives of the work. These serve as a guide and are directly related to the conclusions stated in Section 6.1. The objectives are as follows:

- Study the importance of lymphatic spread as the main prognostic factor in the diagnosis and staging of breast cancer, especially in early stages. Along these same lines, understand the biological background and the usefulness of detecting metastatic tissue in axillary lymph nodes for this diagnosis.
- Develop a model based on Convolutional Neural Networks capable of detecting metastatic tissue in patches of axillary lymph node sections stained with H&E (haematoxylin-eosin).
- Experiment with techniques of different levels of sophistication in creating models from scratch to later analyze their classification capacity. In this sense, demonstrate that the incorporation of these techniques improves the applicability of the models and their suitability in hospitals.
- Use fine-tuning in the creation of models based on pre-trained architectures. Demonstrate the usefulness of this technique and its applicability in virtually any Computer Vision problem.
- Compare cutting-edge techniques with others based on traditional machine learning, such as k-Nearest Neighbors (k-NN), Support Vector Machines (SVM) and Random Forest. Define the advantages and disadvantages of each and the suitability of each solution depending on the problem and the context.
- Provide a tool that facilitates the understandability of deep learning and its applicability in medical imaging. Specifically, implement a technique that allows network validation by expert pathologists, so that they can verify the coherency of the identified patterns.

1.4.2 Sustainable development objectives

Furthermore, it is relevant to highlight objectives directly related to sustainable development. Since the present project aspires to have a real application, it is necessary to establish objectives that favor the environment, the economy, and society. The objectives are:

- Favor the diagnosis and treatment of breast cancer, facilitating these functions for the pathologist thanks to the direct integration of the techniques presented in this document in hospitals.
- Analyze the computational cost and, consequently, the carbon footprint linked to machine learning techniques. Evaluate and compare different models that can benefit from GPU-acceleration to reduce this cost and environmental impact.
- Democratize the technology developed in this thesis and stimulate the creation of projects along the same lines, with direct implementation and the ability to help in settings and problems as sensitive as the diagnosis of breast cancer.

1.5 State-of-the-art review

The application of machine learning techniques to Computer Vision has brought about a real revolution in object detection, classification and segmentation problems. Therefore, it is not surprising that machine learning models have been applied to extremely sensitive fields, such is the case of medical imaging. Medical image comprises the set of procedures used to obtain clinically meaningful information in order to define early diagnoses and prognoses in patients. In this context, an effort has been made in recent years to integrate the human knowledge of medical experts with machine learning techniques. This symbiosis has achieved state-of-the-art performance in the field of medical imaging techniques, such is the case of positron emission tomography (PET), computed tomography (CT), magnetic resonance imaging (MRI) mammography, X-ray and ultrasound [18].

The accumulation of histopathological images has gone hand in hand with an increasing need for computer-assisted diagnosis techniques based on machine learning. However, meaningful feature extraction engineering, capable of describing the images, is considerably complex, requiring extensive subject matter knowledge. For this reason, the incorporation of deep learning [52] techniques represents a real revolution in the field of medical imaging, obtaining state-of-the-art results in pathological image analysis and outperforming in many applications other machine learning techniques that use feature extractors based on hand-crafted learning and traditional classifiers such as k-Nearest Neighbors, Support Vector Machines or Random Forest. Deep learning incorporates feature engineering into its learning process, so there is no need for manual feature extraction, and aims to solve virtually any classification problem [6].

Deep learning is extremely convenient when the number of images available is very large, however this is not usually possible in medical imaging, where the data collection process is tedious and complex. Faced with this challenge, various strategies have been developed, such as dividing the whole histopathological images (whole slide image (WSI) [64]) into two-dimensional or three-dimensional patches [87] [13] [71] [81]; expanding the dataset by generating synthetic images via affine transformations, such as data augmentation [13] [71] [81] [80]; or the use of transfer learning and fine-tuning, inheriting pre-trained models with previously optimized parameters [83]. Furthermore, the use of these techniques has been greatly benefited from advances in high-tech Central Processing Units (CPUs) and Graphics Processing Units (GPUs), and the development of increasingly sophisticated techniques and models [32] [38] [60] [86]. The integration of these techniques has materialized in increasingly refined proposals, with better results in the classification of histopathological images. Specifically, the Convolutional Neural Networks (CNNs) are the ones that have had a greater role in this field, repeatedly demonstrating their applicability and suitability in the clinical environment of hospitals and medical research [62] [19] [101]. In any case, it is relevant to highlight the usefulness of other models such as Deep Generative Models [76] – including Deep Boltzmann Machines (DBMs) and Deep Belief Networks (DBNs) – whose results are widely contrasted in the literature, for example, in cell segmentation [24], brain disease diagnosis [59] and tissue classification [92].

A specific and recurring application of these technologies is the detection, classification and staging of breast cancer. The usefulness of deep learning has been widely demonstrated, for example, in the diagnosis of breast cancer using histopathological images of axillary lymph nodes, in fact, equaling and outperforming expert pathologists, according to several studies [5] [56] [3].

Specifically, CNNs are the technique that stands out in this diagnosis, with state-of-the-art results.

However, it is relevant to underline that CNNs cannot be considered the panacea of all medical imaging problems. There are several examples where traditional methods, such as k-Nearest Neighbors or Support Vector Machines, obtain better results than deep learning techniques [65]. In addition, there are several criticisms that CNNs receive, based on their high computational cost, their tendency to overfit and their instability, which cast doubt on their real usefulness [2]. In addition, another criticism of CNNs is the difficulty of understanding their decision-making process, presenting themselves on most occasions as black-boxes. This problem has been tried to solve, for example, by visualizing the responses of neural networks in their classification process [77] [102] or by using influence functions [45].

1.6 Biological background

1.6.1 Breast cancer

Breast cancer originates from the uncontrolled proliferation of healthy breast cells, which begin to mutate and form a conglomerate of cells known as a tumor. If the tumor is cancerous, it will be a malignant tumor. Malignant tumors can grow and spread throughout the body, unlike benign tumors, which can grow, but not spread. The spread of cancer through the body through the blood or lymph vessels is called metastasis [10].

The diagnosis of breast cancer in women is more common than that of any other type of cancer, including skin cancer. In addition, it is the second most common reason for death from cancer in women, after lung cancer. According to the American Cancer Society, it is estimated that in 2020 more than 276,480 women will be diagnosed with invasive breast cancer in the United States and will produce a total of 42,170 deaths of women in this same country. Currently, there are more than 3 million women who have been diagnosed with this disease throughout the United States. Finally, the average risk that a woman will develop cancer in the United States throughout her life is around 13%, that is, 1 in 8 women will develop this disease [12].

There are several ways to classify breast cancer. In this work, the so-called **staging** is the used method, particularly the TNM staging system. TNM is based on the size of the tumor (T); if it has spread through the lymph nodes (N) and, if so, where and how much; and if it has spread to other parts of the body (M), that is, if it has metastasized. Under the TNM indicator, the patient is identified with one of the five different stages, which cover the development of the cancer from the moment the cell becomes carcinogenic (stage 0) to an advanced stage or metastasis where the cancer has spread throughout the rest of the body (stage 4) [10].

1.6.2 Lymphatic dissemination in breast cancer

The "N" in the TNM indicator corresponds to lymphatic spread. It is relevant to differentiate here between regional and distant lymph nodes. Regionals can be located under the arm (known as axillary lymph nodes), above and below the clavicle, or below the sternum (known as internal mammary lymph nodes). Rather, distant lymph nodes will be located in the rest of the body. Therefore, it is expected that a breast cancer in a lower stage will be present only through regional lymph nodes, reaching the distant ones as the disease progresses.

The spread of cancer by lymph nodes is the most relevant prognostic factor in the initial stages, above the size of the tumor (T) and whether it metastasizes (M). Thus, it is a definitive indicator when diagnosing, classifying, and defining the treatment of a patient with breast cancer [10].

Finally, there is a popular form of diagnosis of breast cancer called sentinel lymph node biopsy (SLNB). The sentinel lymph node is the one to which cancer cells are most likely to spread from the primary tumor. In the case of breast cancer, this is usually the axillary lymph node. Therefore, the procedure will consist on the removal and examination of the sentinel axillary lymph node (SLN) to see if it has metastatic tissue. This study would define whether the cancer has been able to spread throughout the rest of the body, which allows a diagnosis of the patient's stage. Moreover, on examination of the lymph node, the pathologist will require staining to obtain an overview of the tissue samples [47].

1.6.3 The haematoxylin and eosin (H&E) stain

A popular stain is haematoxylin-eosin (H&E). Since most cells do not have their own color, their direct observation by light microscopy does not allow their morphology to be analyzed properly. To be able to observe it, stains are used in order to color different structures in specific ways, making it easier to differentiate them within a tissue.

Specifically, the haematoxylin-eosin stain colours with haematoxylin the acidic structures in blue and purple while using eosin for basic components, colouring them in pink tones [37]. Figure 1.1 shows some histological images stained in H&E.

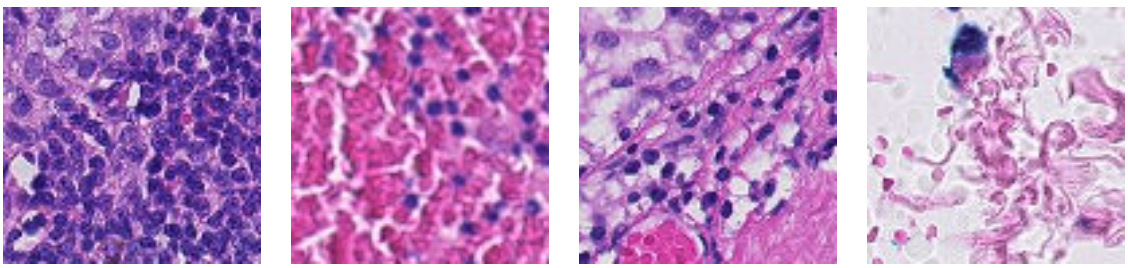


Figure 1.1: Histological images of axillary lymph nodes stained with H&E.

1.6.4 Automatic detection of metastasis in sentinel axillary lymph nodes (SLN)

New technologies in medicine have allowed high resolution digitization of stains with histopathological tissue, including haematoxylin-eosin staining. Advances in scanners allow the storage of large databases of medical images that can be processed by machine learning models for classification [28].

Section 1.5 studies the current state-of-the-art techniques used for medical images. In the present work, the problem is characterized to the detection of metastases in sentinel axillary lymph nodes (SLN). As mentioned, this analysis has a fundamental relevance in the staging of breast cancer and, therefore, in the definition of a treatment for the patient, especially in early stages.

Applying machine learning to this particular problem is highly desirable. Pathologists' general diagnosis of SLN is especially complex, often leading to erroneous conclusions. One study found that expert pathologists change up to 24% of SLN diagnoses after reviewing them [97]. Furthermore, the analysis is considerably tedious and time consuming. It is, therefore, especially interesting to create an algorithm capable of classifying quickly and with high accuracy a SLN susceptible to metastasize.

Chapter 2

Theoretical framework

This chapter develops the theoretical framework in which this work is included. In a first section, Artificial Neural Networks are introduced, specifically a type of neural network widely used in pattern recognition: the multilayer perceptron. In addition, some machine learning techniques used to build more sophisticated models, with better learning capacity and, ultimately, classification power, are summarized. The multilayer perceptron actually serves as the basis for the networks that really concern us: Convolutional Neural Networks (CNNs). CNNs are explained in detail in the second part of the chapter, where their structure, their operation and the hyperparameters that model them are explored. Finally, the last section briefly explains some of the most popular and best performing CNN architectures in the ImageNet database. In this way, the power of the use of transfer learning and fine-tuning is highlighted by inheriting these pre-trained networks and customizing them for any problem.

2.1 Introduction to Artificial Neural Networks

The ambition to create intelligent systems, capable of learning and making decisions based on their own experience, has been a constant for more than 50 years. Modern computers vastly outperform humans thanks to powerful numerical processing, memories capable of storing large amounts of information, and the ability to calculate multiple scenarios and alternatives. However, until today it has not been possible to create a machine capable of totally emulating the human being, that is, his ability to solve abstract problems, to learn from his mistakes or, simply, to imagine or to be creative. It is in artificial intelligence and, more specifically, in Artificial Neural Networks (ANN, from now on), where a symbiosis has been achieved between both domains.

The term "neural network" originates from experiments to find mathematical representations of biological information processing systems, such as those proposed by von der Malsburg [57] and McCulloch [58]. ANNs aspire to solve virtually any problem of classification, prediction, optimization or patter recognition, obtaining state-of-the-art results.

In this section, the multilayer perceptron is introduced as a type of ANN widely present in the specialized literature on artificial intelligence. It is important to note, therefore, that the multilayer perceptron is simply a type of network and is not fully equivalent to ANN. Within the taxonomy proposed in [39] there are other techniques based on ANNs applicable to multiple problems, such as radial basis function networks [9], Hopfield networks [33] or ART models [11].

2.1.1 The multilayer perceptron

The multilayer perceptron is, without a doubt, the most widely used type of ANN. Its popularity is justified by its success in solving considerably complex problems, serving as an alternative and outperforming the results obtained by traditional statistical techniques [78]. The multilayer perceptron also performs well for virtually any problem, regardless of discipline. This is due, among other reasons, to the fact that it does not require any assumption regarding the distribution of the data it receives as input, and can even model complex non-linear functions and obtain very satisfactory results when generalizing to a set of independent data [34].

The multilayer perceptron is constituted as a system of interconnected neurons. A neuron is a computational unit characterized by weights and an activation function. In this way, given an input, the output of the neuron will be computed by applying the activation function on the matrix product between the input and the weights. On the one hand, **weights** are the equivalent, for example, to the coefficients in a linear regression problem. The objective will be to optimize these weights for each of the neuron layers, so that the predictions made are as accurate as possible. On the other hand, the **activation function** modifies the output of the product to implement nonlinearities. It is precisely the superposition of these nonlinearities that allows to model more complex problems. As will be seen later, in the multilayer perceptron learning algorithm – known as back-propagation – it is essential that the derivation of the activation function is easily computable. For this reason, the most commonly used functions are sigmoid, softmax, hyperbolic tangent, and rectified linear unit (ReLU). It is precisely the latter that obtains the best results in deep networks and, therefore, the most used, as demonstrated by Glorot et al. in [26].

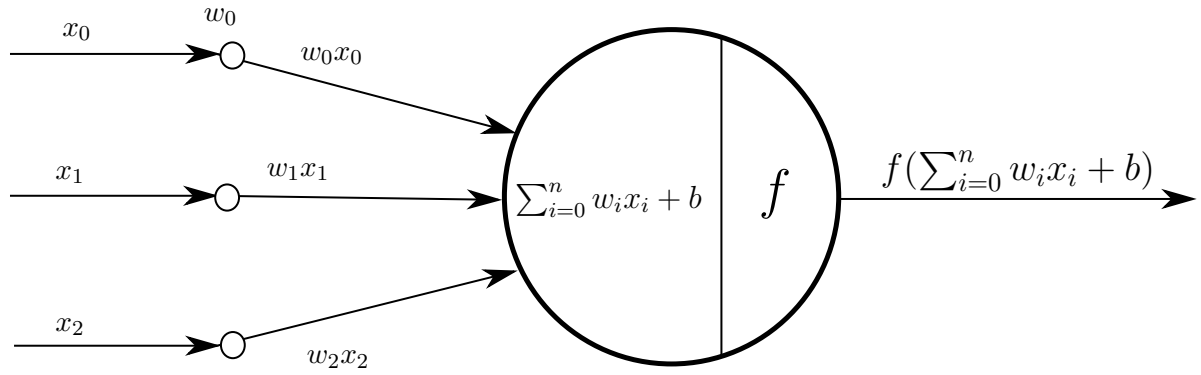


Figure 2.1: Functioning and parameters of a neuron.

The structure and functioning of a neuron is thus defined. These neurons are placed in rows called layers, making up the set of layers the neural network. The multilayer perceptron is described as a fully-connected structure, with all neurons in each layer connected to all neurons in the previous and next. The first layer, known as the visible layer, does not really have a

computational function, it simply represents the vector of inputs. The following layers are known as hidden layers since they are not directly exposed to the input. These are characterized by the weights and an activation function, generally ReLU. Finally, the output layer represents the classifier. In classification problems, such as the one in this document, this will determine which category the object belongs to. Again, it will be characterized by the weights and an activation function, however, in this case the function will be sigmoid or softmax, in most cases. Sigmoid is frequent in binary classification, establishing a threshold of 0.5 to discriminate between the two categories, while softmax is used in any type of classification, computing the object's probability of belonging to each of the categories and selecting the highest. Figure 2.2 illustrates a simple neural network based on the multilayer perceptron.

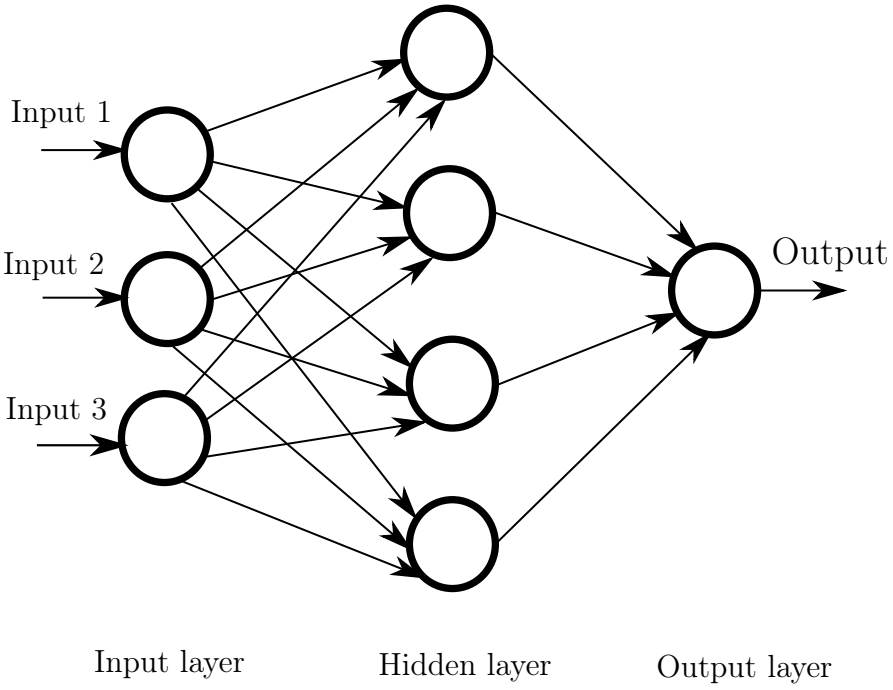


Figure 2.2: Example of a multilayer perceptron with a single hidden layer of four neurons. The input layer is formed by a vector of three components, expressed with three neurons. The output layer has only one neuron, so the network will be used for binary problems.

Finally, it remains to define the process of classification and learning of the multilayer perceptron. These are linked to two very well-known mechanisms in the machine learning literature: forward-propagation and back-propagation.

Forward-propagation is the mechanism of generating an output from an input or, in other words, the process by which the multilayer perceptron predicts the category to which the input belongs. Each hidden layer receives the input data, performs the matrix product and applies the activation function to the result. Next, it sequentially passes the output to the next hidden layer, until it reaches the output layer, where the object is finally classified.

On the other hand, it is in the **back-propagation** mechanism that the value and success of the multilayer perceptron really lies. It has been seen that each neuron is characterized by optimized weights that, during forward-propagation, will result in the most accurate predictions possible. The network learning process, from which these weights are optimized, is known as

back-propagation, and was introduced by Rumelhart et al. in their well-known publication [73]. The idea is as follows: the optimal weights must be those that minimize the loss function, calculated by comparing the predictions made with the true labels of the objects that are classified. These true labels are called the **ground truth**. It should not be forgotten that the problem dealt with in this document is a supervised learning problem and, therefore, the ground truth is known in the training phase, that is, in the network learning phase. In this way, the back-propagation algorithm aims to locate the absolute minimum of the loss function that the system models. To this end, the most widely used technique is the **gradient descent**, along with its modifications and improvements, among which RMSProp, Adam or SGR stand out. Furthermore, the programmer can control this learning process by setting the learning rate and the momentum. The **learning rate** is a hyperparameter that quantifies how much the weights are updated, so a very low learning rate can lead to very slow convergences, while a very high learning rate can cause learning to diverge or not find the absolute minimum. **Momentum**, meanwhile, helps the gradient escape from the local minimums. In short, back-propagation will consist of a process of minimization of the loss function in which the weights will be updated following the gradient descent technique and the defined learning rate. The gradient descent algorithm and the process of updating the weights include considerable mathematical development, illustrated in Appendix A and Appendix B, respectively.

In summary, the operation of the multilayer perceptron can be summarized in the following steps:

1. Initialization of weights. It can be a random initialization or using newer techniques, such as He initialization [30], or Xavier initialization [50]. Initializing with zeros is disregarded in all cases.
2. Forward-propagation: the input vector – with the objects to classify – propagates through the hidden layers until obtaining a prediction.
3. The predictions are compared with the actual values, and the value of the loss function is computed.
4. Back-propagation: the error calculated in the loss function is propagated, updating the weights to minimize this value.
5. The process is repeated until the error and, therefore, the accuracy are satisfactory.

This implementation corresponds to what is known as on-line training, in which the network is trained in sequential order as new data becomes available. However, the most common and, in fact, used in this document, is to use **batch training**. In this one there is a specific training set that will serve to train the network. This training set is divided into a set of **batches** – typically 32, 64 or 128 instances each – which are progressively introduced into the network, which adjusts its weights accordingly. In [4] the interested reader can consult the principles of both approaches and their advantages and disadvantages. Furthermore, every time the entire set of training data passes over the network, an "**epoch**" is said to have passed. Both the size of the batch and the number of training epochs are decisions to be made by the programmer, often selected empirically or by trial and error.

The principles of the multilayer perceptron have been explained in depth in this section. However, this is one of the simplest versions of neural networks. In the next sections, some techniques are introduced to improve network performance, such as the use of dropout and batch normalization,

as well as some practices to ensure correct generalization of the ANN in a separate data set, such as the use of data augmentation. Finally, the training, validation and test processes are detailed. This will serve as a basis for Section 2.2 to delve into the technique used in this work: Convolutional Neural Networks.

2.1.2 Dropout

One of the main problems present in ANNs is the possibility of overfitting. **Overfitting** is produced by over-training the model using the same data set, so that it ends up identifying excessively specific patterns and features of these specific instances, making it impossible to generalize them to an independent dataset.

Srivastava et al. propose in [86] a solution to overfitting through the use of dropout. Dropout consists of randomly ignoring some neurons during the training phase. In this way, during the training with each of the batches, each neuron in the layer will have a probability p of staying connected or, in other words, a $1 - p$ probability of disconnecting and, therefore, not fitting. It is relevant to mention that dropout is applied exclusively in training, keeping all the neurons connected both in the validation and in the test. Figure 2.3 illustrates the operation of the layers of a neural network without and with dropout.

