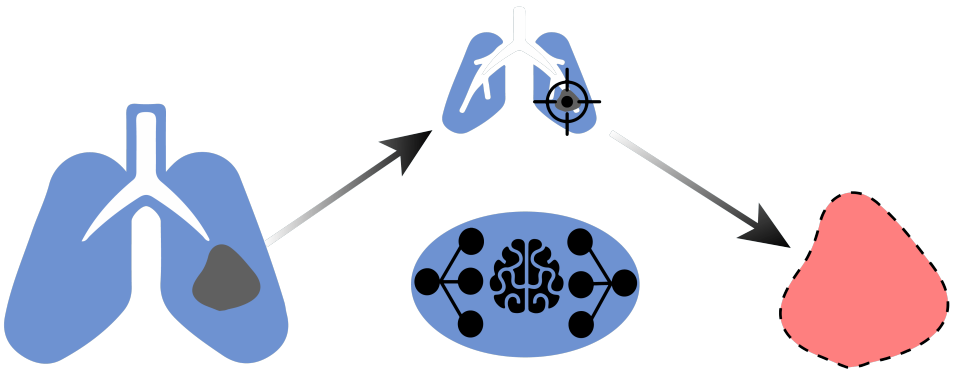
TÉCNICO LISBOA



Automatic Detection and Segmentation of Pulmonary Lesions on CT scans using Deep Convolutional Neural Networks

João Francisco Lourenço Borges de Sá Carvalho

Thesis to obtain the Master of Science Degree in

Biomedical Engineering

Supervisors:
Prof. Mário Alexandre Teles de Figueiredo
Dr. Nickolas Papanikolaou

Examination Committee

Chairperson: Prof. Patrícia Margarida Piedade Figueiredo
Supervisor: Prof. Mário Alexandre Teles de Figueiredo
Member of the Committee: Prof. Maria Margarida Campos da Silveira

November 2019

Preface

The work presented in this thesis was performed at the Computational Clinical Imaging Group at Champalimaud Centre for the Unknown (Lisbon,Portugal), during the period February-October 2019, under the supervision of Principal Investigator Nickolas Papanikolaou. The thesis was co-supervised at Instituto Superior Técnico by Prof. Mário Figueiredo.

Declaration

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

Acknowledgments

During these past five years, and in specific, for the most part of 2019, more people than I could name have helped, inspired, and impacted me. I will nonetheless make an effort to acknowledge everyone that made this stage of my life a bit more special.

I must start by thanking Prof. Mário Figueiredo. Without his experience, guidance, and unrelenting dedication, this project wouldn't have reached its final success. More than a supervisor, he was a teacher, and allowed me to leave this experience with much more than I started with. I would also like to thank Nickolas Papanikolaou for providing me with the opportunity to work in this field, as well as providing keen insight during the critical stages of my project. I must acknowledge and thank the countless hours José Moreira and João Santinha have spent debugging and discussing the challenges that I faced.

Throughout my degree, I have met some of the most dedicated and passionate researchers and professors that I could imagine. All of them contributed to my growth. But from these, I must express my gratitude to Prof. Ana Fred, who provided me with my first contact with research, and shaped my critical thinking from those early stages.

Then, to all my friends and colleagues... Rodrigo, Cristiano, Afonso, Cátia, Gouveia, Valentina, Joana, Alex, Mariana, and many others. I was fortunate to have met you and must thank you for giving me so many moments to cherish.

To my family for their unlimited support and care, over all these years. Above all, to my parents, whose value to me only grows with age, thank you.

Finally, to Leonor: thank you for everything. For your patient, dedication, and unconditional love.

Abstract

Lung cancer is the most pervasive and one of the deadliest types of cancer, leading to more than 2 million cases being diagnosed each year, and a mortality rate of 60%. CT screening trials have played a key role in improving early detection of lung cancer, which has shown to significantly improve patient survival. Apart from lesion detection, tumour segmentation is critical for developing *radiomics* signatures. In this work, we propose a novel hybrid approach for lung lesion detection and segmentation on CT scans, where the segmentation task is assisted by prior detection of regions containing lesions. For the detection task, we introduce a 2.5D *residual deep CNN* working in a sliding-window fashion, whereas segmentation is tackled by a modified *residual U-Net* with a weighted-dice plus cross-entropy loss. Experimental results on the *LIDC-IDRI* dataset and on the lung tumour task dataset within the *Medical Segmentation Decathlon* show competitive detection performance of the proposed approach (0.902 recall) and superior segmentation capabilities (0.709 dice score). Further validation of the models was also performed, with key components of both models tested through several ablation studies, in order to assess its contribution to the final models. These results confirm the high potential of simpler models, with lower hardware requirements, thus of more general applicability.

Keywords

Radiomics, lung cancer, segmentation, deep learning, convolutional neural network, residual connections.

Resumo

O cancro do pulmão é um dos cancros que atualmente afeta mais pacientes, com mais de dois milhões de novos casos por ano, sendo ainda um dos mais mortais, atingindo um indice de mortalidade de 60%. A utilização de rastreio recorrendo a tomografia computorizada foi responsável por um elevado incremento na deteção precoce de lesões pulmonares, tendo mostrado levar a uma melhoria significativa na taxa de sobrevivência dos pacientes com esta doença. Para além da deteção de lesões, a segmentação tumoral é crítica para desenvolver *assinaturas radiómicas*. Neste trabalho, propomos uma abordagem híbrida para a deteção e segmentação de lesões pulmonares em tomografias computorizadas, onde a tarefa da segmentação é assistida pela deteção prévia de regiões que contenham lesões. Para a tarefa de deteção introduzimos uma *CNN residual profunda* 2.5D, capaz de produzir modelos menos complexos, e que é aplicada sob a forma de uma janela deslizante. A tarefa de segmentação é abordada recorrendo a uma rede *U-Net residual* modificada, cujo treino é realizado recorrendo a uma funcão de custo baseada na soma da entropia cruzada e do coeficient dice ponderado. Resultados experimentais com a base de dados *LIDC-IDRI* e na tarefa de segmentação de tumores pulmonares da competição *Decathlon da Imagem Médica* comprovam a capacidade da deteção (sensibilidade de 0.902) e de segmentação (coeficiente de dice de 0.709) da abordagem proposta. Os modelos foram ainda avalidos, sendo as suas principais caraterísticas testas através de diversos testes de ablação com o intuito de verificar a sua contribuição para os modelos finais. Estes resultados confirmam o elevado potencial de modelos mais simples, com necessidades mais baixas de hardware, e consequentemente com uma aplicação mais generalizada.

Palavras Chave

Radiomics, cancro do pulmão, segmentação, *deep learning*, redes neuronais convolucionais.

Contents

List of Figures

List of Tables

Acronyms

AUC Area Under the Curve

CAD Computer Aided Design

CNN Convolutional Neural Network

CT Computed Tomography

CV Cross-Validation

DCNN Deep Convolutional Neural Network

DL Deep Learning

FC Fully Connected

GAN Generative Adversial Network

GPU Graphics Processing Unit

HU Hounsfield Units (HU)

LIDC-IDRI Lung Image Database Consortium - Image Database Resource Initiative

MLP Multilayer Perceptron

NLST National Lung Screening Trial

NN Neural Networks

ReLU Rectified Linear Unit

ROC Receiver Operating Characteristic

ROI Region of Interest

RPN Region Proposal Network

SGD Stochastic Gradient Descent

TNR true negative rate

TPR true positive rate

USA United Sates of America

VOI Volume of Interest

WHO World Health Organization

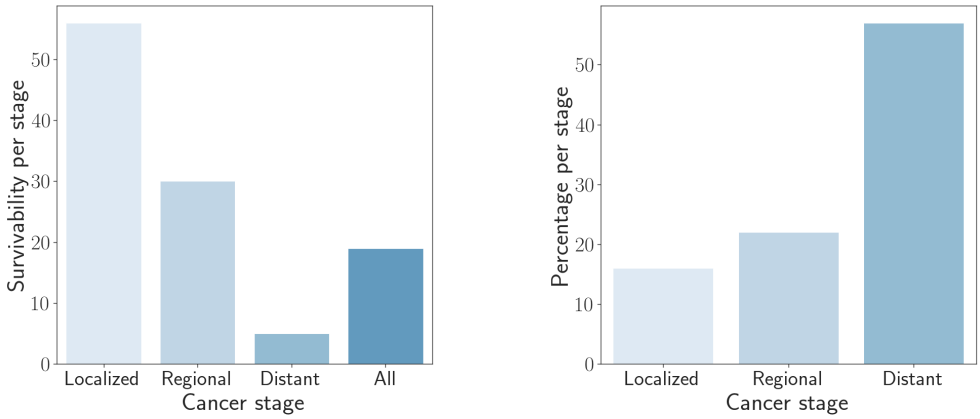
1

Introduction

Contents

1.1 Motivation and Objectives

According to the World Health Organization (WHO) [1], cancer is the second leading cause of death globally, amounting to 9.6 million deaths in 2018. Particularly, lung carcinoma is presently the most pervasive type of cancer, totalling 2.09 million cases. Just in the United States of America (USA), it was estimated that the number of new cases in 2019 would sum to more than 220 000 [88].



(a) Survivability statistics per cancer stage

(b) Distribution of cases per stage

Figure 1.1: Distribution and 5-year survivability per lung cancer stage, during 2008 through 2014. Data from [88].

The best treatment strategies to fight lung cancer mainly rely on early detection, as there is a great increase in mortality rates following the progression of the cancer stage (Fig. 1.1(a)) [9]. However, this is not an easy task, with a large portion of lung cancer cases only being detected in its later stages, which leads to the distribution seen in Fig. 1.1(b). In this context, screening trials with thoracic Computed Tomography (CT) scan have been been shown to prevent fatalities associated with this disease. Particularly, the National Lung Screening Trial (or NLST) showed a decrease in 15-20Yet, most machine learning systems traditionally rely on hand-crafted features (a process known as “feature engineering”), which not only demands domain-specific expertise, but is also time-consuming to set up. Recently, representation models, and specifically neural networks, have lead a transformation in the field by removing this necessity, and producing state-of-the-art performance in several tasks [30, 52]. Particularly deep learning architectures, a sub-set of neural networks, have enabled the fast development of highly accurate models [78], with extraordinary results in several fields, and specifically in computer vision [78, 108]. Recent attempts to transfer this knowledge to the medical imaging field have been mostly focused in trying to use models that were originally developed for computer visions tasks, namely object detection and image recognition [7, 19, 33, 59, 105]. One disadvantage of these approaches is the large complexity of the produced models, which was adapted for the comparatively extensive data-sets in the computer

vision field, but that lead to under-performance in the context of medical imaging tasks when evaluating its ability to generalize [28]. The increased size of these models is also responsible to extreme computational and memory requirements [40, 80], which leads to the necessity of highly capable machines for both training, and posterior inference in unobserved CT images.
The main goal of this dissertation is to develop a simpler model that can perform the two most relevant tasks within lung cancer screening: these are detection and segmentation of lesions. Such model would be of great value to the medical community, due to both the significance of these tasks within radiologists' pipeline, as well as the increased usability of such models in clinical practice.

1.2 Approach and Organization

In order to achieve the goal of a simpler, non-overfitting model, a significant effort was made to avoid previously explored complex models. Through a thorough analysis of basic principles in deep learning, as well as an understanding of both the limitations of the current state-of-the-art, significant and novel components, and methodologies to avoid overfitting, this objective became achievable. A pipeline was then assembled to provide the deep learning models with the best possible pre-processed training data, and extensive testing of several approaches to detection and segmentation models was then made, with the intent of both optimizing the models' architectures, and their training. These two steps were approached with future reproducibility in mind, leading to the production of an open online python library. The final goal of the work was achieved by integrating the detection and the segmentation models into a hybrid model, and then performing considerable evaluation of both, with a comprehensive comparison with the state-of-the-art. An evaluation of the short-comings of this proposal was performed, leading to suggestions for the improvement of the approach and future work directions.

Following the described approach, this dissertation was divided into 6 chapters:

- *Chapter 1*: Motivation and objectives for the work are enunciated. Contextualization of the problem within both the clinical, and the technical setting is provided.

- *Chapter 2*: Background on the data, the clinical problem, and its connection to the recent field of *radiomics* give is given. The technical problem is formalized, and the background for Deep Learning architectures is provided, along with the state-of-the-art for the detection and segmentation task. The chapter ends with an overview of how to avoid overfitting when solving a medical imaging problem using machine learning.

- *Chapter 3*: Pre-processing, curation, and augmentation methodologies are described, together with the details of the implementation.

- *Chapter 4*: Final components of the network are introduced, and both the detection and the segmentation network architectures are defined. Several possible designs are presented, and tested. The training of both models is also described.
- *Chapter 5*: The final hybrid model, *LungSD-Net*, is presented, and all its components are evaluated and compared with the state of the art.
- *Chapter 6*: Conclusions are drawn, and future work is recommended. Final remarks on the reproducibility of the work are made.

2

Lung Cancer and Deep Learning

Contents

2.1 Lung Cancer and Radiomics

This chapter starts by establishing the context for the used data as well as the tasks at hand, through the introduction of key concepts on lung cancer. This will be connected with the emerging field of *radiomics*, where it will integrated within the *radiomics* platform.