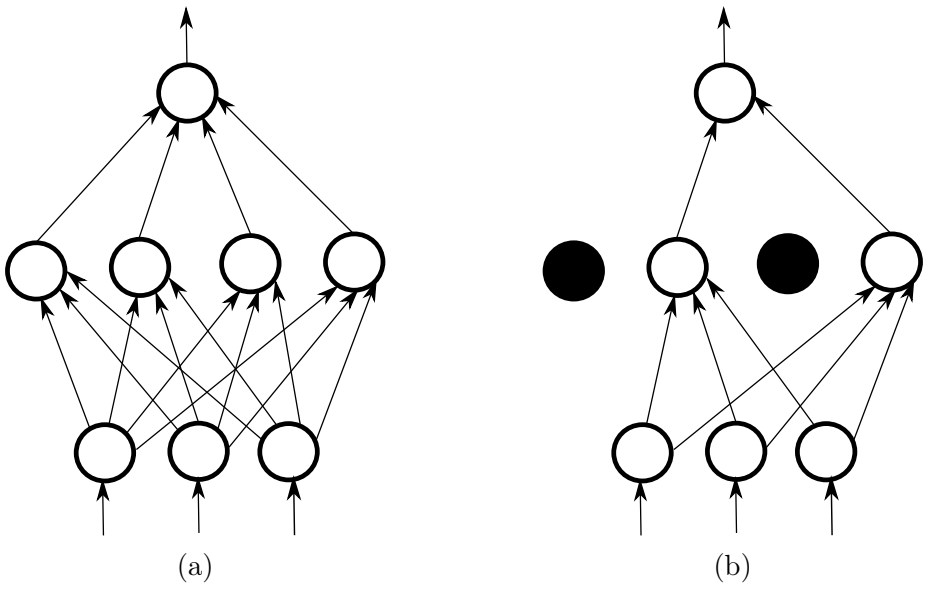


Figure 2.3: (a) Standard Neural Network and (b) After applying dropout.

2.1.3 Batch normalization

Training deep ANNs is considerably complicated due to the fact that the distribution of the inputs of each layer changes during it, as the weights of the previous layers also change. In other words, small changes in previous layers affect subsequent layers. This effect, in addition, is amplified when the network is deeper. This is a great difficulty for the programmer, who must be extremely careful when choosing the initialization of the weights and the hyperparameters that model the training, such as the learning rate. This problem is known as *internal covariance shift* [82] and results in excessively slow trainings.

In practice, this effect can be reduced by using ReLU as activation function [60], correct initialization of weights [25] or small learning rates. However, in 2015 Ioffe and Szegedy propose batch normalization [38], addressing this problem by integrating this normalization as a part of the network. The method consists of normalizing the output of the previous layer by subtracting the batch mean and dividing by the batch standard deviation. However, systematically normalizing the weights in this way will involve adding noise to the network, making the weights be further from the optimal, so the gradient descent algorithm will undo this normalization to minimize the loss function. Therefore, after normalization, two new trainable parameters (γ and β) are added, which will be adjusted during training so that the gradient descent algorithm optimizes them instead of directly performing a denormalization. Specifically, γ works as a "standard deviation" and β as a "mean". Figure 2.4 illustrates the integration of batch normalization to a system.

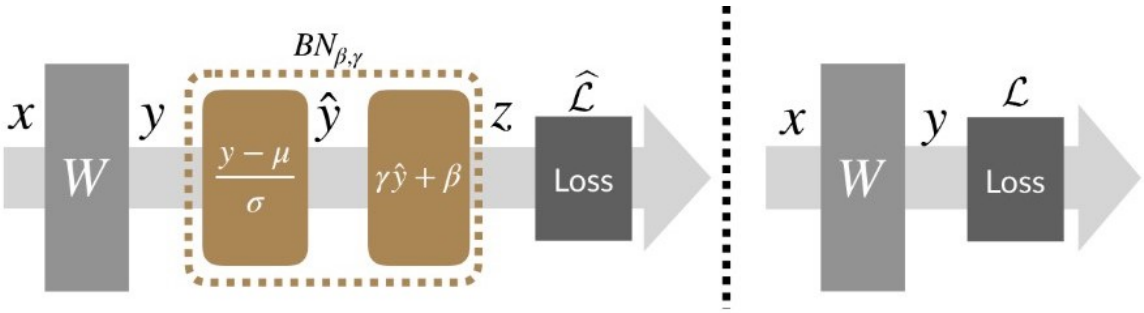


Figure 2.4: On the right a standard network, on the left a network to which batch normalization has been integrated. β and γ are incorporated as new hyperparameters of the system. Retrieved from *How does Batch Normalization Help Optimization?*, by A. Ilyas et. al, 2018, `https://gradientscience.org/batchnorm/`. Copyright by gradient science.

In short, the use of batch normalization allows higher learning rates to be used, speeding up the training of the network, as well as reducing the importance of the initialization of the weights. In addition, it acts as a regulator just like dropout, since it incorporates noise into the system. In the original paper [38] the success of batch normalization is evident in achieving state-of-the-art results in the classification of the ImageNet dataset, accelerating training to require only 7% of the original training steps

2.1.4 Data Augmentation

Neural networks and, more specifically, Convolutional Neural Networks (see Section 2.2) perform considerably well in Computer Vision tasks. However, one of the main problems is their high dependence on large databases for training to avoid overfitting. This problem is especially prevalent in tasks related to medical imaging, where images tend to be a scarce resource, as is the present work.

In order to build useful and generalizable models to independent databases, it is essential to avoid overfitting, and data augmentation is a powerful technique to achieve this. It consists of increasing the training set by generating synthetic images from the training images. This artificial data creation process is generally based on image manipulation: geometric transformations, flipping, color space transformations, rotations, translations, cropping, etc. [84].

The use of data augmentation to regularize and avoid overfitting in medical imaging problems is widely contrasted in the literature, for example, in the classification of skin lesions [23] or liver pathologies [43]. It is relevant to mention that this is not the only regularization technique that exists, but there are many others such as the previously mentioned dropout and batch normalization.

2.1.5 Training, validation and test

In the previous sections, ANNs have been illustrated, using the well-known example of the multilayer perceptron, as well as some regularization techniques that increase the sophistication of the network. This section explains how this model is trained, validated and tested.

There are several approaches that can be found in the literature on how to split the dataset. The traditional stance consists of a single partition on training and test data. The idea is as follows: if the model has been adjusted – that is, it has learned – from a dataset, it would be unfair to evaluate its performance on that same dataset. Therefore, an independent dataset (test) is reserved to test the performance of the fitted model on unknown and therefore unbiased data. Specifically, in the case of supervised learning, the labels of the test set are known and, therefore, testing serves as an evaluation of the model, comparing the correct labels with the predicted ones. This approach is the one proposed, for example, by recognized machine learning books [41] [49].

In this document, however, we bet on a double partition. After dividing the original dataset between training and testing, the training set is divided again between "real" training and validation. This validation set will allow to evaluate the accuracy of the model in each of the epochs, thus studying its learning and identifying the possible overfitting. In this way, the use of the test set is limited to the generalization of the final model. This position is also recurrent in the literature, for example in [75]. In general, the test set represents 10-15% of the total, while the validation 20-25% of the training data.

It is convenient now to summarize each of the identified sets:

- Training set: it is used to adjust the model parameters, these are, the weights of the layers, among others. It tries to find optimal parameters that minimize the loss function.
- Validation set: gives information on the accuracy obtained by the model in each epoch on an independent dataset during training. In this way, it is possible to monitor the learning process and identify overfitting problems.
- Test set: once the model has been trained, it is generalized to an independent dataset to evaluate its performance. Unlike validation, this assessment is only done once and is computed out of training.

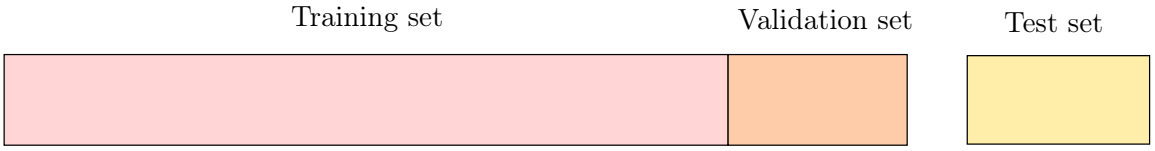


Figure 2.5: Double partition of the dataset: training and validation, and test.

2.2 Convolutional Neural Networks

For decades, conventional machine learning techniques have been limited to processing data in its raw form. Extensive knowledge in the domain and feature extraction engineering was necessary to obtain satisfactory results, so that it was possible to represent the data by means of more meaningful information and, therefore, it was easier to recognize patterns that would serve to classify the objects. This problem was especially prevalent in Computer Vision, due to the complexity of training the algorithm to emulate the behavior of the human eye, that is, of being able to identify textures, colors or shapes.

In this context, the success of deep learning is not surprising. Deep learning is a representation learning technique with multiple levels of representation, so it is capable of decomposing input data into representations of increasing levels of abstraction, starting from raw data to complex patterns. Finding its usefulness in Computer Vision problems is easy: the initial layers will be able to identify general shapes and contours, while, as the network becomes deeper, it will locate more specific and differentiating shapes and features.

Among the deep learning techniques, the most popular have been Convolutional Neural Networks (CNNs, from now on), introduced by LeCun et al. in [53]. CNNs are imposed as an ideal technique for any Computer Vision problem, obtaining state-of-the-art results in problems as diverse as traffic signal recognition [16], biological image segmentation [63] or facial detection [51], among others. However, the total success of CNNs did not come until the ImageNet competition of 2012, where the AlexNet network obtained excellent results in the classification of more than a million images in 1,000 different categories. Furthermore, CNNs have benefited greatly from advances in GPU-acceleration [15], as well as regularization techniques such as dropout, batch normalization or data augmentation.

Some of the advantages of CNNs are the independence with the space and time distributions of the images or the decrease of trainable parameters compared to other types of networks, since it is a system based on filter banks, instead of products with whole matrices like the multilayer perceptron. This section explains in detail the parts of a CNN, its parameters and its operation. In addition, two techniques are introduced that increase CNNs' sophistication and performance: the residual blocks and the inception blocks.

2.2.1 Convolutional layer

The convolutional layers are the feature extractors of the image. Each convolutional layer is characterized by a bank of filters known as **kernels**. The operation consists of a convolution between the features represented as the input of the layer and the kernel, so that they get more and more specific features from the original images as the network gets deeper. In this way, the kernel goes through the image convolving itself with the different sections of it. The kernel functioning is defined by a stride. The stride explains the movement of the kernel in the different dimensions, so that if the stride has a value of two, the filter will move with two pixel shifts, first in the width and then in the height of the image (assuming that it is grayscale and there is no third dimension). This process is repeated as many times as kernels the bank has, a value that can be parameterized by the network designer, obtaining as many results as defined filters. The result of convolving all the filters with the input is known as feature mapping.

In the case of working in multi-dimensional color spaces (e.g. RGB, HSV), the kernel will have this same depth, so that the results of convolving in all the dimensions are added together, resulting in a single dimension matrix. Figure 2.6 illustrates this operation, using a 3x3 filter on the three color channels of the image.

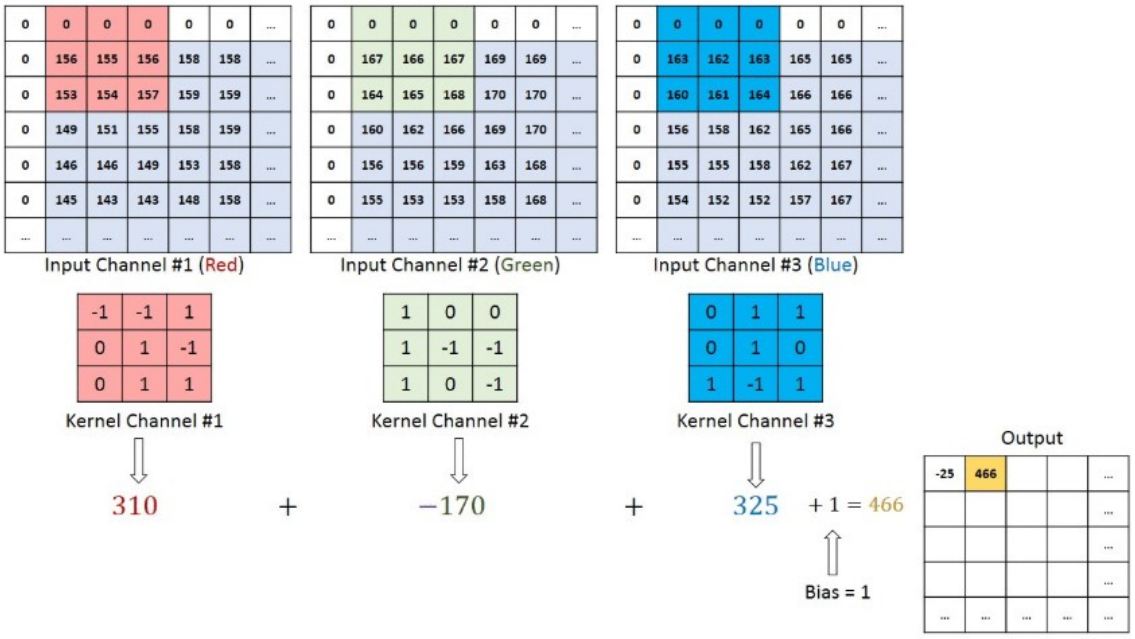


Figure 2.6: Example of convolution with a RGB image. The convolution is computed with a 3x3 kernel with the image section in its three channels (Red, Green, and Blue), and then added together with the bias. The result of the operation will have only one channel. Retrieved from *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*, by S. Saha, 2018, `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53`. Copyright by Towards Data Science.

Due to the border effect, it can be the case that the input does not exactly match the kernel and, therefore, the output does not maintain the same dimensions. Here are two approaches: fill the input with zeros (zero-padding or **same-padding**) or drop the part of the image where the filter does not fit (**valid-padding**), leaving only the valid part of the image. In the first case, the output will be the same size as the input, being smaller in the second case. This decision is parameterizable and will depend on the particular problem

Finally, the result of the operation is passed through an activation function, typically ReLU. As in the multilayer perceptron, this serves to model possible nonlinearities in system relationships.

Thus, in CNNs the parameters to be optimized are the filter coefficients, unlike the multilayer perceptron where complete matrices are fitted. Furthermore, the functioning of the convolutional layers is based on two principles. The first is that images tend to have highly correlated local groups of pixels, representing, for example, the same object. The second is that the pixel distributions are invariant of their location, that is, an object can appear anywhere in the image. Consequently, the result when applying the kernel to the image will be similar in both locations, thus facilitating the identification of patterns in different parts of the image.

2.2.2 Pooling layer

The main problem with convolution is its high computational cost. If, in addition, a high number of kernels is defined, for each input there will be a convolution with K different filters. This dimensionality increases with the depth of the net, considerably slowing down the training. For example, if the input image is 28x28x1 in size and a filter bank with 32 kernels of 3x3 size is applied, the feature mapping will be 28x28x32 (assuming zero-padding), which is 32 times larger than the original. It is therefore interesting to reduce this dimensionality, while maintaining the dominant features extracted.

Since the dominant features are invariant with their position and orientation, a pooling layer can be applied to reduce the dimensionality and, consequently, the computational cost. This layer is also characterized by a kernel, which defines the section in which the pooling is computed and goes through the entire image. There are two popular pooling techniques:

- Average pooling: computes the average of the portion of the image covered by the kernel.
- Max pooling: returns the maximum value of the portion covered by the kernel. Furthermore, max pooling works as a noise suppressor and therefore tends to reproduce better results than average pooling.

Figure 2.7 illustrates an example of each of these techniques, using a 2x2 size kernel.

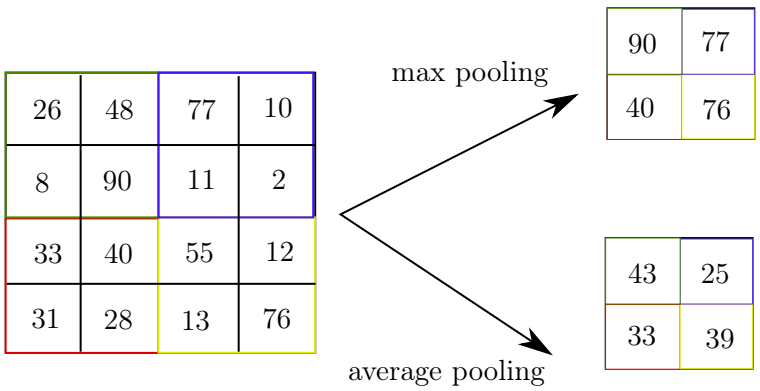


Figure 2.7: Exemplification of the operations of max pooling and average pooling.

2.2.3 Fully-connected layer

The fully-connected layer is basically a multilayer perceptron (see Section 2.1.1) which, after having gone through several blocks of convolutional layers and pooling, is fed with information from which it is capable of finding patterns and, therefore, classifying. The previous convolutional layers should have been able to extract high-level features from which the multilayer perceptron can learn, while also being able to identify possible non-linear relationships in this space.

Therefore, the output of the last convolutional block is simply flattened and inserted into the perceptron, which may be made up of several layers of neurons defined by weights and an activation function. Finally, the object will be classified in a last layer whose activation function will typically be softmax or sigmoid. Figure 2.8 represents a complete CNN, made up of the different layers described throughout these sections.

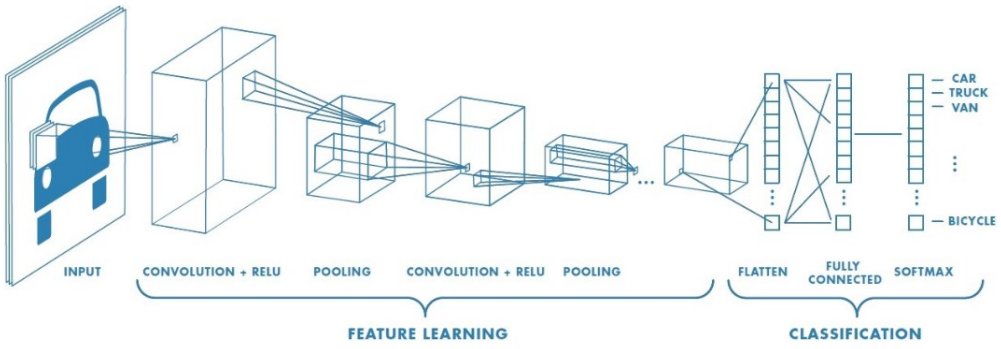


Figure 2.8: Example of a Convolutional Neural Network. Two of the convolutional blocks are shown, consisting of a convolution layer, ReLU activation and pooling. A classifier consisting of a flatten, a fully-connect layer and softmax activation for the classification is added. The network classifies the input among various means of transport. Retrieved from *Basics of the Classic CNN*, by C. Churh Chatterjee ,2019 ,`https://towardsdatascience.com/basics-of-the-classic-cnn-a3dce1225add`. Copyright by Towards Data Science.

In summary, CNNs, the principles of their operation and their basic structure have been introduced. The next two sections describe two techniques used in CNNs that aim to improve their performance, especially in considerably complex problems where pattern identification is not trivial, not even for the human eye.

2.2.4 Residual blocks

In Section 2.1 it was explained that ANNs are universal approximators, which implies that they necessarily have to increase the accuracy with the depth of the network to be able to model any type of problem [34]. Therefore, it would be expected that CNNs could be used for any classification based on Computer Vision, however, this is not the case. ANNs in general and, more specifically, CNNs present training problems when they are excessively deep. For example, due to **vanishing gradients**, the gradient in very deep networks vanishes to very small values, preventing the weights or kernels from changing value and adjusting [46]. Another problem is known as the **degradation** problem, which identifies that counterintuitively the accuracy of training begins to degrade at a point when the network is too deep, not achieving satisfactory results [72].

He et al. introduce in [29] the revolutionary concept of residual blocks to solve these problems. The authors propose the ResNet architecture as a robust network, immune to the problems of vanishing gradients and degradation. The approach is to add a shortcut that allows the output from one layer to feed not only to the subsequent layer but also to a further one, for example two or three layers ahead. Figure 2.9 illustrates the basis of this technique.

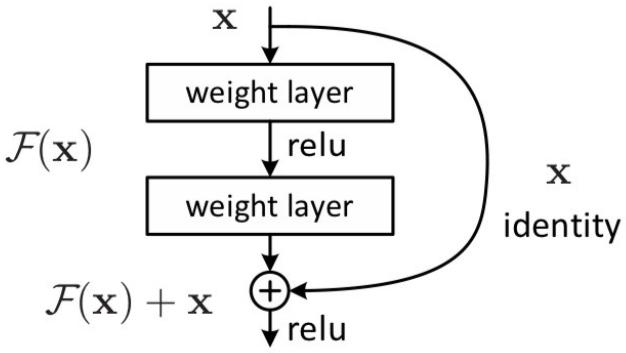


Figure 2.9: Residual learning: a building block. From *Deep Residual Learning for Image Recognition* [30], by K. He et. al, 2015.

Including the information from the previous layers in the subsequent layers implies that the deeper ones have both specific and more general information, which involves that theoretically they will have, at least, the same results as the shallow layers. In addition, adding residual blocks in no case affects negatively the performance of the network since, in the case in which they do not contribute anything, the weights of the residual block would take a value of zero, which would be equivalent to a traditional network. Two basic types of residual blocks can be identified, depending on whether the input / output dimensions coincide or not:

- The identity block. The default residual block simply adds shallow layer information to the deep layer input using a shortcut.

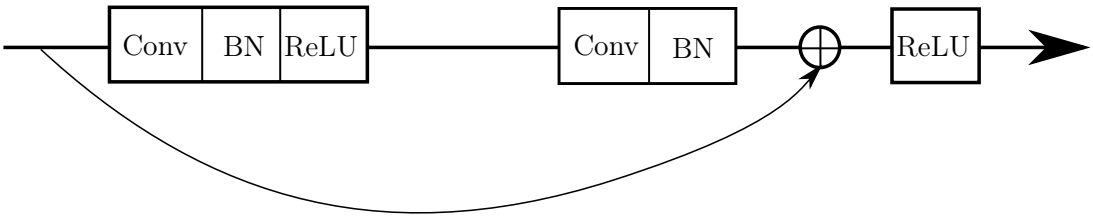


Figure 2.10: Identity block.

- The convolutional block. The information passed through the shortcut is also convolved, therefore changing its dimensions. Furthermore, this implies that the residual block has weights to optimize.