2.1.1 Lung Anatomy and Anatomical Context

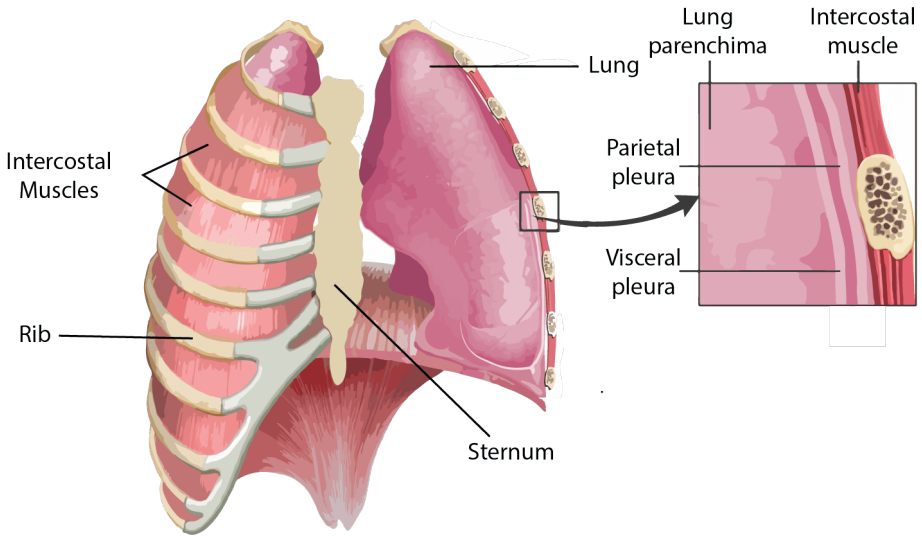


Figure 2.1: Anatomical representation of the lungs adapted from [68]

The lungs are highly complex biological structures located in the thoracic region that serves as the main organs in the respiratory system (Fig. 2.1). Each lung has three borders and two surfaces (costal and mediastinal). These are involved by the pleura, a serous membrane which is comprised of two internal layers (parietal and visceral), separated by the heart and other mediastinum contents that are placed in the central region of the thoracic cavity. The enveloping structures of the lungs are the rib cage, which is composed by the sternum and the ribs, and provides structural rigidity and protection, and the intercostal muscles, which engage in the respiratory exercise [32].

2.1.2 Types of Lung Lesions

According to their location, lung nodules can be classified as isolated, peri-fissural, juxta-vascular, or juxta-pleural [98]. Isolated nodules are well-circumscribed lesions in the central region of the lung parenchyma and, as the name indicates, are detached from any adjacent structure. On the other hand, all the other lesions are connected to other structures. Namely, peri-fissural nodules occur adjacent to lung fissures, whereas juxta-vascular nodules appear attached vascular vessels, and juxta-pleural

nodules are attached to the pleura. These last nodules tend to be more difficult to detect [98]. The evaluation of nodule texture is also relevant, with three main types arising: solid, sub-solid and non-solid lesions. Solid lesions have high contrast in comparison to the parenchyma, and well-defined margins, which leads to both easy detection and segmentation. Both sub-solid and non-solid lesions are the hardest to segment due to its highly irregular margins. Furthermore, non-solid nodules are usually diffused abnormalities, therefore the most difficult to characterize, in opposition to sub-solid nodules, which only partially obscure the lung parenchyma [8].

2.1.3 Computed Tomography

CT is a cross-sectional imaging technique that corresponds to an X-ray attenuation projection along multiple directions, ultimately yielding a complete 3D view by compiling a series of slices from the original 3D object. This is possible through the use of computed reconstruction, which attributes a linear attenuation value, μ, to an individual voxel [14, 44]. In order to standardise the attenuation value to a specific voxel, the so-called Hounsfield Units (HU) are used. These correspond to the normalization of attenuation values with respect to the water attenuation value, and are given by

$$HU = \frac{\mu - \mu_{water}}{\mu_{water}} \times 1000. \tag{2.1}$$

Ideally, water would have a value of zero HU, but due to variations in the acquisition, this is rarely the case. The attenuation coefficients in HU of different types of tissues relevant for the thoracic CT scan are shown in Figure 2.2.

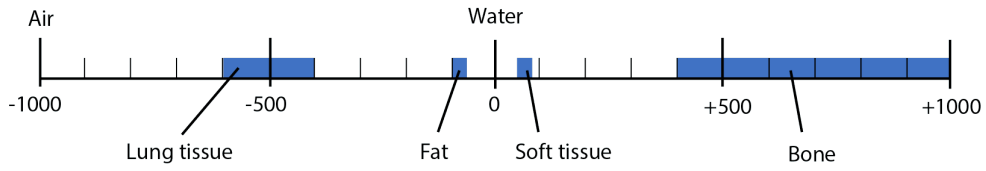


Figure 2.2: HU Values for different tissues commonly seen in a thoracic CT scan: -1000 HU correspond to black and $+1000$ HU correspond to white in a gray-scale image. The wide range of HU of the bone tissue is due to variable bone structures which may have a higher (cortical bone) or lower density (trabecular bone). (from [14])

CT scans allow the user to access internal information of the scanned objects, which has lead to extensive use of this technique in clinical practice [14]. Axial and helical CT scans have, in particular, become the most common medical examination techniques in the case of thoracic scanning [14, 50]. An example of a thoracic CT scan is displayed in Fig. 2.3, illustrating a correspondence of some of the tissues mentioned in Figure 2.2 and their representation in a reconstructed image.

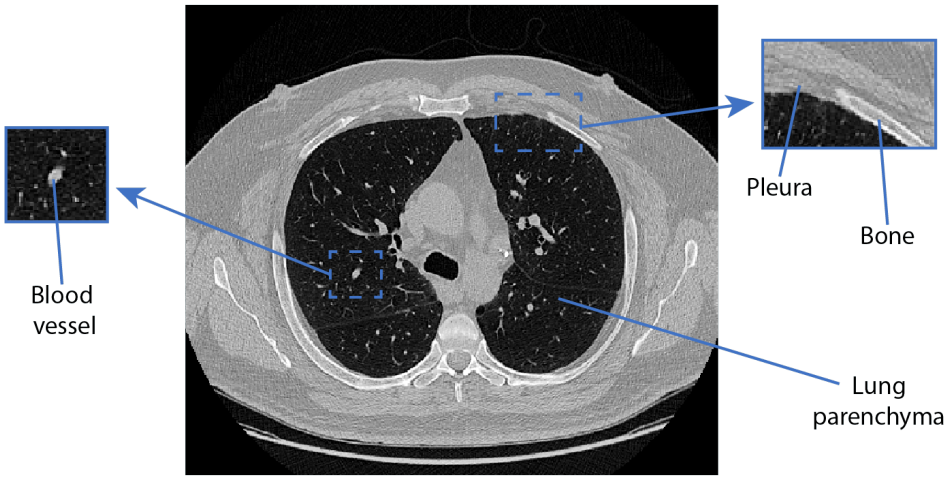


Figure 2.3: The lung parenchyma, due to its percentage of air has an extremely low value in HU. One of the major problems in lung lesion detection is high amount of vascularization in the lungs, which lead to blood vessels being confused as nodules.

In the context of lung cancer, screening trials with helical CT have been been shown to prevent fatalities associated with this disease. Particularly, the NLST showed 15-20% lower risk of death for participants who underwent low dose helical CT scans, in comparison to other methods [96]. This type of program has been extensively implemented in the USA and is currently being adopted by other countries [69].

Comparatively, non-ionizing methods, as magnetic resonance imaging, face several challenges in the context of thoracic scanning, such as the low signal to noise ratio, susceptibility to artifacts arising from multiple air-tissue interfaces, as well as motion artifacts due to cardiac, vascular and respiratory movements [102]. The lung lesion detection and segmentation system developed in this work will be designed to deal exclusively with CT scans.

2.1.4 Radiomics

The large datasets arising from different efforts to improve medical information, together with the advent of high-performance computing, have enabled the development of the field of *radiomics* [70]. It is now possible to extract high-dimensional features from medical images, the so-called *radiomic signatures*. These signatures combined with machine learning methods, such as feature selection methods, classification models, or deep learning algorithms, are used to predict cancer-related outcomes [28]. Within this framework, it is common to talk about the radiomics pipeline [64], which describes several steps between the detection of a lesion, and the final classification of its radiomic signature. The segmentation of *volumes of interest* (VOI) around lesions, and possible sub-regions (*i.e.*, habitats) within the tumour, is critical, as radiomic features are directly extracted from the VOI. Therefore, the development

of semi- or fully-automated methods for both detection and segmentation of lesions, is of great value in clinical application, to avoid the intrinsic inter-reader variability of human VOI segmentation. [66, 70]. Moreover, manual detection and segmentation of lung nodules and tumours require a large amount of tedious, expensive, and time-consuming work by human experts (radiologists) [28, 78]. Automatic tools can thus be extremely useful in daily clinical work and will play a pivotal role in increasing the quality of early cancer detection and diagnosis [70].

2.2 Deep Learning

Currently, most automatic tools used in the tasks of detection and segmentation of lung lesions rely on machine learning methods. These are methods within the field of artificial intelligence that can learn patterns from data and use them to perform predictions based on unobserved data. In the context of this work, we will deal with a sub-set of machine learning methods that seek to find a function, F, that maps the observations into desired outcomes, by learning a set of paremeters of F, θ. These are learned from a set of labeled pairs of observations and outcomes known as *training set*. Traditional machine learning methods come with several drawbacks, one of the main ones being their reliance on engineered/designed data representations [11, 30]. Representation learning algorithms, of which neural networks are a very successful example, are able to learn without any prior feature engineering. These rely on a series of non-linear transformations of the data that lead to increasingly deeper levels of abstraction, and which will be introduced in Sections 2.2.2 and 2.2.3, and further discussed in Section 2.2.4. When the number of non-linear transformations (or, as discussed below, layers) is more than three, the models are called deep [30]. Recently, what once was thought impossible, increasingly more complex models have been able to achieve outstanding results in computer vision tasks [52, 77]. These deep learning architectures have become the state-of-the-art models in tasks such as image recognition, object detection, and object segmentation, and have been extended to several fields, including medical imaging [33].

2.2.1 Problem Formulation

Before any further explanation of neural networks and deep learning, this subsection introduces some notation regarding the tasks at hand by formalizing the lung lesion detection and segmentation problems.

First, the task of finding nodules in a CT image will be defined. Consider the observation $\mathbf{X} \in \mathbb{R}^n$,where $n = H \times W \times D$, which denotes the 3D tensor of dimension $H \times W \times D$ corresponding to the CT image after being stacked into a n-dimensional vector, where each entry represents one voxel (or 3D pixel) of the data scan. The goal is to find the set of coordinates for each lesion that is present in $\mathbf{X}$, which can be denoted as $D = \{(y_1^1, y_2^1, y_3^1), ..., (y_1^N, y_2^N, y_3^N)\}$, where N denotes the number of existing nodules in $\mathbf{X}$ and $(y_1^1, y_2^1, y_3^1) \in \{1, ..., H\} \times \{1, ..., W\} \times \{1, ..., D\}$ is the vector of voxel coordinates of

the i-th lesion. Let F_1 be defined as a function that corresponds to the *detection model*, that is

$$F_1 : \mathbb{R}^n \longrightarrow (\{1, ..., H\} \times \{1, ..., W\} \times \{1, ..., D\})^*, \tag{2.2}$$

where $*$ is the Kleene star. The problem of learning a lung lesion detector is defined as that of finding a function F_1 as defined in Equation 2.2. This problem can also be restructured if a strict bounding-box around each nodule is required. In this case D would be rewritten as

$$D = \{(y_1^1, y_2^1, y_3^1, y_4^1, y_5^1, y_6^1), ..., (y_1^N, y_2^N, y_3^N, y_4^N, y_5^N, y_6^N)\}, \tag{2.3}$$

where y_{1-3} are the coordinates of the nodule's centroid, and y_{4-6} are the sizes of the bounding-box.

Similarly, for the task of segmenting a lung lesion, let $\mathbf{Z} \in \mathbb{R}^m$ be defined as a 3D cubic patch of a CT scan that contains a nodule, and $\mathbf{S}$ with the same dimensions as $\mathbf{Z}$, where each voxel $s_i \in \{1, 0\}$, depending on being part of a nodule or not. Let F_2 be defined as the function that maps $\mathbf{Z}$ to $\mathbf{S}$, from onwards called *segmentation model*:

$$F_2 : \mathbb{R}^m \longrightarrow \{0, 1\}^m. \tag{2.4}$$

Finally, the problem of learning to segment a lung lesion is defined as that of finding F_2.

The solution to these problems will be presented in Sections section 4.2 and section 4.3. For the rest of this chapter, both models will be referred to abstractly as a mapping $F : \mathbb{R}^n \longrightarrow \mathbb{R}^m$.

2.2.2 Neural Networks

Neural networks (NN) are biologically-inspired machine learning models that have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection, and several other domains [52], and will be the backbone of all the work developed in this thesis. Figure 2.4 shows a simple *feedforward forward neural network* with two hidden layers, where, in contrast with other types of networks (such as the recurrent neural network), information only flows from the input, going through each layer, and finally reaching the output. The goal of a *feedforwad network* is to approximate a function F and it is the quintessential deep learning model [30].

The simple architecture presented in Figure 2.4 is called the *multilayer perceptron* (MLP), and it is formed by a set of units that are all connected to each unit in the following layer. The value of the j-th element of layer $k + 1$ computed according to

$$z_j^{k+1} = \sum_{x_i^k \in H_k} w_i^k x_i^k + b^k \tag{2.5}$$

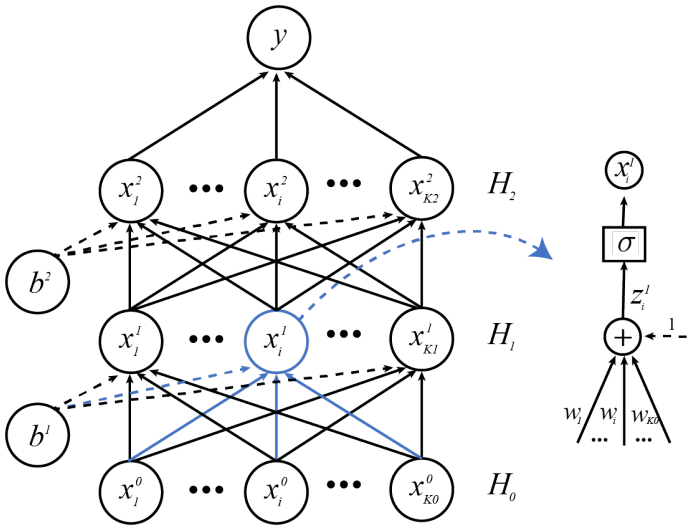


Figure 2.4: Representation of a feedforward network with two stages. The pre-activation z is also presented, followed by the activation function g, that generate the hidden unit x_i^1.

$$x_i^{k+1} = \sigma\left(z_j^{k+1}\right), \tag{2.6}$$

where $\sigma : \mathbb{R} \longleftrightarrow \mathbb{R}$ is called the activation function, H_k is the set of elements in the n-th layer, and b^k is a bias term. Assuming a K_0-dimensional input $\mathbf{X} \in \mathbb{R}^{K_0}$, represented as x^0, and each hidden layer H_k having K_k hidden units, Equations 2.5 and 2.6 can be rewritten as

$$\mathbf{z}^{k+1} = \mathbf{W}^k \mathbf{x}^k + \mathbf{b}^k, \tag{2.7}$$

and,

$$\mathbf{x}^{k+1} = \sigma\left(\mathbf{z}^{k+1}\right). \tag{2.8}$$

with $\mathbf{W} \in \mathbb{R}^{K_k \times K_{k+1}}$, and $\mathbf{b}^k \in \mathbb{R}^{K_k}$.