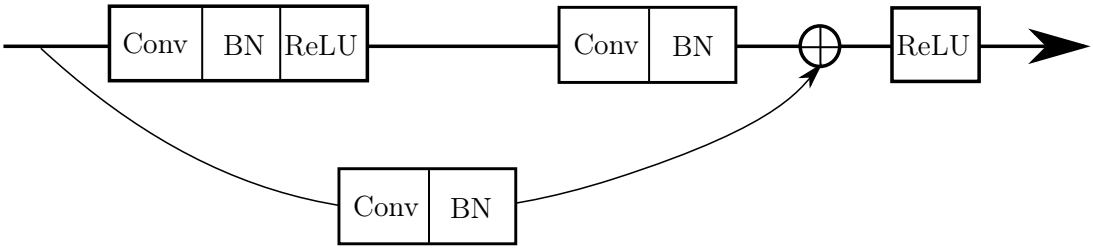


Figure 2.11: Convolutional block.

In addition to ResNet, the CNN research community has proposed several alternatives based on residual blocks that obtain state-of-the-art results. Some are ResNext, developed by Xie et al. in [98], or DensetNet by Huang et al. [36].

2.2.5 Inception blocks

Vanishing gradients and degradation are not the only problems that can arise in the development of a CNN, especially when they are very deep. Section 2.2.1 explained that the designer has to make several decisions when implementing convolutional layers, among which is the kernel size. This decision is not trivial and has a considerable impact on the classification capacity of the network, that is, on the accuracy obtained. Some reasons why kernel size is a determining decision are:

1. The objects to be classified within the image can present considerable size variations. CNNs are invariant to the location and rotation of objects, however, they can have very varied sizes and arrangements. Figure 2.12 shows, for example, different ways in which a dog can be represented. In this sense, it is difficult to choose a filter that can generate good results regardless of whether, for example, the dog is in profile or lying down.

(a) (b) (c)

Figure 2.12: Images of a French bulldog in different positions: (a) lying down (b) standing and (c) in profile.

2. Along the same lines, a large kernel will be more useful when the information in the image is distributed globally, while a small kernel size will be more convenient if the information is distributed locally, that is, concentrated in sections of the picture.
3. In addition, one of the factors to evaluate in a CNN is the computational cost. In general, the use of smaller or larger kernels will imply a lower or higher computational cost, since they determine the number of convolutions performed in each image.

In this context, Szegedy et al. introduce in 2015 the GoogLeNet architecture, also known as Inception [90]. GoogLeNet makes use of so-called inception blocks to overcome the aforementioned problems, achieving state-of-the-art results in classification challenges with the ImageNet dataset. Inception blocks are based on computing several convolutions in parallel, with different kernel sizes, to later concatenate the results of each one. The module introduced in [90] proposes three different convolutions with 1x1, 3x3 and 5x5 size kernels, in addition to a max pooling. Figure 2.13 shows the simplest inception block, known as the naive version.

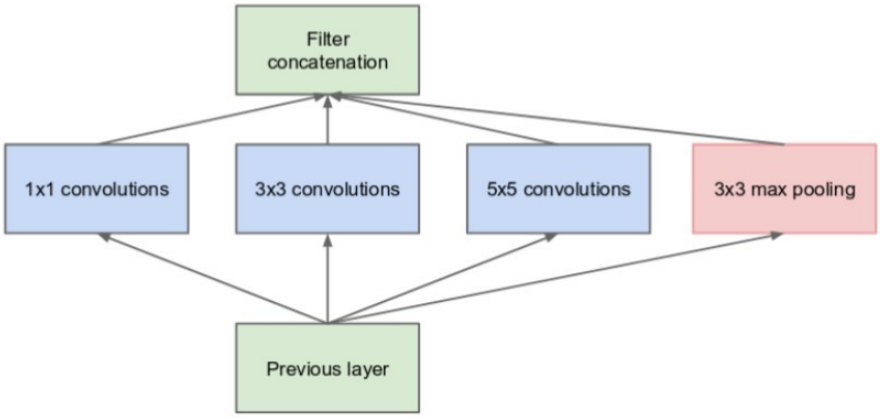


Figure 2.13: Inception module, naïve version. From *Going deeper with convolutions* [90], by C. Szegedy et. al, 2015.

One of the problems raised was that CNNs are computationally very expensive. GoogLeNet proposes to decrease the complexity of the inputs by adding a previous convolution with a 1x1 kernel. This convolution iterates for all pixels in the image, so if, for example, the input image has dimensions 96x96x3, the output will be 96x96x1. Thus, to reduce the number of inputs, this 1x1 convolution is incorporated before the layers with 3x3 and 5x5 convolutions, and after the max pooling layer. The final module is the one shown in Figure 2.14.

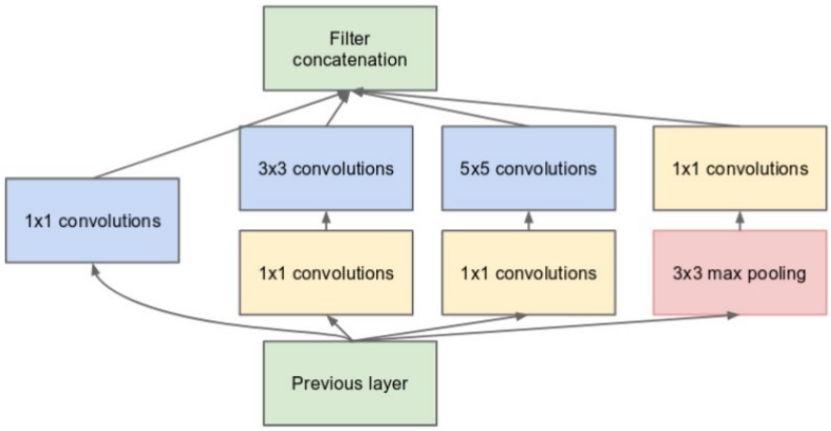


Figure 2.14: Inception module with dimension reductions. From *Going deeper with convolutions* [90], by C. Szegedy et. al, 2015.

The success of Inceptionv1 led to the development of later versions. In [89] Inceptionv2 and Inceptionv3 were presented, which proposed improvements over their previous version. Inceptionv4 was introduced in [88]. Precisely in [88] the Inception-ResNet architecture is also developed, which combines the inception blocks with the residual blocks seen in Section 2.2.4.

In short, the last sections have explained how CNNs are structured. Its basic parts have been detailed, as well as accessory modules used to create more sophisticated networks, with better classification power. In the next section, transfer learning and fine-tuning are introduced, two extremely useful techniques applicable to virtually any Computer Vision problem.

2.3 Transfer learning and fine-tuning with CNN pre-trained architectures

Traditional machine learning advocates for the creation of unique and particular models that are specific to the problem they are trying to solve. The operation of these systems, however, is conditioned on having databases large enough to train the model. Furthermore, this requirement is even more prevalent the more parameters the model has, that is, the deeper the CNN is.

However, human beings do not function in this way. They transfer knowledge about other domains when new challenges arise, so that they never start from absolute ignorance. In order to emulate this behavior, transfer learning was born from pioneering works such as those of Pratt from 1992 in [66] and, more in depth, those published in the Machine Leaning journal in 1997 [67]. Transfer learning consists of exploiting machine learning models that present good results in specific problems for our particular problem. In addition, one of its main applications is in deep learning: inheriting architectures from CNNs that have achieved state-of-the-art performances in recognized databases – for example, ImageNet – as feature extractors, to later add a classifier – the top-level – based on a multilayer perceptron that is particular to our problem, for example, when defining the number of categories. In this way, the weights of the convolutional blocks that extract the contours, figures, textures or colors from the images are maintained, and the top-level layers are exclusively trained.

A strategy within transfer learning is fine-tuning. In this approach, not only the top-level layers are trained, but also the deeper layers of feature extraction. The reason is that it is not enough to add a different top-level classifier to particularize a problem, but rather that the last layers of the network must necessarily be different, since the specific features that are extracted are specific and particular to each problem. For example, if the inherited CNN has shown excellent results in classifying dog breeds, the latest convolutional blocks are most likely optimized, for example, to identify whether the image is of a very furry or a little furry dog. If we want to design a network capable of differentiating between types of wooden furniture (chair, table, wardrobe, etc.), these last convolutional layers must be re-trained and optimized, since the patterns to be recognized now will be, for example, certain specific textures or shapes in the wood.

In short, the use of fine-tuning and transfer learning can overcome problems related to the lack of images or slow training. In this section, we introduce some of the best-known networks in the CNNs literature, which obtained state-of-the-art results in the classification of the ImageNet database. ImageNet is a database with more than 14 million annotated images, belonging to more than 20,000 categories. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is held annually, where the leading experts in deep learning and Computer Vision propose innovative models that compete to obtain the best results in this database. The competition provides over 1.2 million images from 1,000 categories. The ILSVRC has allowed a rapid development in the theory of CNNs, for example, in the presentation of novel techniques such as residual blocks or inception blocks [74]. Some of the pre-trained networks in ImageNet are summarized below: VGG, Inception, ResNet, DenseNet, MobileNet and Xception.

2.3.1 VGG

VGG architecture was created by Simonyan and Zisserman, members of the Oxford Visual Geometry Group, in 2014 in their paper *Very Deep Convolutional Networks for Large Scale Image Recognition* [85]. It is considered the successor to AlexNet [48], the network developed by Krizhevsky et al. that revolutionized the world of CNNs in 2012.

VGG stands out for its simplicity, using a standard architecture with only 3x3 convolutional kernel layers, 3x3 max pooling and ReLU activation for feature extraction and a multilayer perceptron with two fully-connected layers and softmax activation for classification into 1,000 ImageNet categories.

Its training follows the technique called pre-training, in which shallower versions of the same network are adjusted first and then used to initialize the deeper version. However, this makes training very time consuming. In addition, VGG architectures have considerably high memory occupancies, which is also a drawback.

Finally, the difference between VGG16 and VGG19 lies in the number of layers, the first being, therefore, shallower than the second.

2.3.2 GoogLeNet

The GoogLeNet or Inception architecture was introduced in 2014 by Szegedy et al. in their paper *Going Deeper with Convolutions* [90]. Unlike VGG, the Inception architecture proposes an "exotic" structure, different from the CNNs mechanism seen so far. The reason is that it includes inception blocks to solve problems related to kernel size selection. Section 2.2.5 explains in detail what these inception blocks that characterize GoogLeNet consist of.

In addition to using inception blocks, the first version of GoogLeNet (Inceptionv1) is characterized, for example, by using two auxiliary softmax classifiers on two of the inception blocks, to compute an "auxiliary loss function". This serves to overcome the problem of vanishing gradients.

After Inceptionv1, later versions have been raised, for example, Inceptionv2, Inceptionv3 and Inceptionv4. Inceptionv3 is the one used in this work. This architecture was published in 2015 by Szegedy et al. in *Rethinking the Inception Architecture for Computer Vision* [36] and includes batch normalization, factorized 7x7 convolutions, and the use of RMSProp as an optimizer, among other things. In addition, its occupation in memory is lower than that of VGG.

2.3.3 ResNet

Another of the great advances in CNNs came from the Microsoft team led by He et al. in 2016 with the publication of the paper *Deep Residual Learning for Image Recognition* [29]. This post introduces the residual blocks to the world of deep learning, widely-tested performance technique, and ubiquitous in post-ResNet architectures. In Section 2.2.4 the residual blocks, the problems they solve and the advantages of their application are explained in depth.

The network presented in [31] was called ResNet50 since it is made up of 50 layers. The residual blocks serve as a solution for degradation and vanishing gradients. Like VGG, convolutions are mostly 3x3 kernel, however the network is much deeper. The optimizer used in its original version was SGD.

Finally, with the introduction of identity mapping in the 2016 publication *Identity Mappings in Deep Residual Networks*, performance is improved by obtaining state-of-the-art results in ImageNet image classification [31].

2.3.4 Xception

Xception architecture was introduced in 2017 by François Chollet, creator of the Keras library. The original publication *Xception: Deep Learning with Depthwise Separable Convolutions* [14] uses the Inception network as its base, replacing the standard inception blocks with **modified depthwise separable convolutions**. In fact, the name Xception comes from Extreme Inception, surpassing the network created by Szegedy et al. by obtaining higher accuracy scores in the ImageNet database.

It is therefore relevant to briefly explain what the depthwise separable convolutions consist of. These separate the standard convolution into two steps: depthwise convolution and pointwise convolution. Depthwise convolution is a convolution without changing the image depth, defined by the color space (e.g. RGB, HSV), so that the kernel moves through each of the channels separately (3 in the case of RGB). Pointwise convolution is a single convolution with 1x1 kernel, that is, it iterates for each pixel in the image. This process is repeated as many times as kernels the filter bank has. For example, if the input image has a size of 12x12x3 and the kernel 5x5, in the first step three outputs will be obtained – one for each channel – of size 8x8x1 (12 - 5 + 1 = 8) that, when put together, will result in an 8x8x3 image. In the pointwise convolution, an 8x8x1 image is obtained. If a bank with 32 kernels is used, the final size will be 8x8x32. This same result would be the one that would have been obtained by directly computing the normal convolution, however, the number of operations would have been much greater. Therefore, the separable depthwise convolution serves to decrease the computational cost.

The Xception network restates this idea and introduces the separable modified depthwise convolutions, in which the order between the depthwise convolution and the pointwise convolution simply changes. This is because it uses inception blocks, in which convolution with kernel 1x1 always precedes convolution nxn. Additionally, the Xception architecture includes residual blocks, inspired in ResNet.

2.3.5 DenseNet

In 2017 the combined effort of Conwell University, Tsinghua University and Facebook AI Research allowed the creation of DenseNet in the publication *Densely Connected Convolutional Networks* [36]. DenseNet stood out for its high classification power, beating ResNet on ImageNet, using a very low number of parameters.

The DenseNet idea is based on the residual blocks introduced by ResNet. In this architecture, all layers feed all subsequent layers, which implies that the deeper layers have extremely specific and extremely general knowledge. Empirically, this allows the layers to be much more compact.

Maintaining a structure similar to the rest, it mainly uses convolutional layers with 3x3 kernel and ReLU activation in the extraction of features. Also, it uses batch normalization to regularize and average pooling 2x2 to decrease dimensionality. Finally, the classification is based on the softmax function.

2.3.6 MobileNet

Finally, another network that achieved state-of-the-art results with ImageNet in 2017 is MobileNet. The paper *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications* [35] published by Howard et al. introduces this network as a very powerful tool for Computer Vision mobile applications, due to its small size and complexity. The model is based on three techniques: separable depthwise convolutions, width multiplier (α) and resolution multiplier (ρ).

Regarding the separable depthwise convolutions, as explained in the Xception network (Section 2.3.4), they considerably reduce the number of operations by separating the standard convolution in two steps. On the other hand, width multiplier controls the width of the input in each of the layers. If the value of α is one, there will be no compression. Obviously, with lower values of α the accuracy obtained will decrease. Finally, resolution multiplier ρ controls the resolution of the image. Therefore, the computational cost can be reduced at the cost of a decrease in resolution and, therefore, a loss of information. Apart from these particularities, MobileNet maintains a structure with ReLU as activation and batch normalization to regularize.

MobileNet has been shown to deliver excellent results on both complex and simpler problems, thanks to its ability to adapt using the α and ρ value.

In short, these sections have served as a summary of some of the most popular CNN architectures. Fine-tuning is applied to these, so that they customized to any problem, obtaining generally excellent results.

Chapter 3

Methodology

3.1 Materials, apparatus and procedures

This section describes the materials, apparatus and procedures followed and used in the development of the experimental part of the work. It begins with an illustration of the materials used, that is, the hardware and software needed. Next, the database containing the images is explained, that is, their format and the process that was carried out in order to obtain them, as well as some relevant considerations. Finally, the meaning and calculation of the metrics used to evaluate the project results are justified.

3.1.1 Equipment and programming environment

The project was carried out on a computer with an 8th Generation Intel® Core ™ i7-8750H processor with six cores, with 16 GBytes of DDR4 RAM running at 2666MHz. Moreover, the hardware includes a NVIDIA GeForce GTX 1060 graphics card. The operating system used is Windows 10 64-bit.

We have chosen to implement the different models proposed in Google Colab mounted on Google Drive, to facilitate the transfer of files. Google Colab is a free environment based on the open source project Jupyter Notebook that allows its users to run and implement machine learning models in the cloud. Jupyter is an interactive environment that allows Python code to be executed dynamically, acting as a client-server application. In this way, Google Colab allows users to take advantage of the simplicity of Jupyter using Google Virtual Machines (VMs), to which the computer connects remotely. Google Colab offers both Tensor Processing Unit (TPU) and Graphical Processing Unit (GPU) as computing resources. In our case, we benefit from GPU-acceleration as it is supported by our code. Specifically, the GPU used with Google Colab is the NVIDIA Tesla P100 with 16 GBytes of RAM. The following picture, taken directly from Google, pictures a typical Google Cloud infrastructure.

Figure 3.1: Picture of Google's Cloud TPU v3 Pods. Retrieved from *Google Cloud. Cloud TPU*, by Google, `https://cloud.google.com/tpu?hl=es-419`.

The programming language used will be Python 3.7, using the high level framework specialized in deep learning Keras, specifically the version 2.3.1, and using TensorFlow 2.2.0 as backend. In addition, other libraries are required such as Numpy (version 1.18.4) for scientific computing, Pandas (version 1.0.3) for data manipulation and analysis, OpenCV (version 4.1.1) for Computer Vision, and Scikit-learn (version 0.23.1) for data pre-processing and computation of machine learning specific metrics.

3.1.2 Characterization of the database

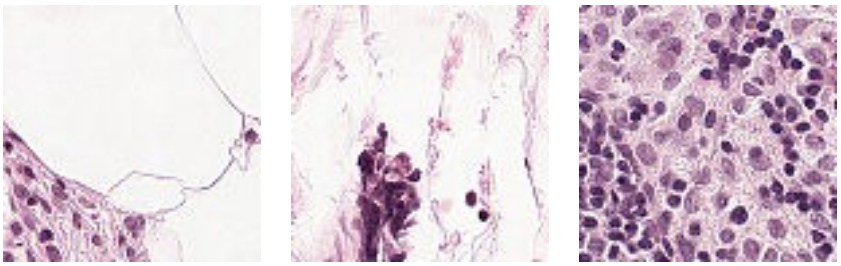
The database used comes from a Challenge organized by Kaggle, a subsidiary of Google LLC. Kaggle is a community specialized in machine learning, where experts and amateurs can share knowledge and keep themselves updated with the newest trends. However, Kaggle's main objective is to organize machine learning competitions where users from all over the world compete to solve a problem. The problem that is addressed in this document can be found in [94], and consists of, as explained on several occasions, identifying metastatic tissue in patches of sections of axillary lymph nodes, stained with H&E. The Kaggle database comes from another database known as PatchCamelyon (PCam) [95]. The difference between the two databases is due to the fact that the Kaggle database has eliminated duplicated images that PCam contained. Also, PCam derives from the original database: The Camelyon16 Challenge.

The Camelyon16 Challenge [96] was a Grand Challenge organized in 2016 by the International Symposium on Biomedical Imaging (ISBI) in which research groups from around the world competed to develop a machine learning algorithm that was capable of identifying metastases in axillary lymph nodes of women with breast cancer. Some notable participants were Harvard Medical School or MIT. The paper [22] details the conditions of the competition. However, in this section it is only relevant to mention, without going into far too many details, how the images were obtained. The original Camelyon16 Challenge dataset was made up of 400 WSIs (Whole Slide Images) obtained from patients at Radboud University Medical Center (RUMC) and University Medical Center Utrecht (UMCU). RUMC images were obtained with a digital scanner Pannoramic 250 Flash II from 3DHISTECH, while UMCU images were obtained with the NanoZoomer-XR Digital slide scanner C12000-01, produced by Hamamatsu Photonics. The

images were classified by expert pathologists. Furthermore, the categories – cancer and non-cancer – are balanced following the proportion 50/50.

Since the Grand Challenge WSIs are considerably difficult to manage, the PCam is a simplification of it. In PCam the images were converted to the HSV color space, blurred and passed through a filter that eliminated those whose saturation was less than 0.07. Each of the WSI was sampled obtaining several 96X96 pixel patches from each WSI, and saved in .H5 format. Each patch is marked with an annotation that takes the value of "1" if the central region of size 32x32 pixels contains metastatic tissue and "0" otherwise. The area outside the 32x32 section does not influence the annotation, it is simply used to facilitate the use of some models that do not apply zero-padding. The sampling was done by iteratively choosing a WSI and selecting a patch within it, that could have a positive (cancer) or negative (non-cancer) annotation with a probability of p. For consistency with the original dataset, the p was selected such that the ratio of positives to negatives was 50/50. The annotations for each image are saved in a .csv file identified with the image to which they refer. The final dataset consisted of 262,144 images for training, 32,768 for validation and 32,768 for test.

The Kaggle challenge, in short, further simplifies PCam. The first difference is that, as mentioned, it eliminates the duplications that appeared by the probabilistic sampling in PCam. In addition, the format of the images is .tif and not .H5, therefore, it facilitates even more their handling. Regarding the rest, it is identical to the PCam database: image size, annotation system, etc. Kaggle also facilitates the split between images: 220,025 for both training and validation and 57,458 for test. However, the whole test set available in Kaggle was not labelled and only the 220,025 set was left available for all training, validation and test. Figure 3.2 shows some of these images with their respective annotations.



(a) Class "0" (b) Class "1" (c) Class "0"

(d) Class "0" (e) Class "1" (f) Class "1"

Figure 3.2: Six randomly chosen images from the training set with their respective labels. Class "1" refers to tissue that metastasizes, while Class "0" refers to healthy tissue. Retrieved from *Histopathologic Cancer Detection Dataset*, by B. Veeling, `https://www.kaggle.com/c/histopathologic-cancer-detection/data`. Copyright by Kaggle

Thus, a first partition was made between training and test, assigning 187,000 images to the first and 33,025 to the second, selected at random. Therefore, an adequate proportion of 15% is maintained for the test. The first dataset, in addition, uses in all the presented models the 20% for the validation, that is, 37,400 images. Figure 3.3 illustrates the proportion of images for each class in the two main sets.