The overall network function of a MLP with $K+1$ layers can then be written as a composition of each of its hidden layers [30]:

$$F^*(\mathbf{x^0}; \theta) = \sigma_K(\mathbf{W}^K \sigma_{K-1}(...\sigma_1(\mathbf{W}^1(\sigma_0(\mathbf{W}^0\mathbf{x^0} + \mathbf{b}^0) + \mathbf{b}^1)...) + \mathbf{b}^K), \tag{2.9}$$

where θ is a vector containing all the parameters of the network, *i.e.*, all the weight matrices W^0,...,W^k and all the bias terms b^0,...,b^k.

One of the main advantages of these models is given by the *Universal Approximation Theorem*, which states that, given enough hidden units, a network with a single hidden layer can approximate arbitrarily well any continuous function [38]. The key concept is that each unit of the hidden layer computes a representation of the input and then propagates it forward. These models are trained using the gradient

descent algorithm (or other versions thereof) which will be explained in Section 2.2.7. Even though simple feedforward networks are not used directly in current state-of-the-art models, they serve as a basis for recent developments, and are still used as an integrating part of newer architectures.

2.2.3 Convolutional Neural Networks

Convolutional neural networks (CNN) were originally developed for handwriting recognition [54], but a wide range of other applications have ensued, becoming one of the most successful architectures [52]. CNN are inspired by the working principles of the primary visual cortex in animals. Several descriptions of a convolutional neural network have been presented, but this section will rely on the original work of [54] and its recent description by the same author in [52], which presents a CNN as being comprised of two main building blocks: convolutional layers and pooling layers.

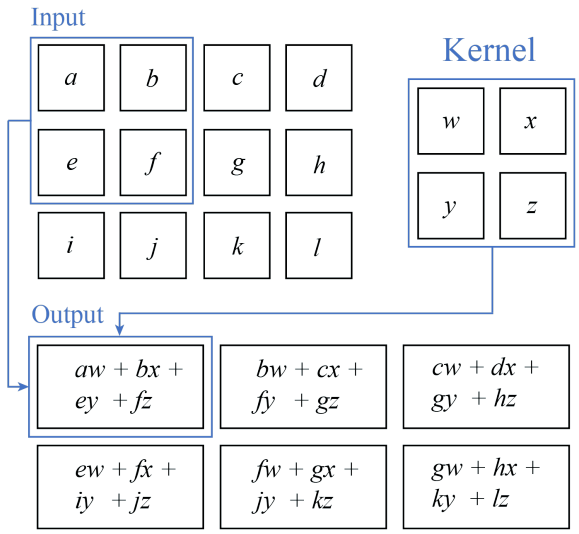


Figure 2.5: Example of 2D convolution with no kernel flipping. Drawn boxes with arrows indicate how the first element of the output is calculated by applying the kernel to the upper-left region of the input tensor. Image adapted from [30]

Convolutional layers are based on the previously described units from the feedforward network, and rely on the convolution operation to calculate the pre-activation, z, which, in the case of images, is extended to a two-dimensional kernel [30],

$$\text{convolution}(\mathbf{X}, \mathbf{K})(i, j) = \sum_m \sum_n \mathbf{X}(m, n)\mathbf{K}(i - m, j - n), \tag{2.10}$$

where $\mathbf{X}$ is a two-dimensional tensor and $\mathbf{K}$ is a two-dimensional kernel. In the context of neural networks, a related function called the cross-correlation is used [30]

$$\text{cross-correlation}(\mathbf{X}, \mathbf{K})(i,j) = \sum_m \sum_n \mathbf{X}(i+m, j+n)\mathbf{K}(m,n), \tag{2.11}$$

which only differs from the convolution by not flipping the kernel. In the context of machine learning programming several libraries also call this function convolution. This convention will be followed throughout the rest of this work, and denoted with $*$. See Fig. 2.5 for an example of a convolution with no kernel flipping applied to a 2-D tensor.

In convolutional layers, the weights that connect layer H_k to layer H_{k+1} are the values of the kernel, and are calculated as shown in Fig. 2.5. This operation can be formalized in the context of NNs by defining each value of the output $\mathbf{X}^{k+1}$ in layer H_{k+1}, which is called the *activation map*, as a result of applying the convolution operation to Equation 2.6, and is given by

$$x_i^{k+1} = \sigma(z_i^{k+1}). \tag{2.12}$$

Here, σ is the previously-defined $\mathbb{R} \longrightarrow \mathbb{R}$ activation function that is applied to the pre-activation, z_i^{k+1}, obtained by convolving the $d \times d$ kernel $\mathbf{K}^k$, with a single $d \times d$ patch of $\mathbf{X}^k$:

$$z_i^{k+1} = \left(\sum_{n=0}^{d-1} \sum_{m=0}^{d-1} \mathbf{X}^k(i+m, j+n) K(m,n) \right) + b_i^k. \tag{2.13}$$

Similarly to the feedforward network, Equations 2.12 and 2.13 can be simplified into:

$$\mathbf{X}^{k+1} = \sigma(\mathbf{X}^k * \mathbf{K} + \mathbf{b}^k). \tag{2.14}$$

where σ denoted the component-wise aplication of the activation function. Besides the variable sizes between kernels, the convolution can also be applied with different shifts in the pixels of the kernel; this is called *stride*. The relationship between size of the image, size of the kernel, and stride is relevant to determine the size of the output. For instances, given an $M \times M$ image and $d \times d$ kernel, with stride s, the resulting output will be of size $N \times N$, where

$$N = \frac{M-d}{s} + 1. \tag{2.15}$$

In practice, it is also common to apply zero-padding to the borders of the input, in order to both reduce the shrinking of the output size, and also give more information about the border pixels. In order to preserve the output size it is common to use a padding size of $\frac{(d-1)}{2}$.

Moreover, often several kernels are applied in a single convolutional layer, yielding one activation map per kernel, which is the result of convolving each kernel with the full space image [77]. Each activation map, $\mathbf{X}_i^k$, can be computed using Equation 2.14 for each kernel, $\mathbf{K}_i^{k-1}$:

$$\mathbf{X}_i^k = \sigma(\mathbf{X}^{k-1} * \mathbf{K}_i^{k-1} + \mathbf{b}^k). \tag{2.16}$$

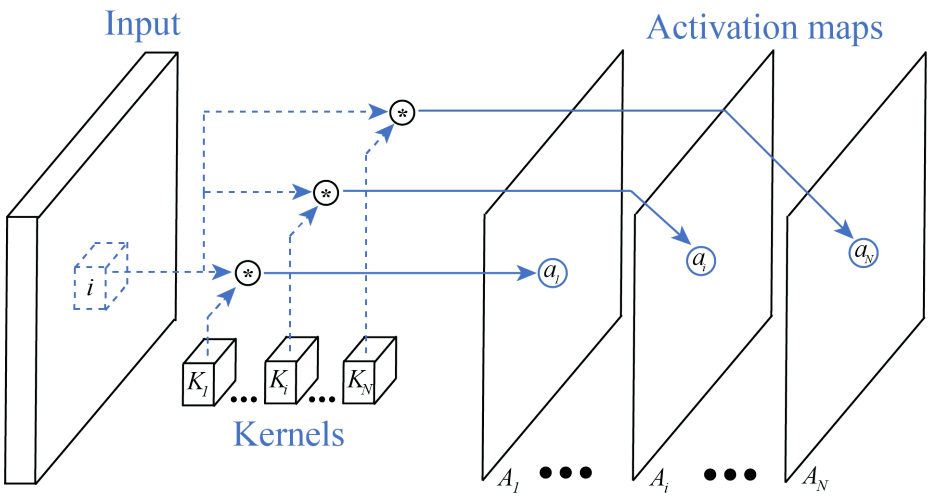


Figure 2.6: Multi-kernel convolution. The inner box i represents the region of the image to which each of the kernels $\mathbf{K}_1$, $\mathbf{K}_i$, and $\mathbf{K}_N$ are applied to generate outputs a_1,a_i, and a_N, in each of the activation maps. To generate the full activation maps $\mathbf{A}_1$, $\mathbf{A}_i$, and $\mathbf{A}_N$ each kernel needs to convolve with the full image. For simplification purposes only three kernels are represented.

Pooling layers are applied alternating with the convolutional layers. However, two or more convolutional layers may appear together in the architecture before a pooling layer. This layer is responsible for making the representation approximately invariant to small translations and oscillations of the input [30] and is usually associated with *downsampling* [52]. Several types of pooling exist, such as the average of a rectangular neighbourhood, the ℓ_2-norm of a rectangular neighbourhood, or a weighted-average based on the distance from the central pixel. One of the most used pooling operations is *max-pooling* which reports the maximum value within a rectangular neighbourhood [30, 47]. This will be the pooling operation used in this work, and is depicted in Figure 2.7.

The application of the convolution operation to neural networks brings several ideas that improve the way these systems encode information. First of all, information is encoded through a typically smaller number of interactions, in comparison to traditional FC neural networks. This is a consequence of using a kernel smaller than the input image (as seen in Fig. 2.6), allowing the network to have fewer parameters. On the other hand, each parameter is used for more than one ”function” in a model, further decreasing the memory requirements and improving the statistical efficiency of the network [30]. Finally, one key interpretation of a CNN is that each layer is trained to extract specific features from the input, with a higher level of complexity found in layers closer to the output [47, 55].

2.2.4 Feature Extraction and Linear Classifiers

Before continuing to the state-of-the-art models in each of the tasks of this work, we will make a short detour by giving some intuition on how these models might be interpreted.

Usually, modern CNN architectures are built with both convolutional and pooling layers, as well as fully connected (FC) layers, *i.e*, layers based on the feedforward networks architecture. In the context of a classification CNN architecture, as those shown in Section 2.2.5, it is possible to interpret the convolutional and pooling layers as *feature extraction sub-networks*, and the latter FC layers as a *decision sub-network* [49]. Depending on the number of hidden layers, this can be a linear classifier (zero hidden layers), or a non-linear classifier (one or more layers). Specifically, in the case of object detection, the decision sub-network is often called a detection network. In the case of a classification problem, it is quite common to have a *softmax unit* as the last activation function [52], which is given by $g : \mathbb{R}^d \longrightarrow [0,1]^d$, such that

$$(g(\mathbf{z}))_l = \frac{\exp(z_l)}{\sum_n \exp(z_n)}, \tag{2.17}$$

thus $(g((z))_1, ..., g((z))_d)$ can be interpreted as the posterior probability classes.

In the case of segmentation CNN architecture, as the ones presented in Section 2.2.6, it is harder to formalize an abstraction for a decision sub-network, as these networks usually don't use FC layers, and, instead, convert the final feature map to pixel-by-pixel (or voxel) classification using a 1x1 convolutional layer.

Another interpretation of a deep CNN is that the last layer works as a linear model applied to a transformed input $\phi(\mathbf{X})$, where ϕ is a non-linear transformation. Under this interpretation, all the layers work to transform the input into a representation that can be used as an input for a classification model. This insight is used to motivate recent work in which the last layer of a deep NN is substituted by another classifier [22, 46, 95] or the penultimate one is used as an input for other models, both isolated or mixed

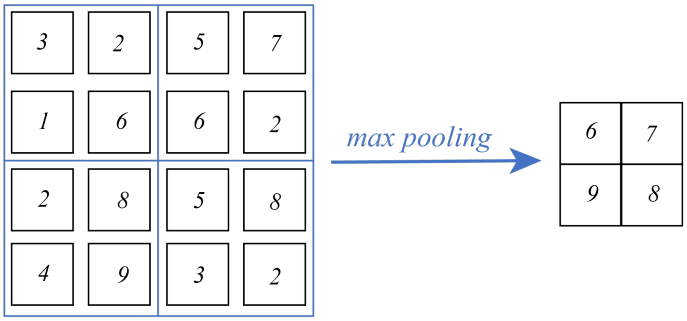


Figure 2.7: Image represents max-pooling operation. A *max* operation is applied to each rectangular neighbourhood (denoted in blue). In this case the neighbourhood is a 2x2 window with a stride of 2.

with engineered features [15, 82].

Figure 2.8 summarizes some of the key concepts. These were intended to deliver some intuition behind a CNN architecture with fully connected layers, in the context of a classification task.

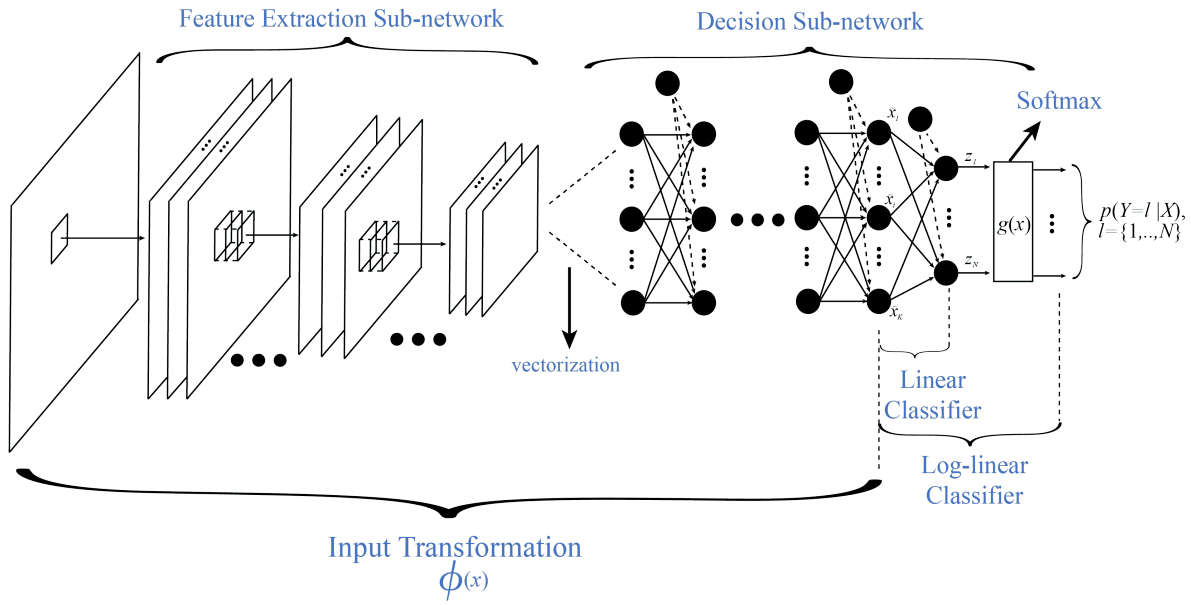


Figure 2.8: Representation of a CNN architecture with fully-connected layers at the end, designed for a classification task. The feature extraction is performed by the convolutional layers followed by pooling, while the fully-connected layers at the end can be interpreted as a classifier.