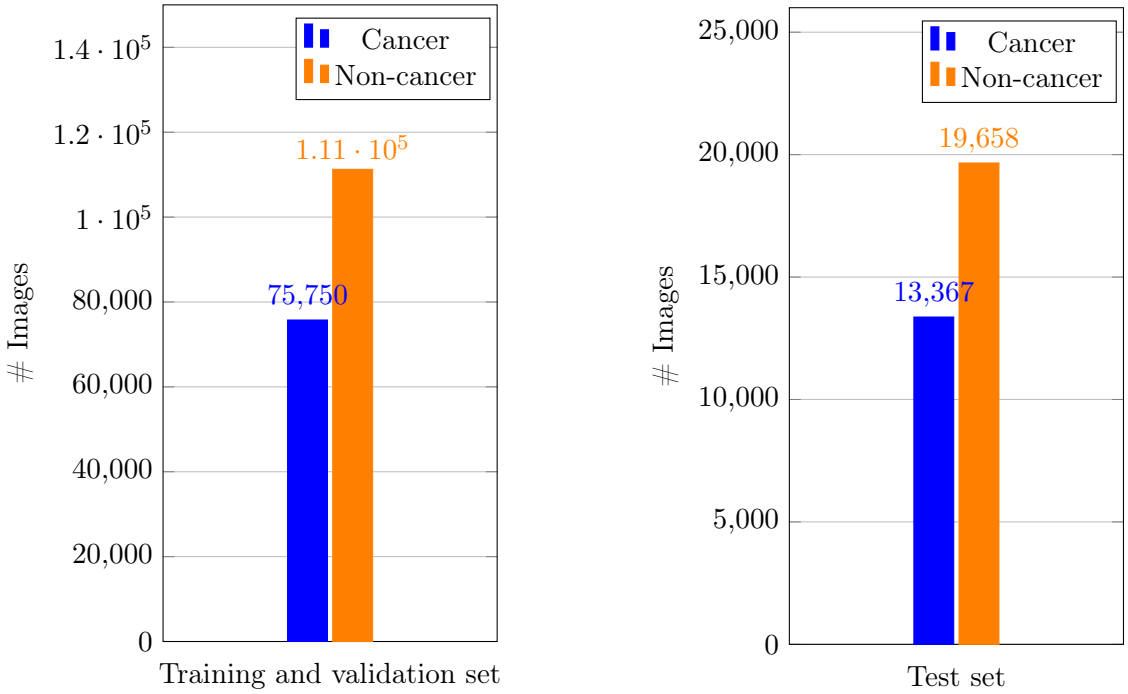


Figure 3.3: Distribution of the classes (cancer and non-cancer) in the training and validation set and the test set.

It is observed that they are somewhat unbalanced, in favor of the "Non-cancer" class in the two sets, with approximately 60% in both cases.

3.1.3 Description of the metrics used

Chapter 4 presents the results of the predictive model, both for validation and for test. Therefore, it is important to define the metrics that will quantify the degree of success of each of the models.

First, the following simple concepts can be defined as they will serve as indicators within the metrics themselves. These are the types of solutions that each classification can lead to, and they are as follows:

True positive (TP): the model detects metastatic tissue in the section and, indeed, the lymph node presents it.

False positive (FP): the model detects metastatic tissue in the section, but the lymph node has only healthy cells.

True negative (TN): the model does not detect metastatic tissue and, indeed, the lymph node presents only healthy cells.

False negative (FN): the model does not detect metastatic tissue in the section, but the lymph node does present it.

These indicators can be summarized in what is known as the confusion matrix, which is exemplified showing the four alternatives graphically in Figure 3.4.

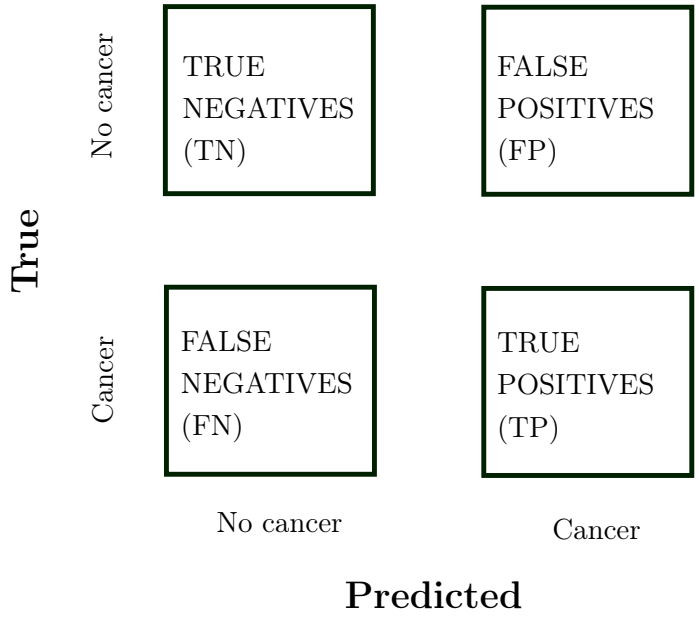


Figure 3.4: Confusion matrix with the possible solutions in our problem.

Furthermore, using these indicators, more complex and informative metrics can be formulated [42]:

Accuracy. It is the most intuitive metric and the one generally used in validation. It is calculated using the quotient between the correctly classified categories and the totals. Its main limitation is the lack of information: it is an excessively generic metric and does not give information on how powerful the CNN is to identify metastasis or healthy tissue. Obviously, the higher this value, the better, reaching the unit if all the predictions have been correct.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.1}$$

Precision. Defines the power of the model to correctly classify a particular category. In the present work, the main interest is to detect in which patches metastatic tissue exists. Therefore, the precision will quantify the proportion of images that actually presented it over the total images that have been classified as such, including the erroneous ones. Again, the higher the value, the better. In the best case, FP=0 and therefore the precision will be the unit.

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

Recall. It quantifies the number of correct predictions of a certain category out of the total of those that really belong to that same category. In this case, it measures the images that

the model has correctly classified with metastasis over the total images that actually present it. Recall is also known as true positive rate or sensitivity.

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

F1-Score. It is a weighted average between precision and recall, so it takes into account both false positives (FP) and false negatives (FN). It is a less intuitive measure but more used than accuracy when evaluating and comparing various models.

$$F1 - Score = 2\frac{Recall * Precision}{Precision + Recall} \tag{3.4}$$

ROC curve. The ROC curve is one of the most relevant metrics for machine learning. It confronts the true positive rate (y axis) with the false positive rate (x axis), the latter being understood as the complementary probability of the **specificity**.

$$Specificity = \frac{TN}{TN + FP} = 1 - FPR \tag{3.5}$$

Where FPR is the false positive rate. The ROC curve is an excellent tool for observing the ability of the model to separate between classes. To deepen about this measure, the interested reader is referred to [103].

Area Under Curve (AUC). After studying the ROC curve, AUC is defined as the area under it. Therefore, it is possible to quantify with a metric the power of the model to separate between classes. As expected, the higher the AUC, the better the model.

3.2 Description of the proposed predictive models

This section describes in detail all the proposed Convolutional Neural Networks. In the first part, the networks from scratch are exposed, that is, those designed by the author in its entirety and that have not benefited from the use of transfer learning or fine-tuning, but have optimized their parameters from scratch. In the second section, on the contrary, pre-trained network architectures are illustrated to which fine-tuning has been applied to particularize them to our problem. Specifically, the inherited CNNs are versions of those exposed in Section 2.3: VGG19, Inceptionv3, ResNet50, Xception, DenseNet201 and MobileNet. The weights of these have been adjusted in the ImageNet dataset.

3.2.1 Convolutional Neural Networks from scratch

Here are explained three CNNs designed from scratch, each with a higher level of sophistication than the previous one. In this way, the first network will be limited to incorporating the elemental layers and techniques, without including any exotic structure. This is, for example, convolutional layers, dropout layers, the use of batch normalization and multilayer perceptron for classification. The second network also includes the use of residual blocks. It draws from the architecture of the first network, which is given greater depth thanks to the use of this technique. Finally, the third network incorporates the inception blocks. An architecture based on a ResNet-Inception

combination was chosen instead of integrating the inception modules in isolation, since it obtained better results. Therefore, the details of each network are specified below. However, in a first part, the characteristics that the three networks have in common are discussed, for example, the use of data augmentation, the partitioning of the dataset or the pre-processing used.

It is relevant to highlight that the networks from scratch exposed in this section represent less than the 10% of the total models tested in the experimental phase. During this phase, up to 35 models were proposed, varying the optimizers, different structures and combinations of residual blocks and inception blocks, or top-levels. In short, the parameters and architectures of the three models described below were empirically chosen based on criteria of classification capacity and applicability to real environments.

First of all, it is necessary to mention what characteristics the three CNNs from scratch share. On the one hand, basic pre-processing has been applied in all of them, both to images and annotations. A factor of 1/255 has been applied to the images, so that the intensity of each pixel is described by a number between 0 and 1 (the RGB space describes the intensity of each color with 8 pixels, that is, with a value between 0 and 255). The reason is given by LeCun himself in [54] and is that simply rescaling makes the back-propagation algorithm more efficient, with a faster convergence. The annotations, on the other hand, have been transformed by means of a binarization, that is, one-hot encoding has been applied to them. Thus, the categorical labels "0" (non-cancer) and "1" (cancer) become 01 and 10, respectively. In addition, data augmentation has been used to generate synthetic images in all three models. The manipulations used have been rotations, zooms, flipping and cropping. It is important to mention that the implementation of data augmentation in TensorFlow discards the original images, therefore, only the synthetic ones are used in training. Logically, in the test no manipulation is applied to the images. The loss function is binary cross-entropy in all cases (See Appendix B). Finally, the partitions of 15% of the total images for the test and 20% of the training for validation are maintained, in addition to a batch size of 32 both in training and test and a total of 50 epochs. The batch size and the number of epochs have been chosen since they empirically show satisfactory results with all the proposed models. In order to make a consistent comparison, these values have been maintained for all CNNs, but in any case, in all of them, the training times were reasonable, in addition to showing a tendency to stabilize with respect to the accuracy scores and the losses in the last epochs. This implies that, in any case, the networks were sufficiently well trained. It is likely that if a higher number of epochs had been chosen there could have been overfitting problems.

Model from scratch #1

Model from scratch #1 is illustrated in Figure 3.5. It is a shallow CNN, made up of three convolutional blocks. Each block is characterized by a convolutional layer, batch normalization, a second convolutional layer, another batch normalization, max pooling and dropout, in that order. The two convolutional layers of each block are identical, with the same number of kernels, kernel size 3x3, stride 1x1, valid-padding, ReLU activation, and Xavier initialization for the weights (see Section 2.2). The number of kernels is increasing for each block, being 32, 64 and 128 for the first, second and third blocks, respectively. The rest of the layers are the same in all blocks: standard batch normalization, max pooling 2x2 and dropout with $p = 0.25$. These convolutional blocks are followed by a multilayer perceptron, which classifies the extracted features. The multilayer perceptron begins by applying a flattening to the feature vector, and then classifies

using three fully-connected layers: the first with 512 neurons and ReLU activation, the second with 128 neurons and ReLU activation, and the last, for classification, with two neurons – one per category – and softmax function. The first two fully-connected layers are separated by dropout with $p = 0.25$. Finally, the optimizer used in training is the Stochastic Gradient Descent (SGD), with a *learning rate* $= 0.001$, *momentum* $= 0.9$ and using the Nesterov momentum (see Annex A). The complexity of the definitive network is characterized by 4,549,538 parameters, of which 4,548,642 are trainable, and training had a computational cost of approximately 7.46 hours.

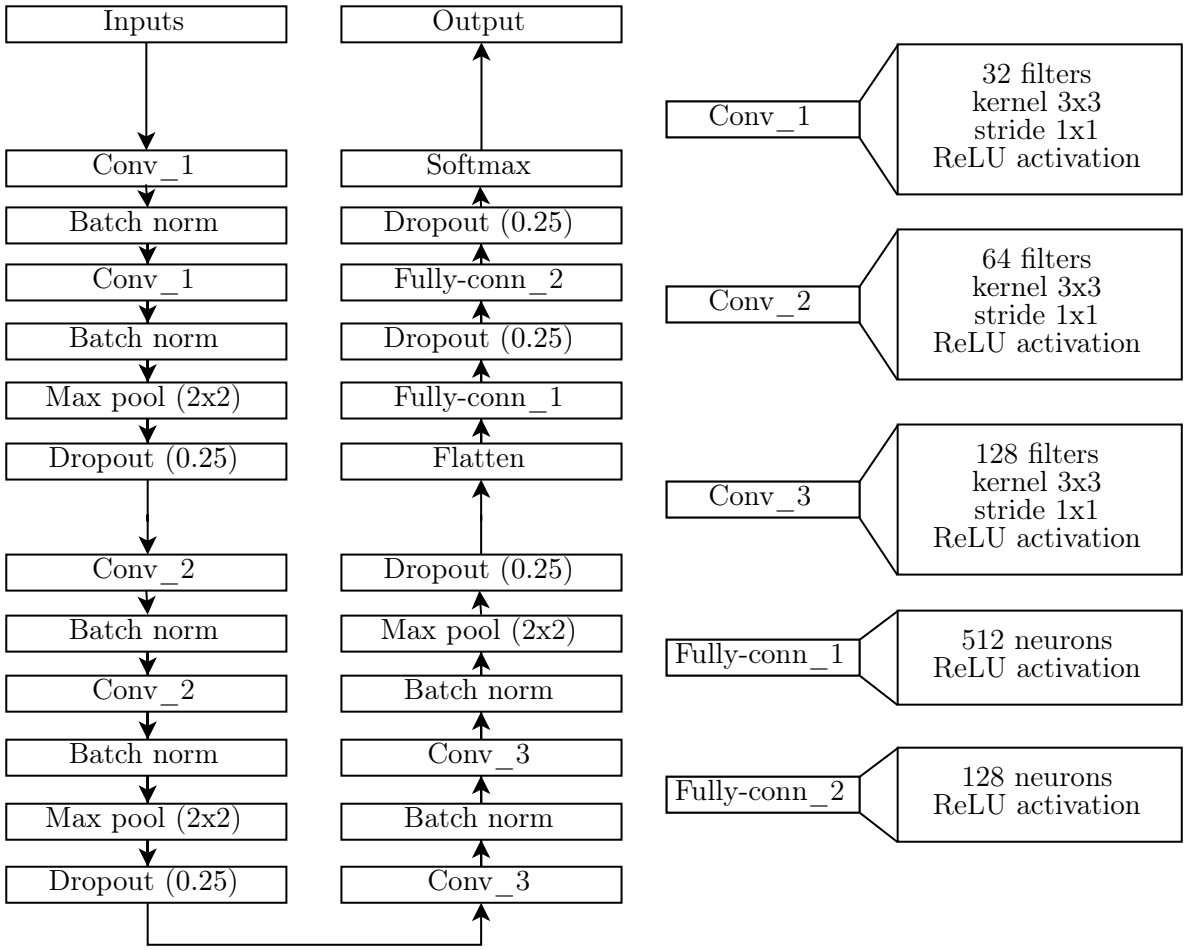


Figure 3.5: Model from scratch #1.

Model from scratch #2

Model from scratch #2 is the one shown in Figure 3.6. Thanks to the incorporation of the residual modules, this network is considerably deeper, both because there are more blocks and because each block has a higher number of convolutional layers. The model is made up of eight blocks, each consisting of a convolutional residual block followed by a max pooling and a dropout. It is recalled that in the convolutional residual block the shortcut is also convolved, unlike the identity residual block. In this way, each convolutional residual block is made up of three convolutional layers – separated by a batch normalization – and a shortcut with another convolutional layer. It is relevant to note that, in this case, unlike the previous one, batch normalization is applied before ReLU activation – which, in fact, is the recommendation in the original paper. For each block, the first and third layers have a kernel size of 1x1, while in the middle layer it is 3x3.

Same-padding and a stride of 1x1 is used in all layers and the initialization of the weights is, again, Xavier. A convolution is applied to the shortcut identical to that of the third layer of each block. The number of kernels is increasing with the depth of the network, being for each block the same for the first and second layers and the quadruple of these for the third and the shortcut. Thus, the first block has filters [32, 32, 128], the second [64, 64, 256] and so on up to the eighth with [1024, 1024, 4096]. Regarding the other layers (dropout, batch normalization and max pooling) they are identical to those of the first CNN. It has been decided to keep dropout as a regulator as it experimentally shows good performance in residual blocks [100]. After the eighth block, a Global Average Pooling (GAP) is applied to reduce dimensionality and prevent overfitting. GAP is a widely used technique on CNN that demonstrates excellent results, the interested reader can be further informed in the paper [55]. The classifier used is a multilayer perceptron identical to that of the previous CNN. Finally, the chosen optimizer is now Adam with *learning rate* = 0.0001 (see Annex A). The complexity of the definitive network is characterized by 34,403,106 parameters, of which 34,362,786 are trainable, and training had an approximate computational cost of 6.94 hours.

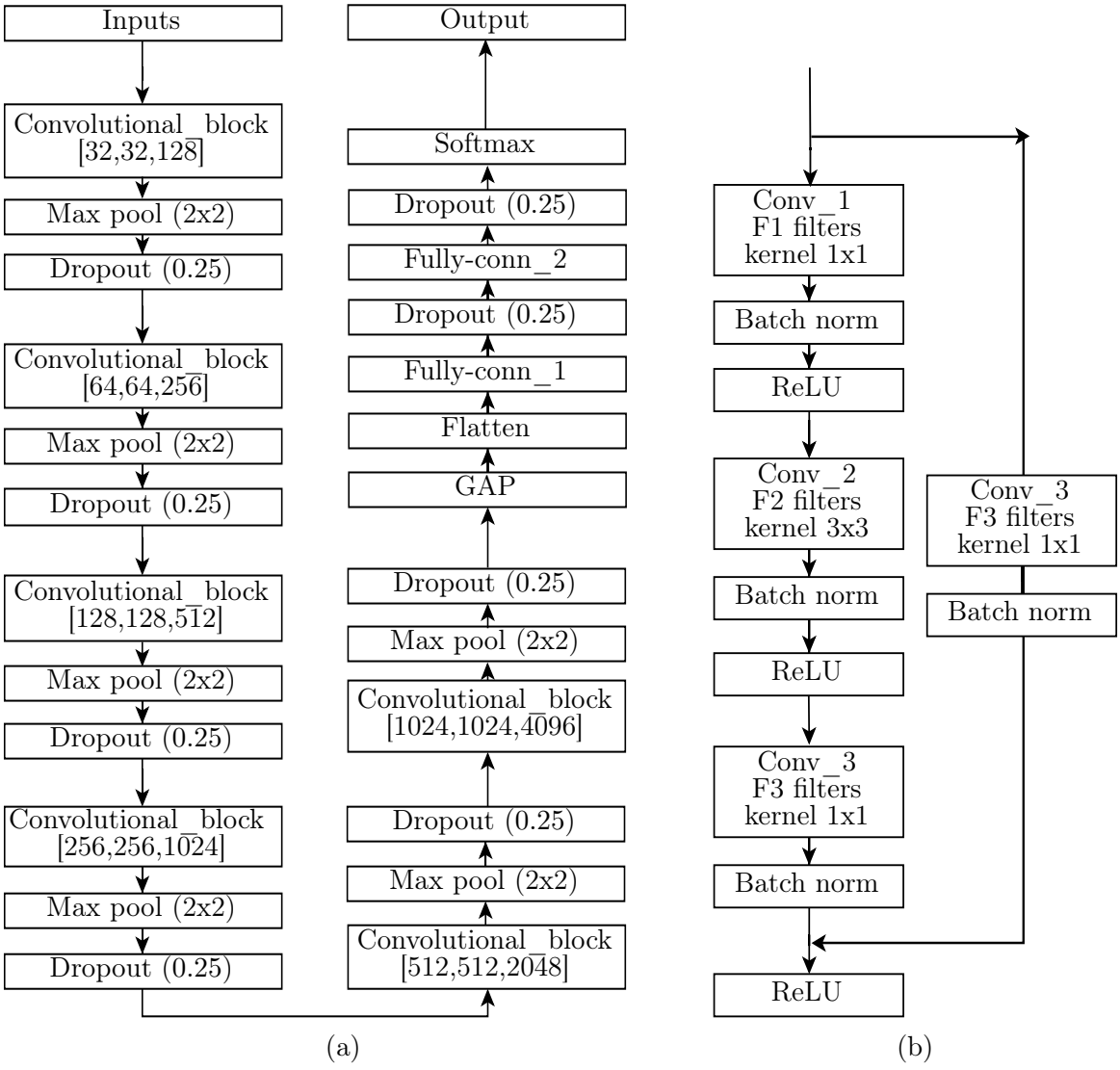


Figure 3.6: (a) Model from scratch #2 and (b) Convolutional_block [F1,F2,F3]

Model from scratch #3

Finally, Model from scratch #3 is the one in Figure 3.8. This architecture uses inception blocks, whose structure is explained in Figure 3.7. The difference with respect to the previous is the use of a ResNet-Inception module, which combines the characteristic blocks of both architectures. The ResNet-Inception module is the same as the convolutional residual block of the previous CNN, but now the convolutions are based on inception modules, that is, three convolutions in parallel of different resolutions (1x1, 3x3, 5x5) are computed and subsequently concatenated. In our case, the three parallel convolutions use same-padding, Xavier initialization, and the same number of kernels. They therefore differ only in kernel size. A max pooling is also carried out in parallel. In addition, it is recalled that before the 3x3 and 5x5 convolutions and after the max pooling, a 1x1 convolutional layer is applied, which will have the same characteristics, but with half the filters. In the proposed model, only one ResNet-Inception module is used as the first block. Therefore, blocks 2 to 8 are identical to the second model. Then a GAP and the classifier are incorporated, which, again, are identical to the those used in previous models. Finally, the chosen optimizer is Adam with *learning rate* = 0.0001 (see Annex A). The complexity of the final network is characterized by 35,172,066 parameters, of which 35,129,826 are trainable, and training had an approximate computational cost of 18.43 hours. It is precisely this high computational cost that has limited the implementation of more ResNet-Inception modules.

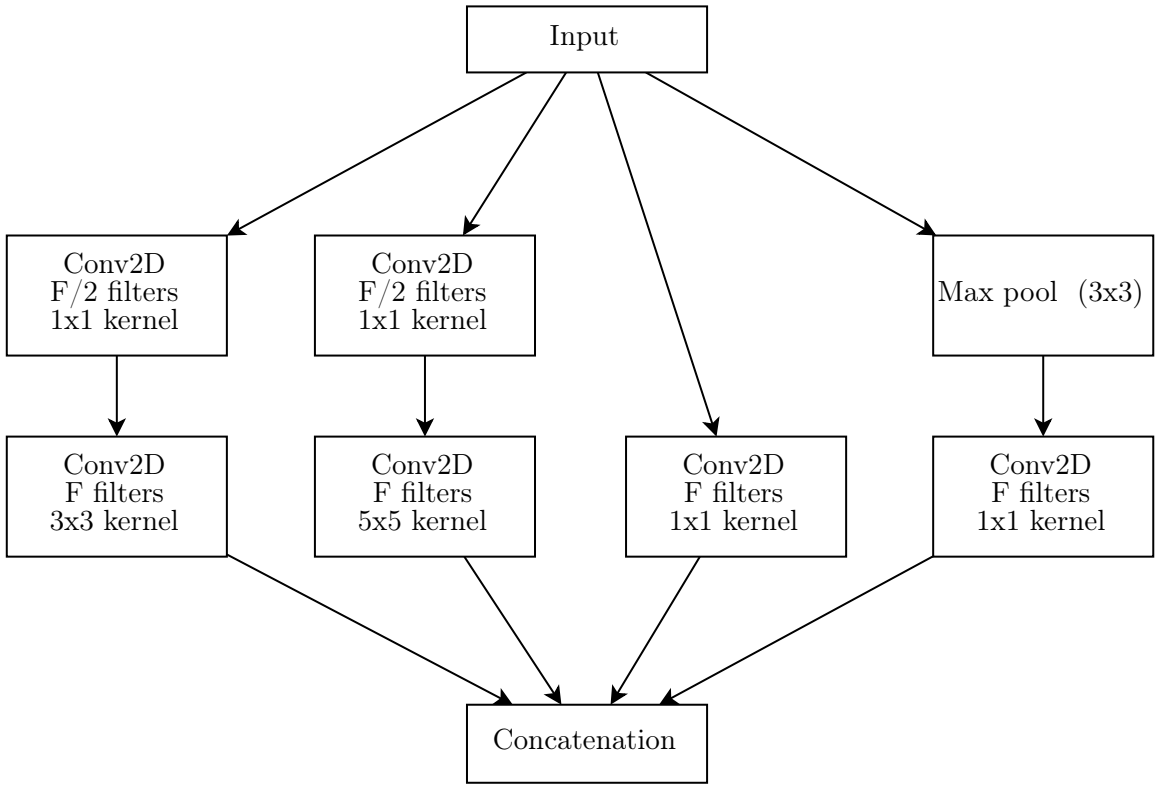


Figure 3.7: Inception_block with F filters. Structure of the inception blocks used in Model from scratch #3.