2.2.5 Object Detection Networks

The generic object detection task in computer vision closely relates to the task of finding lesions in medical images. In recent years we have seen a large improvement that was mostly due to new advances in DL models applied to object detection. This leads to a large knowledge transfer between fields, as will be shown, and enabled several deep learning models to be used in medical imaging.

2.2.5.A Image Classification

The generic object detection task saw strong improvements after the development of deep CNNs for image classification, starting with *Alexnet* [47]. Its results were mostly due to an increase in the number of kernels per layer, as well as stacked convolutional layers [108]. This work was then followed by the *VGG-16* [89], which increased the number of hidden layers to 16. Even though with an extremely large number of parameters (originally it was trained during 2-3 weeks on 4 GPUs), it became the preferred choice for extracting features and served as the backbone for the early object detection models

based on DL. The next great improvement in the image recognition task was the use of residual connections, which yielded one of the state-of-the-art models, the ResNet [36]. The residual connection will be presented in Section 4.1.1. Almost in parallel (even if slightly earlier), the inception module [94] was produced, allowing a strong reduction of the size of the models with almost no detriment to its classification accuracy.

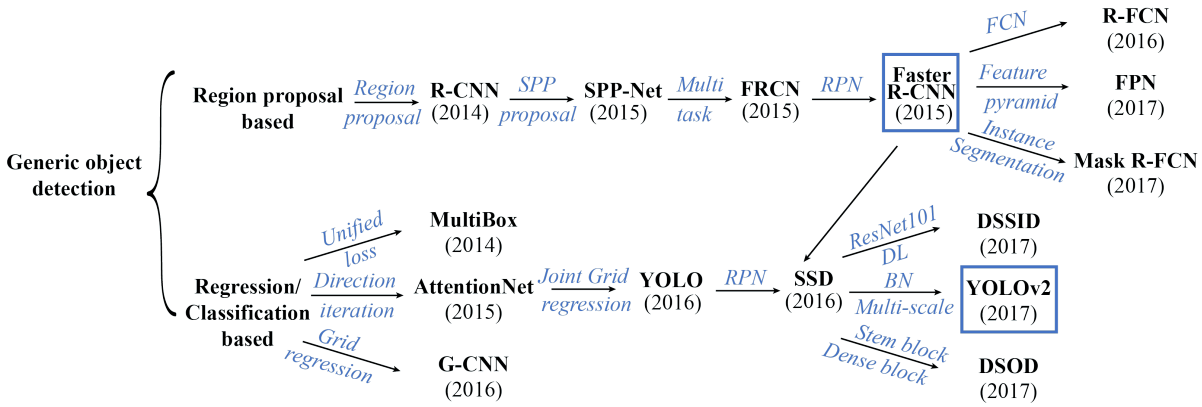


Figure 2.9: Two main frameworks for DL object detection models: region proposal based and regression/classification based. SPP: Spatial Pyramid Pooling, FRCN: Fast R-CNN; RPN: Region Proposal Network; FCN: Fully Convolutional Network; BN: Batch Normalization; DL: Deconvolutional Layer. Adapted from [108].

2.2.5.B Generic Object Detection Models in Computer Vision

Finally, getting closer to the final architectures that are currently used to tackle detection problems, generic object detection models followed two categories (Fig. 2.9): (1) architectures that rely on a two-step process, first generating region proposals, then classifying each proposal into different object categories; (2) architectures that applied an unified approach to achieve both categories and locations, by approaching the task as a regression or classification problem [108]. Recent region proposal models have in their core three main components: a backbone network that is responsible for extracting the features (feature extraction sub-network), a detection sub-network that uses the features to find a vector of coordinates and a bounding box for each object found, and a Region Proposal Network (RPN). The intermediate RPN sub-network is convolutional and produces a prediction of object bounds and scores, and connects the features to the sub-detection network [81]. These sub-networks are usually trained separately. On the other hand, one-step frameworks map straight from the image features of the last feature map, to the bounding box coordinates and class probabilities. You Only Look Once (YOLO) [79], one of the recent models following this scheme, performs this task by first dividing the input image into a grid and then predicting both a bounding box, as well as a confidence score for each class. Finally, the lastest improvement in object detection was the introduction of the Retina-Net [58], which works as a single unified network composed of a backbone and two task-specific sub-networks dedicated to the

classification and box regression tasks.

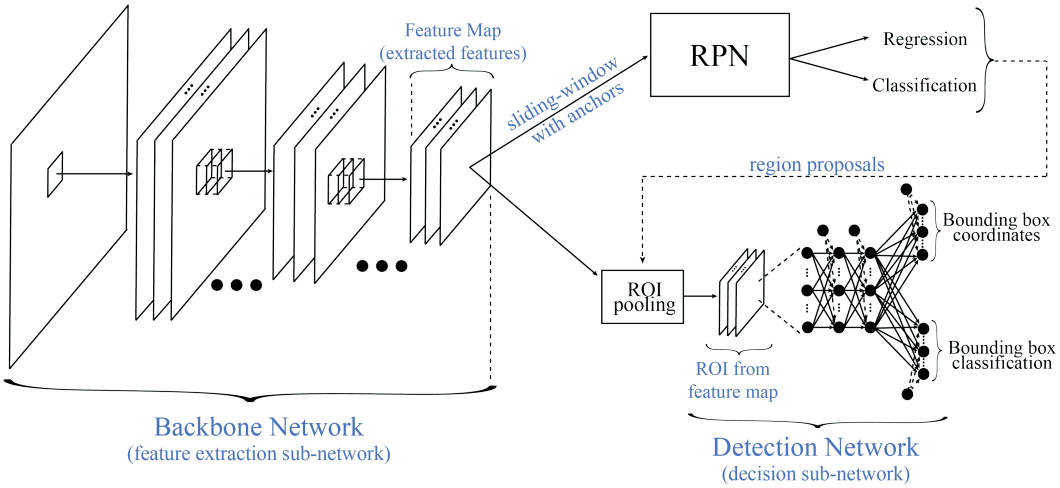


Figure 2.10: Scheme of the Faster R-CNN architecture. The feature map produced by the backbone network will be used by the RPN, in a sliding-window fashion, to generate a set of rectangular object proposals that are used by a Region of Interest (ROI) pooling layer to extract feature map regions. These are used by a detection network to produce the true bounding box and a classification for each object found. Details on the region proposal mechanism are given in Fig.2.12

.