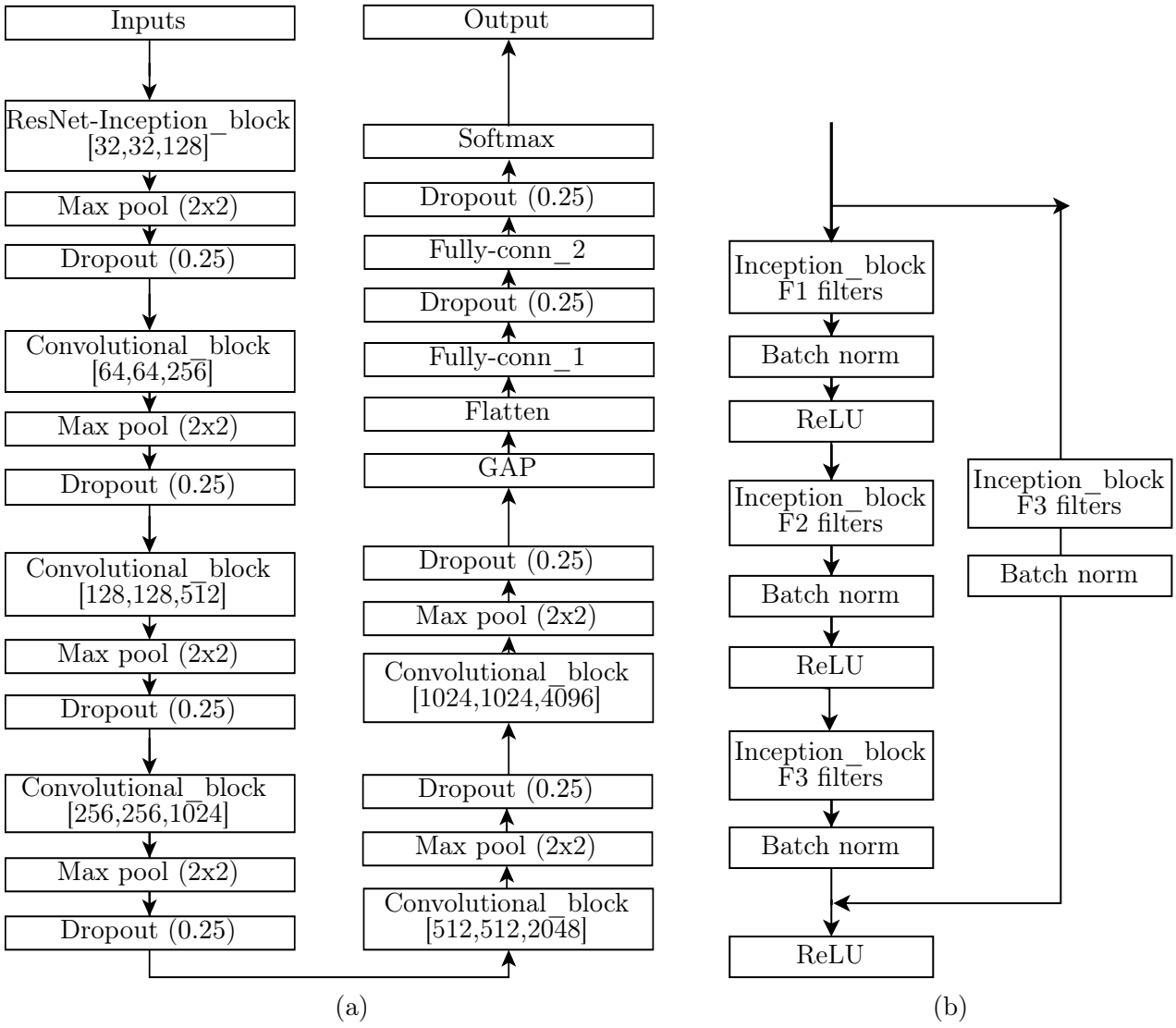


Figure 3.8: (a) Model from scratch #3 and (b) ResNet-Inception_block [F1,F2,F3]

The three implemented CNNs from scratch are thus defined. Chapter 4 presents the results of these models in our problem in particular, using specialized machine learning metrics, such as those described in Section 3.1.3. In addition, its performance is compared with that obtained with other popular networks to which fine-tuning is applied to adapt them to the present problem. The following section summarizes these networks and the design decisions that have been made for their fine-tuning.

3.2.2 CNNs based on fine-tuning of pre-trained architectures

In Section 2.3 transfer learning and fine-tuning were explained as two extremely useful techniques applicable to any image classification problem. This section presents the CNNs resulting from applying fine-tuning to some of the most popular architectures that present state-of-the-art results with the ImageNet dataset. In fact, the base-models will have the weights optimized after classifying the ImageNet images. To these models, fine-tuning will be applied, retraining some of the deeper layers and incorporating a classifier, since only the feature extraction part is inherited. Specifically, the selected models are: VGG19, Inceptionv3, ResNet50, Xception, DenseNet201 and MobileNet.

First of all, it is important to highlight how the different architectures have been configured and trained, since the process has been the same for the six models. First of all, the architecture is inherited with the optimized weights in ImageNet, not including the top-level, that is, the classifier. The classifier is incorporated into this base-model, consisting of a Global Average Pooling and three fully-connected layers: the first of 1024 neurons and ReLU activation, the second 512 and this same activation, and the third of 2 neurons – since there are two possible categories – and softmax function. The fully-connected layers are separated by dropout with $p = 0.2$. The classifier is identical on all models except the VGG19, as will be indicated. Next, a brief 3-epochs training is carried out, but optimizing exclusively the top-level. This will serve to initialize the classifier weights with reasonable values. Fine-tuning is then applied, unfreezing some of the deeper convolutional blocks of the base-model. The number of layers to be fine-tuned has been empirically obtained, performing three to five tests per model. Finally, the network is trained with the optimizer and the indicated loss function. The optimizer is specific to each model and the loss function is binary cross-entropy (see Annex B). Regarding the training images, data augmentation has been applied in the same conditions as in the CNNs from scratch of the previous section. Finally, the annotations have been transformed to one-hot, and the pre-processing for the images is the one that each of the inherited models used when classifying ImageNet, so that in each case it will be different. The batch size and number of epochs are, again, 32 and 50, respectively.

For the VGG19, fine-tuning is applied to convolutional blocks 2 to 5, leaving only the first block frozen. In other words, practically the entire network is optimized. Also, in this case the classification function is sigmoid instead of softmax and with two neurons instead of one. The reason is that with softmax and with one-neuron sigmoid it was not possible to optimize the weights during training and, therefore, the maximum accuracy score obtained was unsatisfactory, probably because the gradient descent was trapped in a local minimum. This is particularly curious, especially since with the sigmoid function it is normal to use only one neuron. In any case, the two-neurons sigmoid was the one that obtained the best results and, therefore, is the selected architecture. The optimizer used is SGD with $learning\ rate = 0.001$, $decay = 0.001/50$, $momentum = 0.9$ and without using the Nesterov momentum (see Annex A). The complexity of the definitive network is characterized by 25,269,826 parameters, of which 25,009,666 are trainable, and training had an approximate computational cost of 6.54 hours.

In Inceptionv3 fine-tuning is applied from layer 172 onwards – the model has 310 hidden layers. The optimizer used is SGD with $learning\ rate = 0.0001$, $momentum = 0.9$ and with Nesterov momentum (see Annex A). The complexity of the definitive network is characterized

by 24,426,786 parameters, of which 18,839,938 are trainable, and training had an approximate computational cost of 6.28 hours.

In ResNet50 fine-tuning is applied from layer 103 onwards – the model has 174 hidden layers. The optimizer used is Adamax with *learning rate* = 0.001 (see Annex A). The complexity of the definitive network is characterized by 26,211,714 parameters, of which 22,074,882 are trainable, and their training had an approximate computational cost of 7.78 hours.

In Xception fine-tuning is applied from layer 76 onwards – the model has 131 hidden layers. The optimizer used is Nadam with *learning rate* = 0.0001 (see Annex A). The complexity of the definitive network is characterized by 23,485,482 parameters, of which 15,868,290 are trainable, and training had an approximate computational cost of 6.67 hours.

In DenseNet201 fine-tuning is applied from layer 481 onwards – the model has 706 hidden layers. The optimizer used is SGD with *learning rate* = 0.001, *decay* = 0.001/50, *momentum* = 0.9 and without Nesterov momentum (see Annex A). The complexity of the definitive network is characterized by 20,814,914 parameters, of which 9,475,330 are trainable, and training had a computational cost of approximately 7.13 hours.

Finally, in MobileNet fine-tuning is applied from layer 44 onwards – the model has 86 hidden layers. The optimizer used is Adam with *learning rate* = 0.00001 (see Annex A). The complexity of the definitive network is characterized by 4,804,290 parameters, of which 4,513,282 are trainable, and training had an approximate computational cost of 6.55 hours.

Table 3.1 summarizes the characteristics of the six models based on fine-tuning, indicating the frozen layers, the optimizers used in their training, the total and trainable hyperparameters, and the computational cost of their training, understanding this as the time taken in the same.

Model	Frozen layers	Total layers	Optimizer	Parameters (trainable)	Cost (hours)
VGG19	6	21	Extended SGD: decay and momentum	25,269,826 (25,009,666)	6.54
Inceptionv3	171	310	Extended SGD: momentum and Nesterov	24,426,786 (18,839,938)	6.28
ResNet50	102	174	Adamax	26,211,714 (22,074,882)	7.78
Xception	75	131	Nadam	23,485,482 (15,868,290)	6.67
DenseNet201	480	706	Extended SGD: decay and momentum	20,814,914 (9,475,330)	7.13
MobileNet	43	86	Adam	4,804,290 (4,513,282)	6.55

Table 3.1: Summary of the models based on fine-tuning. The optimizer, the frozen and total layers, the trainable and total parameters and the computational cost in training are indicated..

All the models based on fine-tuning used in the classification of the present problem are thus defined. Chapter 4 illustrates the results of each of these proposals, along with those of the CNNs from scratch

Chapter 4

Results

Once the models are proposed, this chapter shows the results obtained in the classification.

It is important not to forget what the main objective of the system is, since it will serve to interpret the results obtained, especially when generalizing to the test set. It is recalled, therefore, that the proposed system aims to detect metastatic tissue in patches of axillary lymph node sections stained with H&E.

In a first section, the results obtained by the CNNs designed from scratch are exposed, while the second section will summarize the performance of models based on fine-tuning of pre-trained architectures. In both cases, the learning curves are shown first and then the value of various metrics that evaluate the ability of the model of generalization to an independent dataset. The metrics are those described in Section 3.1.3: accuracy, precision, recall, F1-score, specificity, and AUC, in addition to displaying the confusion matrices and ROC curves. Finally, in the third section, it is demonstrated the usefulness of Grad-CAM to validate the models, providing the pathologist with a tool to visualize the network activations.

4.1 Convolutional Neural Networks from scratch

In Section 3.2.1 three CNNs from scratch models are presented, each more sophisticated than the previous one, which integrate the techniques exposed throughout this document. The evaluation of these three models is developed in three parts. First, the learning curves are displayed, where the losses obtained by computing binary cross-entropy are presented together with the accuracy for each of the 50 epochs, both for the training and for the validation. Second, the confusion matrices are shown, indicating the meaning of each of the solutions. Third, the value of calculating the metrics in Section 3.1.3 is presented, as well as the ROC curves linked to each model. The analysis of these results is carried out in the Discussion (Chapter 5).

The learning curves show the losses and accuracy of the model in each of the epochs, both in training and validation. It is to be hoped that as the epochs increase, the better adjusted will be the model weights and, therefore, the lower the losses and the higher the accuracy scores. In the

case of training this will always happen since the losses and the accuracy are computed on the same dataset on which the weights have been optimized. However, in the validation the model is evaluated in an independent set and, therefore, it will serve to study if the model learns to generalize with increasing epochs or if, for example, there are overfitting problems. Figure 4.1 shows the curves of the three models from scratch.

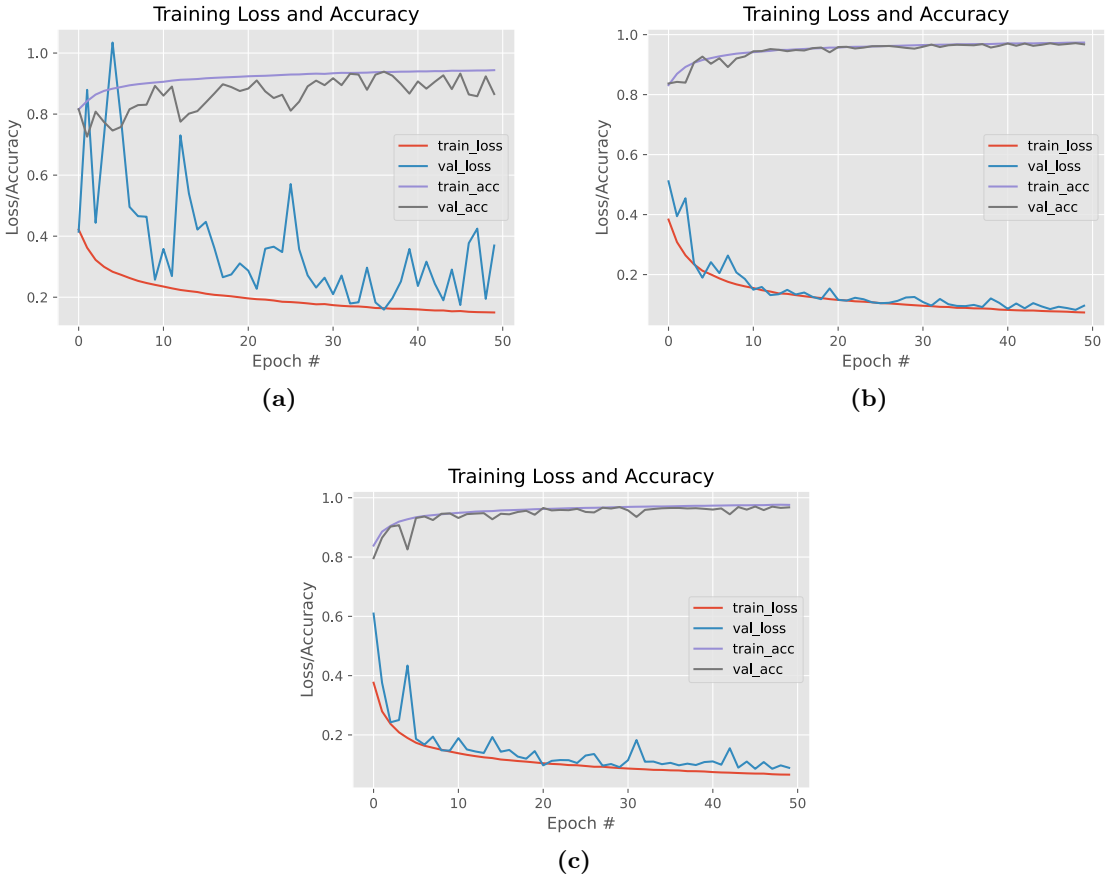


Figure 4.1: Learning curves for (a) Model from scratch #1 (Initial model) (b) Model from scratch #2 (Model with residual blocks) and (c) Model from scratch #3 (ResNet-Inception).

Now, Figure 4.2 shows the confusion matrices obtained when evaluating the models in the test set, that is, the independent set of 33,025 images. The x-axis represents the predictions and the y-axis the true labels. In each possible solution both the absolute and relative values are indicated. Figure 3.4 in Section 3.1.3 showed the equivalence of this problem to the general confusion matrix: true positive, true negative, false positive, false negative.

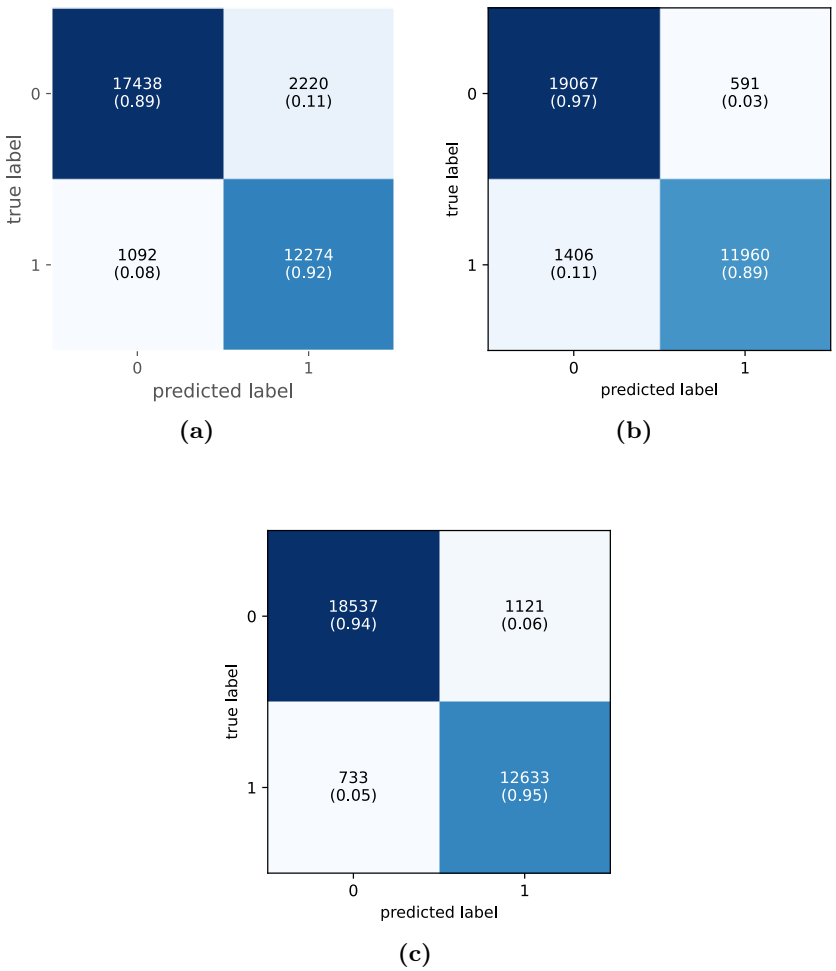


Figure 4.2: Confusion matrices for the models (a) Model from scratch #1 (Initial model) (b) Model from scratch #2 (Model with residual blocks) and (c) Model from scratch #3 (ResNet-Inception).

Finally, Table 4.1 summarizes the scores for each of the three models on the metrics defined in Section 3.1.3.

Model	Accuracy	Precision	Recall	F1-Score	Specificity	AUC
Model from scratch #1	89.97	84.68	91.83	88.11	88.71	96.40
Model from scratch #2	93.95	**95.29**	89.48	92.29	**96.99**	98.30
Model from scratch #3	**94.39**	91.85	**94.51**	**93.16**	94.30	**98.60**

Table 4.1: Figures of merit for the models from scratch.

The ROC curves can be seen in Figure 4.3. These face the true positive rate and the false positive rate. They are also used to calculate the AUC.

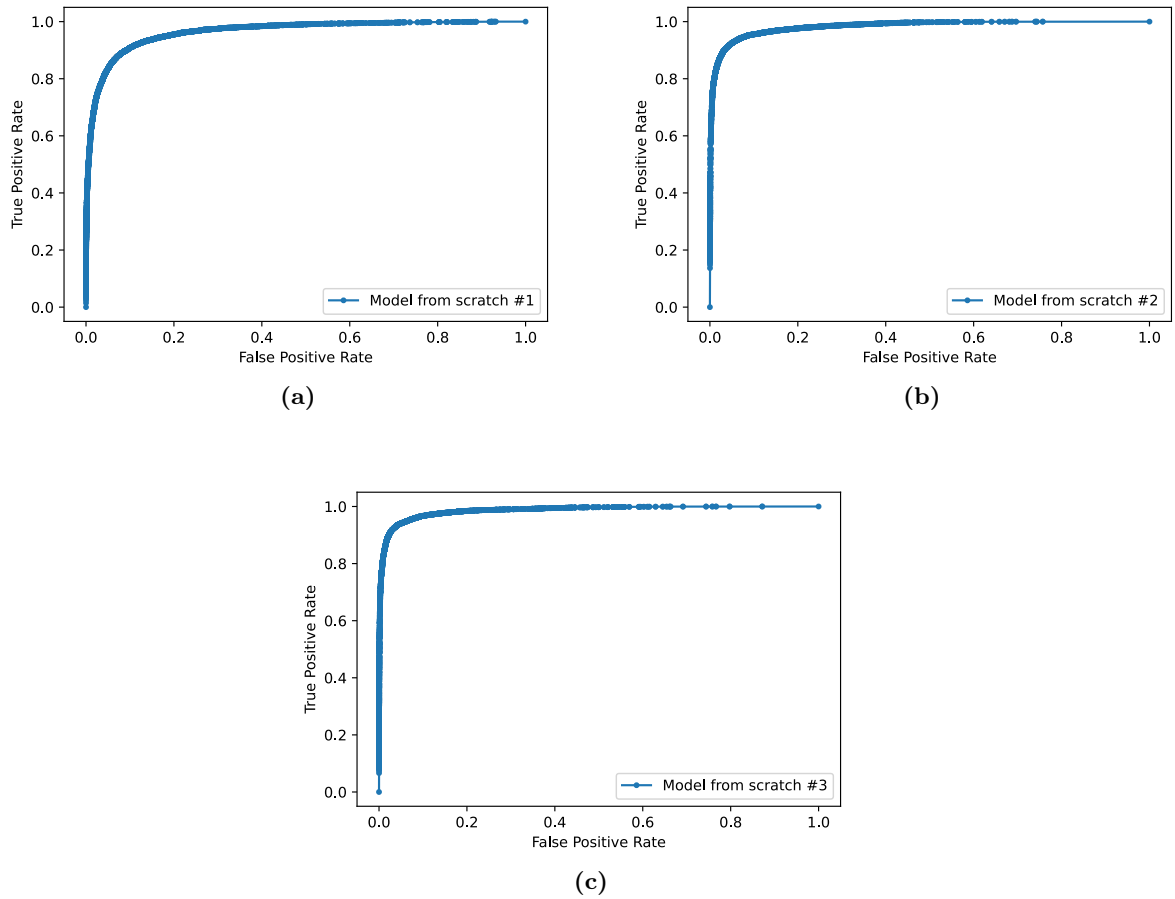


Figure 4.3: ROC curves for the models (a) Model from scratch #1 (Initial model) (b) Model from scratch #2 (Model with residual blocks) and (c) Model from scratch #3 (ResNet-Inception).

4.2 CNNs based on fine-tuning of pre-trained architectures

Similarly, this section presents the results obtained by the six pre-trained architectures to which fine-tuning has been applied: VGG19, Inceptionv3, ResNet50, Xception, DenseNet201 and MobileNet (Section 3.2.2). In this way, the learning curves and the confusion matrices are visualized, in addition to the results of the metrics and the representation of the ROC curves.

First, Figure 4.4 shows the learning curves. Again, the values of the losses and accuracy in the 50 epochs, both for training and for validation, are represented.

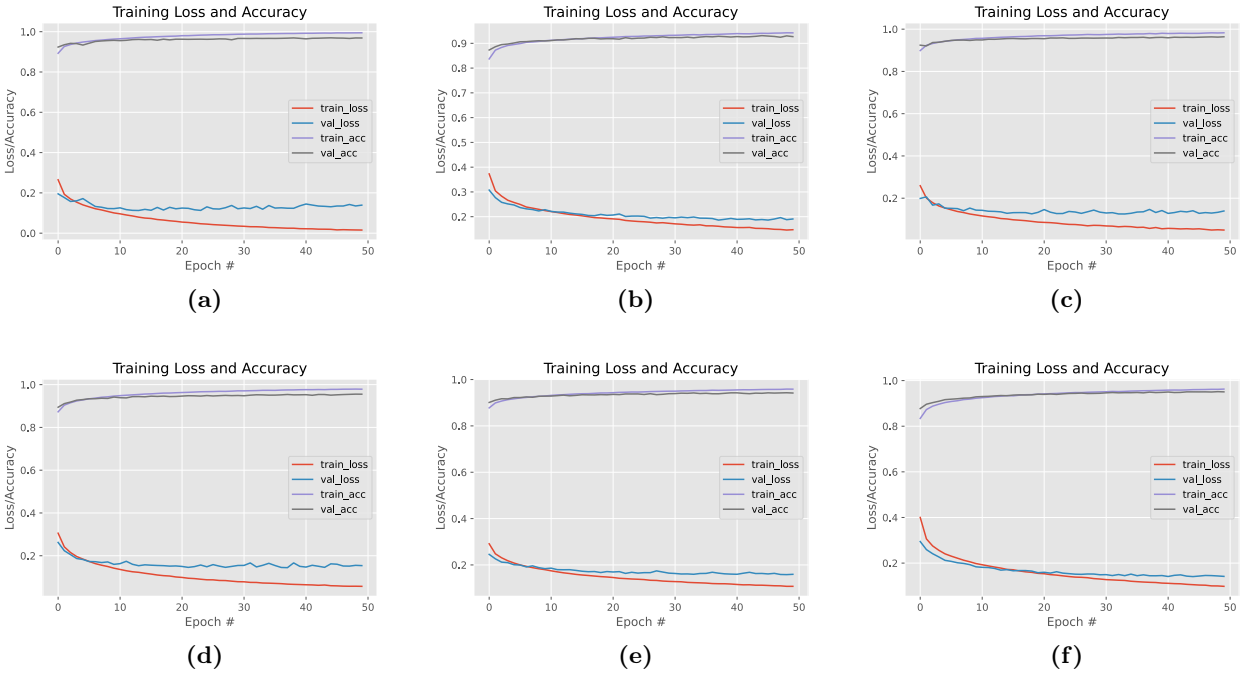


Figure 4.4: Learning curves for the pre-trained models (a) VGG19 (b) Inceptionv3 (c) ResNet50 (d) Xception (e) DenseNet201 (f) MobileNet.

Second, the confusion matrices linked to the evaluation in the test set of the six models are presented in Figure 4.5.

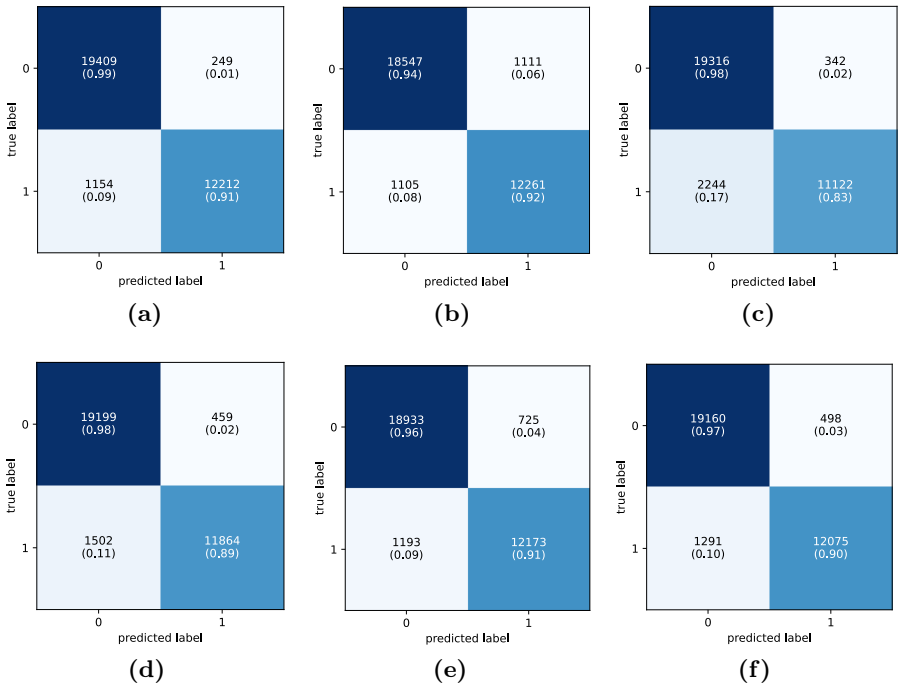


Figure 4.5: Confusion matrices for the pre-trained models (a) VGG19 (b) Inceptionv3 (c) ResNet50 (d) Xception (e) DenseNet201 (f) MobileNet.

Finally, Table 4.2 shows the metrics obtained by each model. In addition, Figure 4.6 shows the representations of the ROC curves.

Model	Accuracy	Precision	Recall	F1-Score	Specificity	AUC
VGG19	**95.75**	**98.00**	91.37	**94.57**	**98.73**	**99.10**
Inceptionv3	93.29	91.69	**91.73**	91.71	94.35	98.00
ResNet50	92.17	97.02	83.21	89.59	98.27	97.10
Xception	94.06	96.28	88.76	92.37	97.67	98.50
DenseNet201	94.19	94.38	91.07	92.70	96.31	98.60
MobileNet	94.58	96.04	90.34	93.10	97.47	98.60

Table 4.2: Figures of merit for the pre-trained models.

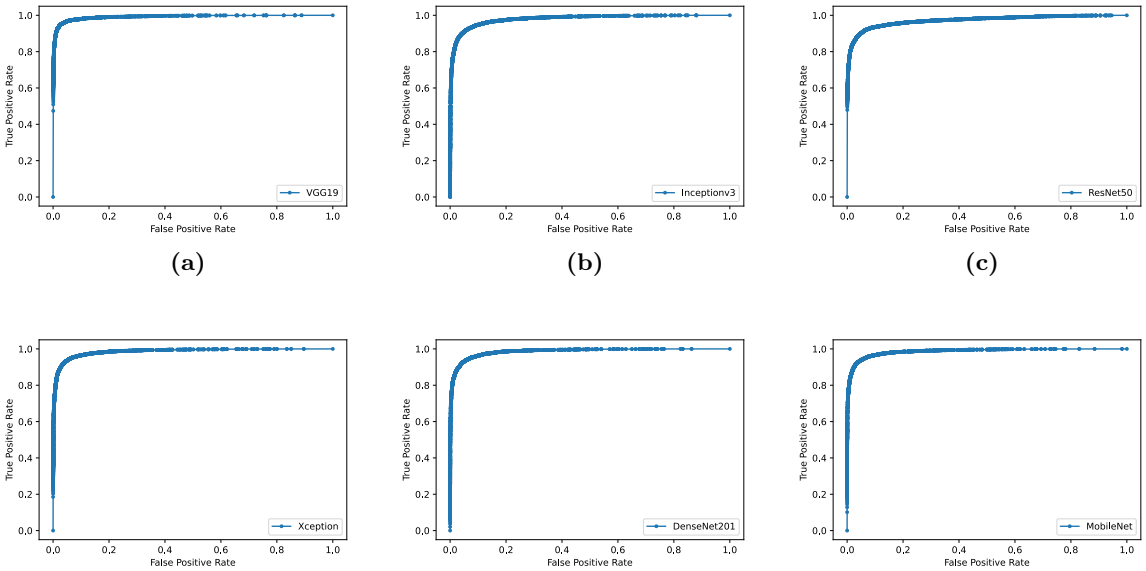


Figure 4.6: ROC curves for the pre-trained models (a) VGG19 (b) Inceptionv3 (c) ResNet50 (d) Xception (e) DenseNet201 (f) MobileNet.

This defines the behavior of all models – both from scratch and inherited networks – in training, validation and test, using learning curves and specific machine learning metrics. In Chapter 5, we discuss these results, giving them meaning in our particular problem. In this way, its classification capacity and its suitability for application in a practical environment, for example, in a hospital, are evaluated.

4.3 Visualization of the relevant histopathological patterns with Grad-CAM

One of the main problems of deep learning and, more specifically, of CNNs is the difficulty of its interpretation. CNNs are understood as black-boxes capable of solving any classification problem, which implies that on many occasions the network designer is unaware of the actual structure of the network, the activations of the neurons – which represent the identified patterns – or, simply, what the final prediction is based on.

Faced with this problem, Selvaraju et. al created the Gradient-weighted Class Activation Mapping (Grad-CAM). The foundations and operation of this technique can be found in the original paper [79]. Grad-CAM allows to visualize the activations of the network, so that the designer is able to validate its operation, checking that it correctly identifies the patterns to make a decision based on them. Therefore, it is a useful tool for debugging the system.

The implementation of Grad-CAM in our models provides us with a tool for understanding CNNs. It is logical the reluctance to use CNNs in sensitive tasks, such as the diagnosis of breast cancer, if these are presented as black-boxes, of which their operation is not really known and on what it is based to classify. With Grad-CAM, the pathologist will be able to validate the functioning of the networks, verifying that the patterns identified by them are coherent and actually lead to the identification of tissue with metastasis. Furthermore, Grad-CAM can even identify patterns that were previously unknown by the experts in this diagnosis. Therefore, the utility of this technique is maximum in sensitive diagnoses such as the one of breast cancer. Figure 4.7 shows some examples of the use of Grad-CAM in our database, specifically in the classification made by the VGG19 pre-trained network, since it is the one that presents the highest accuracy scores.

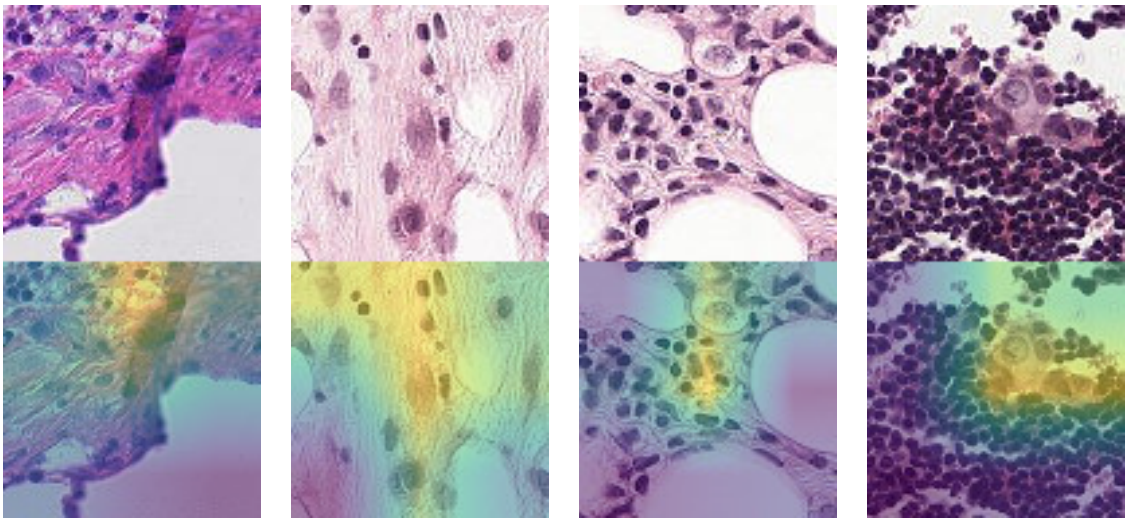


Figure 4.7: Grad-CAM of four randomly selected images in our dataset, all correctly classified as cancerous. The classifier model used is the one based on VGG19. The yellow areas are those in which the network identifies patterns to classify the image, therefore, they are supposed to be the sections of the lymph node where the metastatic tissue is located. The same exercise could be done with sections without cancer, observing the histological features identified by the network to rule out the presence of metastases.

Chapter 5

Discussion

Throughout Chapter 2, an exhaustive description of the nine proposed models has been developed: three CNNs from scratch and six architectures based on fine-tuning of pre-trained networks. In this chapter, the results are discussed. The analysis will be made of all the models simultaneously, without maintaining the separation of the previous chapter. In this way, based on the different metrics and graphs, the suitability and classification capacity of the models is defined, highlighting those that potentially would have better results in a real environment.

In addition, in a second section, the performances of these models are compared with those obtained using traditional machine learning techniques.

5.1 General discussion

Here all the results obtained in Chapter 4 are analyzed. In this way, the learning curves of the models are first discussed, identifying if any have any particular characteristic, such as the presence of overfitting. Second, the confusion matrices are studied, giving meaning to the different solutions that each model presents. Finally, the metrics obtained by each model are compared, indicating which is more suitable for certain applications.

Figures 4.1 and 4.4 showed the learning curves of the models from scratch and those based on fine-tuning, respectively. In general terms, the behavior of training and validation is adequate in the nine models: there is a tendency to increase accuracy and decrease losses, both in training and in validation. This means that the selected optimizers are able to find the absolute minimum or an adequate local minimum of the loss function, and gradually converge. It is noteworthy that the pre-trained models have more stable learning, since they start from already optimized weights. The model from scratch #1 is particularly unstable, especially in the early epochs. In none of the graphs an evident overfitting is identified, therefore, the positive effect of batch normalization, dropout and data augmentation is evident. Finally, it is identified that in some pre-trained models – such as MobileNet and DenseNet – in the initial epochs the training losses are higher than the validation losses or, what is the same, the accuracy is lower. This is due to

the fact that dropout is not applied in the validation and, therefore, it may show better results than the training itself.

The CNNs confusion matrices can be seen in Figures 4.2 and 4.5. These give a general idea about the successes and errors of each model. The two errors they can make are diagnosing a healthy lymph node with cancer (false positive) and not detecting cancer in a lymph node that does present metastasis (false negative). Both errors are assumed to be equally serious, and therefore one model will be no better than the other based on which of the two errors it makes the least. Of the three models from scratch, the one that best detects metastatic tissue is the model from scratch #2 (97%), while the one that best identifies healthy tissue is the model from scratch #3 (95%). However, in average terms the one that is most successful and, therefore, has the best applicability will be the model from scratch #3. On the other hand, the VGG19, ResNet50 and Xception pre-trained models obtained excellent results to identify cancer, with 99%, 98% and 98% respectively. Of these three, the one that best detects tissue without metastasis is VGG19 (91%). Inceptionv3 is the most accurate in this diagnosis (92%). If we take into account all the models, it could be concluded that VGG19 and model from scratch #3 have the best applicability in a real environment, presenting good results for classifying any type of lymph node section.

Finally, Tables 4.1 and 4.2 summarized the most significant metrics for the present problem. Based on these, it can be concluded that:

- The model with the best **accuracy** is the VGG19 pre-trained network, with 95.75%. Therefore, in general terms, it will be the one that best classifies the lymph node sections. However, this metric does not give information on whether it is best in detecting cancer or healthy tissue.
- The model with the best **precision** is, again, the VGG19 with 98%. Therefore, it is the CNN that makes the least mistake when it comes to detecting cancer, since it precision quantifies the ratio between the correct predictions of cancer and the total ones made with this verdict.
- The model with the best **recall** is the model from scratch #3 with 94.51%. Therefore, it is the model that has the best capacity to detect tissues with metastases. This metric only takes into account patches with lymph nodes that present cancer.
- If **F1-Score** is used (which serves as the average between precision and recall), the best model will be the one based on the VGG19 architecture with 94.57%, the model from scratch #3 occupying the second position with 93.16%.
- The model with the best **specificity** is, again, the VGG19 with 98.73%. It is the most powerful model for identifying healthy tissue.
- If we look at the **AUC** metric, the best model is VGG19 with 99.10%.

It seems that it is not possible to determine exactly which is the best model. Although the one based on the VGG19 pre-trained network dominates on most metrics, its recall is considerably low compared to, for example, the model from scratch #3. Therefore, the usefulness of each proposal will depend on the application (e.g. the model from scratch #3 is better at detecting cancer, but VGG19 makes less mistakes) and, in any case, it should be reviewed by an expert pathologist.

What is demonstrated is that the incorporation of more sophisticated techniques in the networks from scratch has allowed the creation of CNNs capable of competing with powerful pre-trained networks modified by fine-tuning. The use of residual blocks and the ResNet-Inception architecture provide models with better classification capacity and, ultimately, make them more useful and applicable to real and sensitive environments, such as cancer diagnosis in hospitals. Model from scratch #3 represents the use of these techniques, and is capable of outperforming all pre-trained networks in the recall metric.

In addition, all pre-trained networks have been found to perform generally well. Specifically, for our particular problem, the VGG19, MobileNet and DenseNet networks present very satisfactory results in practically all the metrics. VGG19 is imposed as the best in all aspects except in the recall, where it is surpassed by Inceptionv3, model from scratch #1 and model from scratch #3. It is relevant to note that VGG19 is the simplest of the networks along with MobileNet, and they are exactly the ones that seem to be most useful in the present problem. Precisely the deepest networks, such as ResNet50 and Inceptionv3, are outperformed in accuracy both by the rest of pre-trained and by the models from scratch #2 and #3. DenseNet – which is the deepest network – presents considerably good results overall. Table 5.1 summarizes the order of the nine proposals from best to worst classification capacity. For simplicity, only accuracy has been taken into account.

Model	**Accuracy**
VGG19	95.75
MobileNet	94.58
Model from scratch #3	94.39
DenseNet201	94.19
Xception	94.06
Model from scratch #2	93.95
Inceptionv3	93.29
ResNet50	92.17
Model from scratch #1	89.97

Table 5.1: Accuracy scores for all the proposed models.

In addition, another criterion to take into account is the memory occupation of each of the models. Table 5.2 summarizes the total and trainable parameters, and the computational cost of the training – in hours – of the three models from scratch and the six based on fine-tuning. Memory occupation is proportional to the number of parameters of the CNNs, therefore, it is easy to see that, in average terms, the models based on fine-tuning have a lower memory occupancy than the models from scratch – with the exception of the model from scratch #1, which is particularly simple. On the other hand, within the fine-tuning based models, the VGG19, ResNet50 and Inceptionv3 are those characterized by a greater number of parameters. In contrast, MobileNet has a significantly lower number, up to six times less than these three. Memory occupancy is a fundamental variable in certain environments, for example, if the device that uses the network has a reduced memory capacity, such as a mobile phone. In this case, the use of VGG19 could be rejected despite having the best results. Failing that, considering

the number of network parameters, MobileNet could be the best solution, as it is also the model with the second best results in terms of accuracy score.

Model	Total parameters	Trainable parameters	Cost (hours)
Model from scratch #1	4,549,538	4,548,642	7.46
Model from scratch #2	34,403,106	34,362,786	6.94
Model from scratch #3	35,172,066	35,129,826	18.43
VGG19	25,269,826	25,009,666	6.54
Inceptionv3	24,426,786	18,839,938	6.28
ResNet50	26,211,714	22,074,882	7.78
Xception	23,485,482	15,868,290	6.67
DenseNet201	20,814,914	9,475,330	7.13
MobileNet	4,804,290	4,513,282	6.55

Table 5.2: Parameters – both total and trainable – and computational cost – in hours – of all the proposed models.

Finally, through Grad-CAM we have been able to visualize network activations, so that we could identify the patterns on which a CNN – specifically VGG19 – is based to classify the images. This serves both to validate the proposed model – as the pathologist can verify its correct functioning – and to detect unknown patterns, with special interest in complex diagnoses such as that of the present work.

5.2 Comparison with other traditional machine learning techniques

One of the objectives indicated in Chapter 1.4 was to compare Convolutional Neural Networks with other traditional machine learning techniques, such as k-Nearest Neighbors (k-NN), Support Vector Machines (SVM) and Random Forest.

CNNs are imposed as the state-of-the-art solution in Computer Vision since they aspire to classify any type of problem with excellent results. However, this is at the cost of high computational times – requiring GPU-acceleration – and the need to learn through huge databases. In the Motivation (Section 1.1), it was explained that, in parallel to the present work, the thesis related to the final degree project in Business Administration and Management [1] was prepared, whose author is also Javier Abad Martínez. This thesis addresses the same problem as the current document, but proposing solutions based on traditional machine learning algorithms. Therefore, this section compares the results obtained using both approaches: traditional machine learning and Convolutional Neural Networks.

It is important to highlight that this section does not aspire to be a frontal comparison, since the experimental conditions were not the same and, therefore, it would not be fair. For example, for training in traditional machine learning techniques, we used five times fewer images, and the test images were different, which makes any comparison of accuracy score unfair. Also, the framework used in training was not TensorFlow but Scikit-learn, which does not support GPU-acceleration. GPU-acceleration means that processing speeds are multiplied by up to a factor of

x50 when compared to CPU-based processing. In short, this section is only intended to illustrate a different approach to CNNs, to highlight the advantages and disadvantages of each technique.

Comparison criteria are computational cost and classification capacity. The computational cost is evaluated through the training and test times. In this case, it has been assumed that the time is directly proportional to the number of images and that its distribution is such that it invests the same time for all the images. Thus, to obtain the values of "Training time per image" and "Test time per image", the total training time and the test time have simply been divided by the number of images in each one, that is, 187,000 and 33,025 respectively. Regarding the classification capacity, accuracy has been chosen for simplicity.

The CNNs chosen are the five with the best accuracy scores: VGG19, MobileNet, model from scratch #3, DenseNet and Xception. They are compared with the three techniques that obtained the best results in [1]. In all three, color histograms [40] are used as feature extractors, using a different classifier for each one: k-Nearest Neighbors [17], Support Vector Machines [93] and Random Forest [8]. The interested reader is recommended to delve into these techniques in the papers indicated in the references. Table 5.3 shows the values of each of the models for these metrics.

Model	**Training time per image (ms)**	**Test time per image (ms)**	**Accuracy**
VGG19	119.52	0.73	95.75
MobileNet	126,.10	0.58	94.58
Model from scratch #3	354.80	2.57	94.39
DenseNet201	137.26	1.15	94.19
Xception	128.41	0.58	94.06
Color + k-NN	1.65	5.87	89.86
Color + SVM	460.13	10.96	89.73
Color + Random Forest	2.07	0.04	89.29

Table 5.3: Training times, test times and accuracy scores for the five best CNNs and the three best models based on traditional machine learning, obtained in [1]

These data are located in Figure 5.1 and Figure 5.2. Figure 5.1 represents the time per image that each model requires in the training set to subsequently obtain an accuracy score in an independent test set. In contrast, Figure 5.2 reports the time it takes for the model to evaluate the test set and obtain the indicated accuracy.

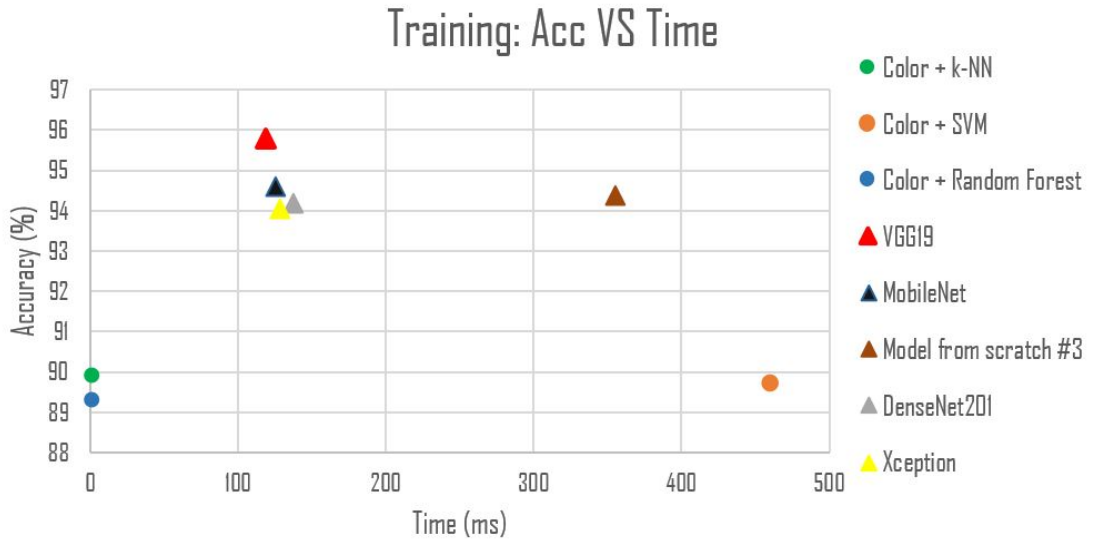


Figure 5.1: Comparison between traditional models and CNNs based on their accuracy and training times (in milliseconds). Circles represent the former, while triangles represent the latter.

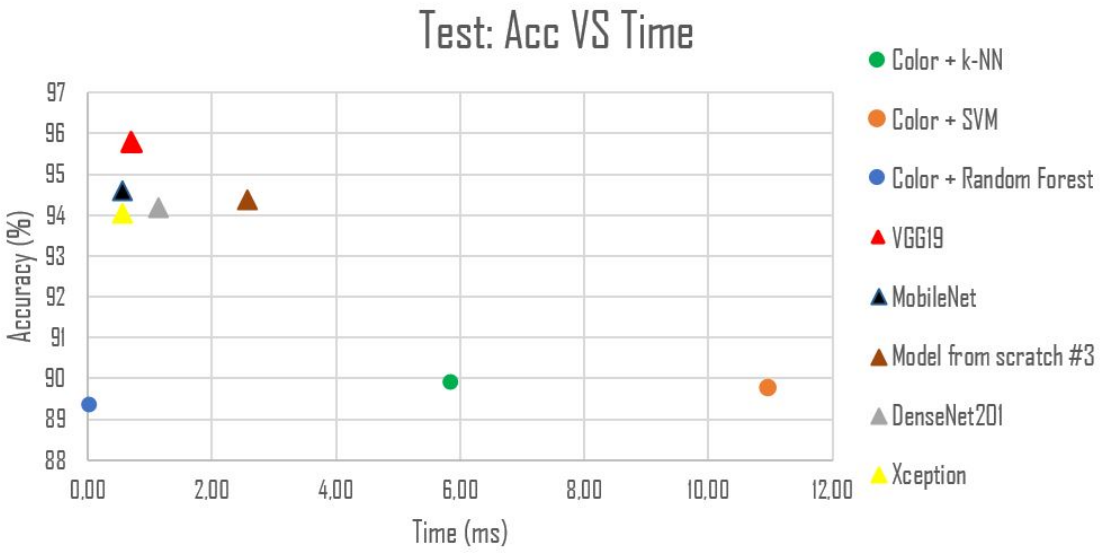


Figure 5.2: Comparison between traditional models and CNNs based on their accuracy and test times (in milliseconds). Circles represent the former, while triangles represent the latter.

Models based on CNNs are indicated with triangles. All of them obtain considerably better accuracy scores than any of the other models. Regarding training times, they are much higher than most models based on traditional machine learning, with the exception of those that use SVM as a classifier. In particular, the network from scratch has a high computational cost. This is due to the CNNs training process, based on epochs, which implies that they adjust their models as many times as epochs have been indicated by the programmer. One of the problems of this system is the danger of overfitting, which is, in fact, one of the main criticisms of CNNs. In addition, CNNs have used a much bigger database, specifically by a factor of five. Finally, the test times are reasonable, standing at better values than most traditional models.

It is recalled that, in any case, CNNs have been accelerated by GPU. If they had not, the times could have been up to fifty times higher. This would imply that the training lasted several days. This is one of the reasons that CNNs have become so popular recently: accelerated GPU-based computing. However, in environments where there are time restrictions and there is no GPU, traditional machine learning models are very likely to be useful, despite having lower accuracy scores. Similarly, special implementations of traditional machine learning models could also make use of GPUs, reducing times enormously. Furthermore, it is recalled that CNNs require large databases. Therefore, in case this resource is scarce, there can be interesting alternatives based, for example, on k-Nearest Neighbors, Support Vector Machines or Random Forest.

It can be concluded that the choice of model will depend on the environment and the application. With GPU-acceleration and many images, CNNs are most likely the best solution to a Computer Vision problem, however, overfitting must be controlled and tests must be done in independent test sets to fully validate the models. On the contrary, when there is a temporary restriction – or GPU acceleration is not possible – or the number of images is low, and the accuracy score is not relevant – for example, if it is going to be reviewed later by an expert – it is possible that traditional machine learning models are more convenient.

5.3 Limitations

This section highlights the assumptions, possible biases and simplifications taken in the development of the project. These come both from the database itself and from the techniques used and, therefore, limit the conclusions and generalizations that have been reached.

Regarding the original database, it had several limitations, many of them related to the simulation of the pathologist's exercise as the classifier of the patches. The database has been enriched with images that do show metastatic tissue in order to have a significant number of images in each of the categories. However, this diagnosis is not the usual one in the routine work of a pathologist, where, in most cases, the sentinel axillary lymph nodes (SLN) do not present metastases. Therefore, the comparison of the proposed models with the clinical reality of the expert pathologists is not totally fair. In addition, in the usual exercise of their profession, pathologists have more factors when defining their diagnosis and are not based exclusively on SLN sections. This, in short, makes comparing the performance of algorithms and pathologists more unfair.

On the other hand, the models have been specifically trained to exclusively discriminate between metastatic and non-metastatic tissue. Therefore, they do not have the ability to diagnose other diseases that could be present in the lymph nodes, such as lymphomas, sarcomas, or infection. The detection of these diseases is relevant in the routine diagnosis of pathologists and has not been incorporated in the present work. Therefore, again, the conditions in which this work has been carried out do not fully emulate clinical reality.

Furthermore, the images used are a simplification of the original database of the Camelyon16 Challenge. The original database was made up of WSI of higher resolution and, therefore, with more information. The current one, on the other hand, is made up of patches resulting from sampling the WSI, saved in lower resolution formats, such as .tif. Additionally, the test set was not available, so the number of total images was reduced compared to the original dataset.

Finally, the use of Google Colab VMs has meant a limitation at the level of computational resources for creating more sophisticated networks from scratch. Model from scratch #3 already had a very long training time, which increased exponentially with the addition of more ResNet-Inception blocks. One of the problems of using Google Colab is that the sessions cannot run for more than 24 hours and, therefore, it was not possible to carry out experiments with other models that could possibly have outperformed those exposed in this document.

Chapter 6

Conclusions and future work

6.1 Conclusions

Finally, this section presents the final conclusions of the work. These are directly identified with the objectives established in Section 1.4. The conclusions are as follows:

- The diagnosis of the stage of breast cancer based on lymphatic spread is essential when defining a prognosis and a treatment. Here the detection of metastasis in sentinel axillary lymph nodes (SLN) acquires special relevance. Therefore, its correct and early diagnosis is essential to combat one of the main causes of death in women.
- It has been possible to design models based on Convolutional Neural Networks capable of detecting metastatic tissue from sections of SLN stained with H&E. Therefore, the applicability of deep learning and, more specifically, of CNNs in extremely sensitive fields such as medical imaging is evident. The proposed networks present good enough performances for their integration and use in hospitals, for example, helping in the treatment of patients giving an early diagnosis, so that the pathologist starts from a base when defining the final diagnosis. Another implementation could be the use of these algorithms after the pathologist's diagnosis, so that it can be contrasted and reviewed in the case of issuing different verdicts.
- It has been shown that it is possible to create models from scratch capable of competing with some of the most popular pre-trained networks. Furthermore, the increase in the sophistication of the network has gone hand in hand with a better classification capacity and, ultimately, greater utility and applicability in hospitals. The use of regularizing techniques such as dropout, batch normalization and data augmentation has served to prevent overfitting, which is imposed as one of the biggest problems present in CNN training. Finally, the integration of the residual and inception modules has served to create a unique and particular architecture for the present problem, with quite good results in the detection of metastatic tissue in axillary lymph nodes.

- The great usefulness of fine-tuning in virtually any classification problem and, more specifically, medical imaging has been exposed. The use of this technique in pre-trained architectures – such as VGG19, ResNet50 or MobileNet – provides the designer with a very powerful tool to achieve good results with relatively fast and stable training, since it benefits from networks with previously optimized weights. The results of the pre-trained networks have been satisfactory regardless of the inherited base-model, the one based on VGG19 and MobileNet having an especially good performance. Specifically, MobileNet is characterized by a smaller number of parameters and, therefore, with less memory occupation, a criterion to take into account in certain environments. In short, all these networks could have an application in hospitals, for example, by reviewing the initial diagnosis of the pathologist, as discussed above.
- A comparison between CNNs and other traditional machine learning techniques has been developed. Although it has not been totally fair, it has been possible to identify some CNN requirements that other techniques do not need, such as huge databases and high computational costs, which require GPU-acceleration. In any case, it is concluded that it is important to be familiar with both spheres, the most sophisticated and the most traditional, and that the applicability of each one will depend on the specific problem and the available resources.
- It has been possible to develop a tool based on Grad-CAM to facilitate the understanding of the applicability of deep learning and CNNs in medical imaging. The use of Grad-CAM – exemplified in the network based on VGG19 – allows the pathologist to visualize the patterns that the network identifies to detect cancer, so that the expert can validate its correct operation. Furthermore, this specific diagnosis, based on the extraction and analysis of the sentinel axillary lymph node, stood out for being especially complex, leading experts to error on many occasions. Therefore, the proposed technique also makes it possible to identify previously unknown patterns that can be used for later classifications or to start new investigations in the field.
- Furthermore, this project has a dimension and utility linked to sustainable development, that is, it has an environmental, social, and economic component. The author supports the integration of machine learning techniques into any field, especially the medical field. It has been demonstrated that democratizing these technologies has a clear social value, since they help to deal with extremely sensitive issues such as breast cancer diagnosis, serving as support for the pathologist and in no case serving as a substitute for it. In addition, computational cost has been studied as a variable to take into account due to the carbon footprint. Using simpler models like k-Nearest Neighbors and Random Forest instead of CNNs has been shown to help minimize this impact.

6.2 Future work

Finally, in this section some possible future work routes are indicated, to improve the system proposed in this document and potentially obtain better results.

In the present work, several deep learning solutions have been investigated, however, all of them based on Convolutional Neural Networks. It would therefore be interesting to compare the results obtained with those of other deep learning techniques with state-of-the-art performance in medical imaging, such as Generative Adversarial Networks (GANs) or Deep Boltzmann Machines (DBMs). Some successful examples of these techniques are those presented by Suk on the use of DBMs in the detection of Alzheimer's disease [87] or those presented by Yi et al. in the literature review [99] on the usefulness of GANs in medical imaging.

Furthermore, one could experiment with the combination of the CNNs presented in this document and the traditional algorithms of [1]. One of the main difficulties of traditional machine learning is the extraction of representative features of the images. Therefore, models could be designed whose features extraction was carried out with CNNs and the classification with algorithms such as k-Nearest Neighbors, Support Vector Machines or Random Forest, instead of with a multilayer perceptron.

Another problem that could be addressed is the shortage of labelled images. It is a prevalent problem in medical imaging, to which is added the privacy of patients and the use of sensitive information, which often leads to reluctance to share their personal data. In addition, it is especially complex in diagnosis such as the one in the present work, where the work of the pathologist in classifying is of fundamental importance. Therefore, unsupervised, semi-supervised or weakly supervised learning models could be proposed, which do not depend entirely on this labelling.

It would also be interesting to refine the proposed model, for example, by adding filters that highlight the colors of the lymph node sections and thus make it easier to identify the patterns. For example, it is proposed to improve the brightness and saturation of the images, as this could potentially have an impact on better classification capacity and, ultimately, higher accuracy scores.

Finally, some of the limitations raised in Chapter 5.3 could be overcome. The current database is a simplification of the original (Camelyon16 Challenge), therefore, to make sure of the classification capacity of the proposed models, they could be tested with these WSIs that, in addition, will have a better resolution and will give more information for the classification. Obviously, this change would involve incorporating a part of image and database treatment into the system, since the handling of WSIs is more complex than that of images in .tif format.

In the case of implementing all of the above, it would result in a more real system and potentially with greater classification capacity, in short, a more reliable application in hospitals.

Appendices

Appendix A

The gradient descent algorithm

A.1 Mathematical background

The gradient descent algorithm is one of the most popular and used in machine learning, fundamentally serving as the basis of the Neural Networks learning mechanism. The principle of the descent by gradient is the descent by the slope of a surface that represents the loss function of the system. In this way, with each epoch the losses will be minimized by optimizing the weights accordingly, until the absolute minimum of the function is located and converged, or at least a local minimum that leads to sufficiently good results. This concept is illustrated in Figure A.1.

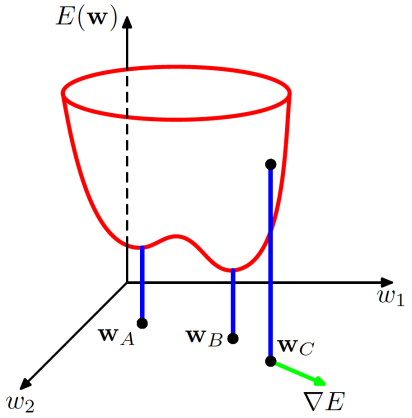


Figure A.1: Weights update is done by gradient descent, taking steps in the opposite direction to the gradient of the loss function. The optimization process is similar to the exploration of a surface, looking for its minimum. From *Pattern recognition and machine learning* [7], by C. M. Bishop, 2006

Where w_i are the weights, in this case, of the two layers of neurons that make up this network. $E(w)$ is understood as the loss function to be minimized. Furthermore, the non-linear relationships between E and the neural network weights w_i mean that the error surface is not perfectly convex. In fact, there will be several local minimums, so it will not always be necessary, as

previously mentioned, to find the absolute or global minimum, but there may be local minimums with sufficiently good results. In general, all parameters of the model, including the weights w_i, are noted as θ.

Due to the absence of an analytical solution, the gradient descent opts for an iterative process that achieves a value of the parameters θ such that $\nabla_\theta E = 0$. The most elementary expression of the descent by gradient is shown in equations A.1 and A.2, of updating the parameters and obtaining $\Delta\theta_t$ respectively. η is the learning rate.

$$\theta_{t+1} = \theta_t + \Delta\theta \tag{A.1}$$

$$\Delta\theta = -\eta\nabla_\theta E(\theta) \tag{A.2}$$

The default approach to gradient descent is the batch gradient descent, where the gradient of the loss function is computed for the entire training set. However, since gradients are calculated for a single update, the algorithm is considerably slow, and some datasets are excessively large and do not fit in memory. Faced with this problem, the most effective and most popular approach is **stochastic gradient descent** (SGD). The parameters are updated for each training sample. In each iteration, $\nabla_\theta E$ is calculated for this sample and the parameters θ are updated dynamically. In addition, another version of SGD is the one based on mini-batches, in which this same system is followed, but updating the parameters for each mini-batch, that is, each data set into which the training set is divided. Its main advantage is being less computationally intensive, converging faster and usually reaching minimums with better results than the original algorithm.

However, the traditional SGD still has some drawbacks. Some of them are the difficulty of choosing an adequate learning rate, which avoids excessively slow convergences or that diverges so that an optimal minimum is not found, or the risk of being trapped in a local minimum with unsatisfactory results and that does not allow to sufficiently minimize the loss function. Considering these challenges, the following section presents the optimizers used in this work.

A.2 Gradient descent optimization algorithms

Here are some techniques and approaches to face the challenges aforementioned.

SGD extensions: step decay. Learning is optimized so that at the beginning of training the learning rate has higher values, and it decreases as it converges. Some proposals are the exponential decay and the $1/t$ decay.

SGD extensions: Momentum [68]. SGD has difficulties in areas where one dimension is more steeply than the other, which are usually close to local optima. Momentum accelerates the SGD in the convenient direction, avoiding unnecessary oscillations that slow down convergence. Similarly, it helps to escape from unsatisfactory local minimums. γ is the momentum rate, usually set to 0.9 or similar.

$$v_t = \gamma v_{t-1} + \eta\nabla_\theta E(\theta) \tag{A.3}$$

$$\theta_{t+1} = \theta_t - v_t \tag{A.4}$$

SGD extensions: Nesterov accelerated gradient [61]. Momentum will help to converge the algorithm faster, therefore, it is convenient to make it more intelligent. Nesterov accelerated gradient incorporates information to the algorithm about the future position, so it includes a corrective factor to avoid converging excessively quickly to erroneous local minimums.

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta E(\theta - \gamma v_{t-1}) \tag{A.5}$$

$$\theta_{t+1} = \theta_t - v_t \tag{A.6}$$

Adagrad [21]. Gradient descent algorithm that adapts the learning rate depending on the parameter. For example, it will use lower learning rates for associated parameters with frequent features, while it will use high learning rates with infrequent features. G_t is a diagonal matrix where each element of the diagonal i, i is the sum of the squares of the gradients with respect to the parameter θ_i until the step t, and g_t is the gradient at time step t. On the other hand, ϵ is a term that prevents from division by zero, usually with a value around 10^{-8}. Therefore, it eliminates the need to manually modify the learning rate.

$$g_t = \nabla_\theta E(\theta_t) \tag{A.7}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t \tag{A.8}$$

RMSProp [27]. Modifies Adagrad to prevent aggressive reduction of the effective learning rate. It includes a term of decay, traditionally around 0.9.

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2 \tag{A.9}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{A.10}$$

Adam (Adaptive Moment Estimation) [44]. Method that computes adaptive learning rates for each parameter – using the term v_t – in addition to incorporating a m_t function similar to momentum. In short, it is a softened version of RMSProp. β_1 and β_2 are the decay rates, the author proposes the default values of 0.9 and 0.999, respectively.

$$\hat{m_t} = \frac{m_t}{1 - \beta_1^t} = \frac{\beta_1 m_{t-1} + (1 - \beta_1) g_t}{1 - \beta_1^t} \tag{A.11}$$

$$\hat{v_t} = \frac{v_t}{1 - \beta_2^t} = \frac{\beta_2 v_{t-1} + (1 - \beta_2) g_t^2}{1 - \beta_2^t} \tag{A.12}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{A.13}$$

Adamax [44]. Generalizes the update expression v_t in the Adam method. Specifically, it is shown that using a normalization l_∞ the convergence is more stable.

$$u_t = max(\beta_2 \cdot v_{t-1}, |g_t|) \tag{A.14}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{u_t}\hat{m}_t \tag{A.15}$$

Nadam [20]. Adam version incorporating the Nesterov momentum.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\hat{v}_t + \epsilon}(\beta_1 \hat{m}_t + \frac{(1-\beta_1)g_t}{1-\beta_1^t}) \tag{A.16}$$

Appendix B

The back-propagation mechanism

B.1 Mathematical background

The back-propagation mechanism [73] is the one that allows the learning of a neural network. It is important to remember that each neuron is characterized by weights w, of which one is denoted as b, being the independent term, and an activation function that must be easily differentiable. The back-propagation mechanism minimizes the loss function and updates the value of the weights, therefore, it has a direct relationship with the optimizer that we choose, for example, SGR, RMSProp or Adam.

The objective of the back-propagation is to compute the partial derivatives $\frac{\partial C}{\partial w}$ and $\frac{\partial C}{\partial b}$ of the loss function that models the system with respect to the weights w and the independent term b. The most common loss functions are presented in the next section, however, it is relevant to mention what conditions must be met. The first assumption is that it can be expressed by averaging various loss functions. In other words, it can take the form of a summation of type $C = \frac{1}{n}\sum C_x$. The second assumption is that it can be written as a function of the network outputs. For example, in the case of the quadratic function (B.1) it can be written so that the system outputs are included (B.2).

$$C = \frac{1}{N}\sum_{x} ||y(x) - a(x)^L||^2 \tag{B.1}$$

$$C = \frac{1}{N}\sum_{j} (y_j - a_j^L)^2 \tag{B.2}$$

In short, the indicated partial derivatives must be computed. For this, an intermediate function δ_{lj} is introduced that symbolizes the error of neuron j in layer l. Depending on the value of this error, the cost function may be reduced a little or a lot. The expression obeys equation B.3. The back-propagation mechanism will serve to compute the value of δ_{lj} to later relate it to $\frac{\partial C}{\partial w}$ and $\frac{\partial C}{\partial b}$.

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} \tag{B.3}$$

Once this concept has been introduced, the four equations that determine the behavior of back-propagation are now summarized. Its assimilation is complex and its instantaneous understanding is not expected. In fact, for this reason its use is reduced to a black-box on which the system is built.

The first equation B.4 is the error of the output layer δ_L. The partial derivative measures the rate of change of the cost function as a function of the activation of the output of layer j. The second term measures how fast activation changes z_j^L. Its usual formulation is by means of a matrix expression B.5.

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L) \tag{B.4}$$

$$\delta^L = \Delta_a C \odot \sigma'(z^L) \tag{B.5}$$

The second equation B.6 is for the error δ_l in terms of the error in the next layer,δ_{l+1}. The product with the transposition of the weight matrix is understood as the movement "backwards" in the network.

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{B.6}$$

The third equation B.7 explains the rate of change of the loss function with respect to any bias in the network.

$$\delta_j^l = \frac{\partial C}{\partial b_j^l} \tag{B.7}$$

Finally, the fourth equation B.8 explains the rate of change of the loss function with respect to any weight in the network.

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l \tag{B.8}$$

B.2 Popular loss functions

Finally, once broadly defined how the back-propagation mechanism works, some of the most important loss functions are introduced here. The choice of the loss function has a fundamental relevance in the design of the model, since it is what would lead the optimizer to find the absolute minimum or a sufficiently good local minimum [69].

In regression problems, the most common is to use Mean Squared Error Loss (B.9). It is the desired one when the target variable follows a Gaussian distribution. Its computation is simple and deals with the differences between the predicted and the real values.

$$L = \frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)^2 \tag{B.9}$$

On the other hand, in classification problems, the use of cross-entropy is recommended (B.10). It computes the average of the differences between the probabilities of the classes predicted for an object ($\hat{y}$) and to the ones which it really belongs (y). In problems with a binary nature, this function is called a binary cross-entropy. In the case of being a multiclass problem, it is called categorical cross-entropy. Also, in problems where classes are mutually exclusive, tags tend to be encoded using one-hot encoding. In this case, the loss function is called sparse categorical cross-entropy. A general version can be found in equation B.11. Here, y_{ij} is a binary indicator that defines if the object has been correctly classified, and p_{ij} is the probability of assigning the class j to the object i. In general, there are M objects and N possible classes.

$$L = -\sum_{i=1}^{N} y_i log(\hat{y}_i) \tag{B.10}$$

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \log\, p_{ij} \tag{B.11}$$

Bibliography

[1] J. Abad. "A comparison of machine learning models for the detection of metastatic tissue in axillary lymph nodes ". B.S. Thesis. Universitat Politècnica de València (UPV), 2020 (cit. on pp. 2, 4, 54, 55, 61).

[2] V. Antun, F. Renna, C. Poon, B. Adcock, and A. Hansen. "On instabilities of deep learning in image reconstruction and the potential costs of AI". In: *Proceedings of the National Academy of Sciences* (May 2020). DOI: 10.1073/pnas.1907377117 (cit. on p. 7).

[3] P. Bandi, O. Geessink, Q. Manson, M. Van Dijk, M. Balkenhol, M. Hermsen, B. E. Bejnordi, B. Lee, K. Paeng, A. Zhong, et al. "From detection of individual metastases to classification of lymph node status at the patient level: the camelyon17 challenge". In: *IEEE transactions on medical imaging* 38.2 (2018), pp. 550–560 (cit. on p. 6).

[4] R. Battiti. "First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method". In: *Neural Computation* 4 (Mar. 1992). DOI: 10.1162/neco.1992.4.2.141 (cit. on p. 14).

[5] B. E. Bejnordi, M. Veta, P. J. Van Diest, B. Van Ginneken, N. Karssemeijer, G. Litjens, J. A. Van Der Laak, M. Hermsen, Q. F. Manson, M. Balkenhol, et al. "Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer". In: *Journal of the American Medical Association* 318.22 (2017), pp. 2199–2210 (cit. on p. 6).

[6] Y. Bengio. *Learning deep architectures for AI*. Now Publishers Inc, 2009 (cit. on p. 6).

[7] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006 (cit. on p. 65).

[8] L. Breiman. "Random Forests". In: *Machine Learning* 45.1 (Oct. 2001), 5–32. ISSN: 0885-6125. DOI: 10.1023/A:1010933404324 (cit. on p. 55).

[9] D. Broomhead and D. Lowe. "Radial basis functions, multi-variable functional interpolation and adaptive networks". In: *Royal Signals and Radar Establishment Malvern (United Kingdom)* RSRE-MEMO-4148 (Mar. 1988) (cit. on p. 12).

[10] Cancer.Net. *Breast Cancer Guide: Introduction*. Tech. rep. American Society of Clinical Oncology (ASCO), July 2019 (cit. on pp. 7, 8).

[11] G. A. Carpenter and S. Grossberg. "Adaptive Resonance Theory". In: *Encyclopedia of Machine Learning*. 2010 (cit. on p. 12).

[12] The Cancer Statistics Center. *How Common Is Breast Cancer?* Tech. rep. The American Cancer Society, Sept. 2019 (cit. on p. 7).

[13] J. Cheng, D. Ni, Y. Chou, J. Qin, C. Tiu, Y. Chang, C. Huang, D. Shen, and C. Chen. "Computer-Aided Diagnosis with Deep Learning Architecture: Applications to Breast Lesions in US Images and Pulmonary Nodules in CT Scans". In: *Scientific Reports* 6 (Apr. 2016), p. 24454. DOI: 10.1038/srep24454 (cit. on p. 6).

[14] F. Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1800–1807 (cit. on p. 27).

[15] D. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber. "Flexible, High Performance Convolutional Neural Networks for Image Classification." In: July 2011, pp. 1237–1242. DOI: 10.5591/978-1-57735-516-8/IJCAI11-210 (cit. on p. 18).

[16] D. Ciresan, U. Meier, and J. Schmidhuber. "Multi-column deep neural networks for image classification". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 3642–3649 (cit. on p. 18).

[17] T. M. Cover and P. E. Hart. "Nearest neighbor pattern classification". In: *IEEE Transactions on Information Theory* 13 (1967), pp. 21–27 (cit. on p. 55).

[18] G. Currie, E. Hawk, E. Rohren, A. Vial, and R. Klein. "Machine learning and deep learning in medical imaging: Intelligent imaging". English. In: *Journal of Medical Imaging and Radiation Sciences* 50.4 (Dec. 2019), pp. 477–487. ISSN: 1876-7982. DOI: 10.1016/j.jmir.2019.09.005 (cit. on p. 6).

[19] Q. Dou, H. Chen, L. Yu, L. Zhao, J. Qin, D. Wang, V. C. Mok, L. Shi, and P. Heng. "Automatic Detection of Cerebral Microbleeds From MR Images via 3D Convolutional Neural Networks". In: *IEEE Transactions on Medical Imaging* 35.5 (2016), pp. 1182–1195 (cit. on p. 6).

[20] T. Dozat. "Incorporating Nesterov Momentum into Adam". In: 2016 (cit. on p. 68).

[21] J. Duchi, E. Hazan, and Y. Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12 (July 2011), pp. 2121–2159 (cit. on p. 67).

[22] B. Ehteshami Bejnordi et al. "Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Cancer". In: *Journal of the American Medical Association* 318 (Dec. 2017), pp. 2199–2210. DOI: 10.1001/jama.2017.14585 (cit. on p. 30).

[23] A. Esteva, B. Kuprel, R. Novoa, J. Ko, S. Swetter, H. Blau, and S. Thrun. "Dermatologist-level classification of skin cancer with deep neural networks". In: *Nature* 542 (Jan. 2017). DOI: 10.1038/nature21056 (cit. on p. 17).

[24] A. Fakhry, H. Peng, and S. Ji. "Deep models for brain EM image segmentation: novel insights and improved performance". In: *Bioinformatics* 32 (Mar. 2016), p. 165. DOI: 10.1093/bioinformatics/btw165 (cit. on p. 6).

[25] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Journal of Machine Learning Research - Proceedings Track* 9 (Jan. 2010), pp. 249–256 (cit. on p. 16).

[26] X. Glorot, A. Bordes, and Y. Bengio. "Deep Sparse Rectifier Neural Networks". In: vol. 15. Jan. 2010 (cit. on p. 12).

[27] A. Graves. "Generating Sequences With Recurrent Neural Networks". In: (Aug. 2013) (cit. on p. 67).

[28] J. Griffin and D. Treanor. "Digital pathology in clinical use: Where are we now and what is holding us back?" In: *Histopathology* 70 (Jan. 2017), pp. 134–145. DOI: 10.1111/his.12993 (cit. on p. 9).

[29] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778 (cit. on pp. 21, 26).

[30] K. He, X. Zhang, S. Ren, and J. Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *IEEE International Conference on Computer Vision (ICCV 2015)* 1502 (Feb. 2015). DOI: 10.1109/ICCV.2015.123 (cit. on pp. 14, 22).

[31] K. He, X. Zhang, S. Ren, and J. Sun. "Identity Mappings in Deep Residual Networks". In: vol. 9908. Oct. 2016, pp. 630–645. ISBN: 978-3-319-46492-3. DOI: 10.1007/978-3-319-46493-0_38 (cit. on pp. 26, 27).

[32] G.E. Hinton and R.R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science (New York, N.Y.)* 313 (Aug. 2006), pp. 504–7. DOI: 10.1126/science.1127647 (cit. on p. 6).

[33] J. Hopfield. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities". In: *Proceedings of the National Academy of Sciences of the United States of America* 79 (May 1982), pp. 2554–8. DOI: 10.1073/pnas.79.8.2554 (cit. on p. 12).

[34] K. Hornik, M. B. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2 (1989), pp. 359–366 (cit. on pp. 12, 21).

[35] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: (Apr. 2017) (cit. on p. 28).

[36] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. "Densely Connected Convolutional Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2261–2269 (cit. on pp. 23, 27).

[37] R. Iglesis, R. Sandoval, R. Schwartz, and R. Olguin. "Hispatological analysis of the sentinel lymph node. Techniques and results". In: *Revista Chilena de Cirugía* 54 (2002), pp. 380–3 (cit. on p. 8).

[38] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: (Feb. 2015) (cit. on pp. 6, 16).

[39] A. K. Jain, J. Mao, and K. M. Mohiuddin. "Artificial neural networks: A tutorial". In: *Computer* 29.3 (1996), pp. 31–44 (cit. on p. 12).

[40] A. K. Jain and A. Vailaya. "Image retrieval using color and shape". In: *Pattern Recognition* 29 (1996), pp. 1233–1244 (cit. on p. 55).

[41] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*. Vol. 112. Springer, 2013 (cit. on p. 17).

[42] N. Japkowicz. "Why Question Machine Learning Evaluation Methods ? (An illustrative review of the shortcomings of current methods)". In: 2006 (cit. on p. 33).

[43] J. Jotheeswaran and S. Tanwar. "Survey on Deep Learning for Medical Imaging". In: 5 (Dec. 2018), pp. 1608–20 (cit. on p. 17).

[44] D. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (Dec. 2014) (cit. on pp. 67, 68).

[45] P. W. Koh and P. Liang. "Understanding black-box predictions via influence functions". In: *arXiv preprint arXiv:1703.04730* (2017) (cit. on p. 7).

[46] J. F. Kolen and S. C. Kremer. "Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies". In: *A Field Guide to Dynamical Recurrent Networks*. 2001, pp. 237–243 (cit. on p. 21).

[47] D. Krag et al. "Technical outcomes of sentinel-lymph-node resection and conventional axillary-lymph-node dissection in patients with clinically node-negative breast cancer: results from the NSABP B-32 randomised phase III trial". In: *The lancet oncology* 8 (Nov. 2007), pp. 881–8. DOI: 10.1016/S1470-2045(07)70278-4 (cit. on p. 8).

[48] A. Krizhevsky, I. Sutskever, and G. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Neural Information Processing Systems* 25 (Jan. 2012). DOI: 10.1145/3065386 (cit. on p. 26).

[49] M. Kuhn and K. Johnson. *Applied predictive modeling*. Vol. 26. Springer, 2013 (cit. on p. 17).

[50] S. Kumar. "On weight initialization in deep neural networks". In: (Apr. 2017) (cit. on p. 14).

[51] S. Lawrence, C. L. Giles, A.Chung Tsoi, and A. D. Back. "Face recognition: a convolutional neural-network approach". In: *IEEE Transactions on Neural Networks* 8.1 (1997), pp. 98–113 (cit. on p. 18).

[52] Y. LeCun, Y. Bengio, and G. Hinton. "Deep Learning". In: *Nature* 521 (May 2015), pp. 436–44. DOI: 10.1038/nature14539 (cit. on p. 6).

[53] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on pp. 1, 18).

[54] Y. LeCun, L. Bottou, G. Orr, and K. R. Müller. "Efficient BackProp". In: (Aug. 2000) (cit. on p. 35).

[55] M. Lin, Q. Chen, and S. Yan. "Network In Network". In: (Dec. 2013) (cit. on p. 37).

[56] G. Litjens, P. Bandi, B. E. Bejnordi, O. Geessink, M. Balkenhol, P. Bult, A. Halilovic, M. Hermsen, R. van de Loo, R. Vogels, et al. "1399 H&E-stained sentinel lymph node sections of breast cancer patients: the CAMELYON dataset". In: *GigaScience* 7.6 (2018), giy065 (cit. on p. 6).

[57] C. von der Malsburg. "Frank Rosenblatt: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms". In: *Brain Theory* (Jan. 1986), pp. 245–248. DOI: 10.1007/978-3-642-70911-1_20 (cit. on p. 11).

[58] W. S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133 (cit. on p. 11).

[59] B. Munsell, C. Wee, S. Keller, B. Weber, C. Elger, L. Silva, T. Nesland, M. Styner, D. Shen, and L. Bonilha. "Evaluation of machine learning algorithms for treatment outcome prediction in patients with epilepsy based on structural connectome data". In: *NeuroImage* 118 (June 2015). DOI: 10.1016/j.neuroimage.2015.06.008 (cit. on p. 6).

[60] V. Nair and G. E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *ICML*. 2010 (cit. on pp. 6, 16).

[61] Y. Nesterov. "A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$". In: 1983 (cit. on p. 67).

[62] D. Nie, L. Wang, Y. Gao, and D. Sken. "Fully convolutional networks for multi-modality isointense infant brain image segmentation". In: vol. 2016. Apr. 2016, pp. 1342–1345. DOI: 10.1109/ISBI.2016.7493515 (cit. on p. 6).

[63] F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P. E. Barbano. "Toward automatic phenotyping of developing embryos from videos". In: *IEEE Transactions on Image Processing* 14.9 (2005), pp. 1360–1371 (cit. on p. 18).

[64] L. Pantanowitz. "Digital images and the future of digital pathology". In: *Journal of pathology informatics* 1 (2010) (cit. on p. 6).

[65] N. Pathan and M. E. Jadhav. "Medical Image Classification Based on Machine Learning Techniques". In: *Advanced Informatics for Computing Research*. Ed. by A. K. Luhach, D. S. Jat, K. Bin G. Hawari, X. Gao, and P. Lingras. Singapore: Springer Singapore, 2019, pp. 91–101. ISBN: 978-981-15-0108-1 (cit. on p. 7).

[66] L. Y. Pratt. "Discriminability-Based Transfer between Neural Networks". In: *Advances in Neural Information Processing Systems 5, [NIPS Conference]*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1992, 204–211. ISBN: 1558602747 (cit. on p. 25).

[67] L. Y. Pratt and S. Thrun. "Machine Learning-Special Issue on Inductive Transfer". In: (1997) (cit. on p. 25).

[68] N. Qian. "On the momentum term in gradient descent learning algorithms". In: *Neural networks* 12.1 (1999), pp. 145–151 (cit. on p. 66).

[69] L. Rosasco, E. De Vito, A. Caponnetto, M. Piana, and A. Verri. "Are Loss Functions All the Same?" In: *Neural computation* 16 (June 2004), pp. 1063–76. DOI: 10.1162/089976604773135104 (cit. on p. 70).

[70] F. Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65 6 (1958), pp. 386–408 (cit. on p. 1).

[71] H. R. Roth, L. Lu, J. Liu, J. Yao, A. Seff, K. Cherry, L. Kim, and R. M. Summers. “Improving Computer-Aided Detection Using Convolutional Neural Networks and Random View Aggregation”. In: *IEEE Transactions on Medical Imaging* 35.5 (2016), pp. 1170–1181 (cit. on p. 6).

[72] P. Roy, S. Ghosh, S. Bhattacharya, and U. Pal. “Effects of Degradations on Deep Neural Network Architectures”. In: *ArXiv* abs/1807.10108 (2018) (cit. on p. 21).

[73] D. E. Rumelhart and J. L. McClelland. “Learning Internal Representations by Error Propagation”. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. 1987, pp. 318–362 (cit. on pp. 14, 69).

[74] O. Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision* 115 (Sept. 2014). DOI: 10.1007/s11263-015-0816-y (cit. on p. 25).

[75] S. Russell and P. Norvig. “Artificial intelligence: a modern approach”. In: (2002) (cit. on p. 17).

[76] R. Salakhutdinov. “Learning Deep Generative Models”. In: *Annual Review of Statistics and Its Application* 2.1 (2015), pp. 361–385. DOI: 10.1146/annurev-statistics-010814-020120 (cit. on p. 6).

[77] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K. Müller. “Evaluating the visualization of what a deep neural network has learned”. In: *IEEE transactions on neural networks and learning systems* 28.11 (2016), pp. 2660–2673 (cit. on p. 7).

[78] R. J Schalkoff. “Pattern recognition”. In: *Wiley Encyclopedia of Computer Science and Engineering* (2007) (cit. on p. 12).

[79] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 618–626 (cit. on p. 49).

[80] A. A. A. Setio, F. Ciompi, G. Litjens, P. Gerke, C. Jacobs, S. J. van Riel, M. M. W. Wille, M. Naqibullah, C. I. Sánchez, and B. van Ginneken. “Pulmonary Nodule Detection in CT Images: False Positive Reduction Using Multi-View Convolutional Networks”. In: *IEEE Transactions on Medical Imaging* 35.5 (2016), pp. 1160–1169 (cit. on p. 6).

[81] W. Shen, M. Zhou, F. Yang, C. Yang, and J. Tian. “Multi-scale Convolutional Neural Networks for Lung Nodule Classification”. In: vol. 24. July 2015, pp. 588–99. DOI: 10.1007/978-3-319-19992-4_46 (cit. on p. 6).

[82] H. Shimodaira. "Improving predictive inference under covariate shift by weighting the log-likelihood function". In: *Journal of Statistical Planning and Inference* 90 (Oct. 2000), pp. 227–244. DOI: 10.1016/S0378-3758(00)00115-4 (cit. on p. 15).

[83] H. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers. "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning". In: *IEEE transactions on medical imaging* 35.5 (2016), pp. 1285–1298 (cit. on p. 6).

[84] C. Shorten and T. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning". In: *Journal of Big Data* 6 (Dec. 2019). DOI: 10.1186/s40537-019-0197-0 (cit. on p. 16).

[85] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv 1409.1556* (Sept. 2014) (cit. on p. 26).

[86] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (June 2014), pp. 1929–1958 (cit. on pp. 6, 15).

[87] H. Suk, S. Lee, and D. Shen. "Hierarchical Feature Representation and Multimodal Fusion with Deep Learning for AD/MCI Diagnosis". In: *NeuroImage* 101 (July 2014). DOI: 10.1016/j.neuroimage.2014.06.077 (cit. on pp. 6, 61).

[88] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *AAAI Conference on Artificial Intelligence* (Feb. 2016) (cit. on p. 24).

[89] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the Inception Architecture for Computer Vision". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2818–2826 (cit. on p. 24).

[90] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9 (cit. on pp. 23, 24, 26).

[91] A. M. Turing. "I.—Computing machinery and intelligence". In: *Mind* LIX.236 (Oct. 1950), pp. 433–460. ISSN: 0026-4423 (cit. on p. 1).

[92] G. van Tulder and M. de Bruijne. "Combining Generative and Discriminative Representation Learning for Lung CT Analysis With Convolutional Restricted Boltzmann Machines". In: *IEEE Transactions on Medical Imaging* 35.5 (2016), pp. 1262–1272 (cit. on p. 6).

[93] V. Vapnik and C. Cortes. "Support-Vector Networks". In: *Machine Learning* 20.3 (Sept. 1995), 273–297. ISSN: 0885-6125. DOI: 10.1023/A:1022627411411 (cit. on p. 55).

[94] B. Veeling. *Histopathologic Cancer Detection Dataset*. https://www.kaggle.com/c/histopathologic-cancer-detection/overview. [Online; accessed May-2020]. Nov. 2018 (cit. on p. 30).

[95] B. Veeling. *The PatchCamelyon benchmark (PCam) Dataset*. https://github.com/basveeling/pcam. [Online; accessed May-2020]. June 2018 (cit. on p. 30).

[96] B. Veeling, J. Linmans, J. Winkens, T. Cohen, and M. Welling. "Rotation Equivariant CNNs for Digital Pathology". In: (June 2018). arXiv: 1806.03962 [cs.CV] (cit. on p. 30).

[97] J. Vestjens et al. "Relevant impact of central pathology review on nodal classification in individual breast cancer patients". In: *Annals of oncology : official journal of the European Society for Medical Oncology / ESMO* 23 (Apr. 2012), pp. 2561–6. DOI: 10.1093/annonc/mds072 (cit. on p. 9).

[98] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He. "Aggregated Residual Transformations for Deep Neural Networks". In: July 2017, pp. 5987–5995. DOI: 10.1109/CVPR.2017.634 (cit. on p. 23).

[99] X. Yi, E. Walia, and P. Babyn. "Generative Adversarial Network in Medical Imaging: A Review". In: *Medical Image Analysis* 58 (Aug. 2019), p. 101552. DOI: 10.1016/j.media.2019.101552 (cit. on p. 61).

[100] S. Zagoruyko and N. Komodakis. "Wide Residual Networks". In: (May 2016) (cit. on p. 37).

[101] W. Zhang, R. li, H. Deng, L. Wang, W. Lin, S. Ji, and D. Shen. "Deep Convolutional Neural Networks for Multi-Modality Isointense Infant Brain Image Segmentation". In: *NeuroImage* 108 (Jan. 2015). DOI: 10.1016/j.neuroimage.2014.12.061 (cit. on p. 6).

[102] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling. "Visualizing deep neural network decisions: Prediction difference analysis". In: *arXiv preprint arXiv:1702.04595* (2017) (cit. on p. 7).

[103] K. Zou, J. O'Malley, and L. Mauri. "Receiver-Operating Characteristic Analysis for Evaluating Diagnostic Tests and Predictive Models". In: *Circulation* 115 (Mar. 2007), pp. 654–7. DOI: 10.1161/CIRCULATIONAHA.105.594929 (cit. on p. 34).