2.2.5.C Lung Lesion Detection Models

Even though these models were developed to find a huge number of classes, not comparable to those in a lesion detection task, several approaches in the field were based on these architectures. However, one problem arises when comparing medical imaging with RGB images: the later is 2D, whilst the former is 3D. Several approaches have only relied on models which use information from single slices of the original volume to perform their tasks, therefore ignoring the intrinsic 3D nature of the data [64]. Recently, there has been an increased interest in using 2.5D models, which are based on multiple slices of the image data [33], as well in using pure 3D architectures, where the convolution operations of the DCNN are performed on the 3D data [16]. Although there are clear advantages in using 3D DCNNs, there are also severe limitations, due to very high GPU memory requirements and higher potential for overfitting [42].

Most of the works in computer vision try to tackle the problem of finding the *detection model* defined in Equation 2.2, with the formulation of D as a set of the coordinates of each lesion and their bounding box. Some have engaged this task based on a region-proposal-based model, mostly using the *Faster R-CNN* [81], the architecture that first introduced the RPN (Fig. 2.12(a)), and which was able to deliver real-time object detection in several real-world applications [108]. Details of the Faster R-CNN are described in Figure 2.10. Most of the works that followed that approach had to adapt those architectures

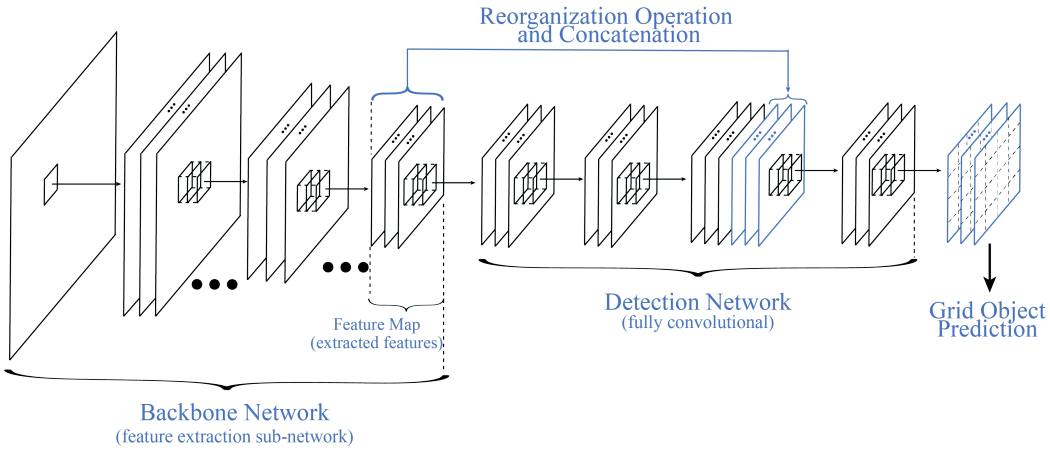
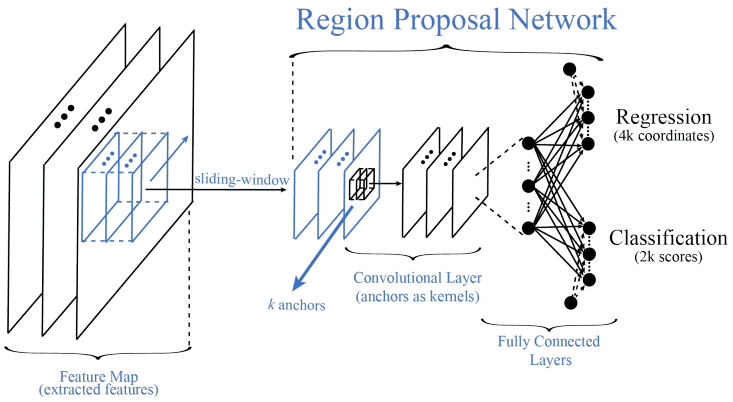


Figure 2.11: Scheme of the YOLOv2 architecture. A backbone network produces a feature map that is used by a fully convolutional detection network, with no pooling layers to generate the final output (presented in Figure 2.13(b)). The feature map are reorganized through the operation described in Figure 2.13(a), and concatenated to a layer inside the detection network, in order to provide higher grain features.

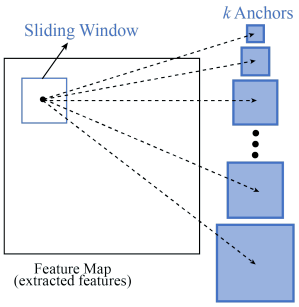
to the different requirements of medical imaging. The addition of deconvolution layers to a 2.5D *Faster R-CNN* [19] has produced state-of-the-art performance for non-3D architectures. A 2D Faster R-CNN relying on boosting approaches for both the RPNs and the decision sub-networks was proposed in [105]. Recently, a hybrid model that performs both detection and classification of lung lesions, the *DeepLung*, uses a 3D *Faster R-CNN* for the first stage [109]. The architecture of this model was upgraded using both residual connections, and dense connections. These last are simple extensions of residual connections that allow to increase the propagation of information across different levels of the NN. This model was then called, a 3D *Faster dual-path R-CNN* and, even though with heavy computational and memory requirements, it was able to achieve state-of-the-art results in the lung detection task.

Several works have been based on a newer version of the *YOLO*, the *YOLOv2* [80], which combines some strategies used in the *Faster R-CNN*, such as the use of anchor boxes and dimension clusters, with further improvements, such as batch normalization and multi-scale training. These enhancements led to a slight increase in accuracy, and a vast speed improvement in comparison to the *Faster R-CNN*. One of the main goals of *YOLOv2* was to make the representation easier to learn. This extremely fast model was also applied, with small adjustments, to the lesion detection task [7], leading to competitive results.

Finally, the *RetinaNet* was also very recently applied to this task, by being modified to allow 3D images as input, and integrated an end-to-end deep learning based approach for lung cancer screening. The full pipeline of this approach led to better-than-human results in the task of classifying a lesion as malignant or non-malignant [6].

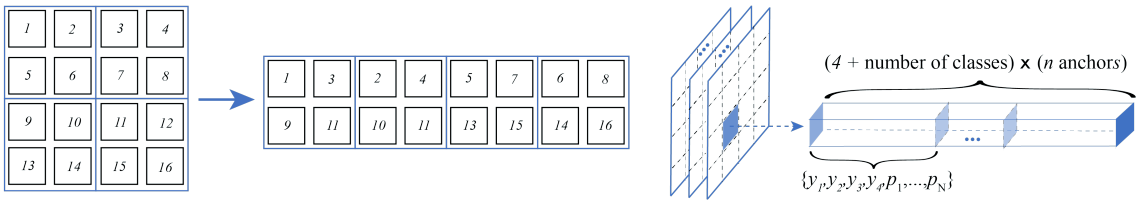


(a) Region Proposal Network (RPN)



(b) Sliding-window and anchors

Figure 2.12: Region proposal mechanism. a) In a sliding window fashion, patches of the feature map are extracted, which are convoluted with k predefined anchors, to generate a fixed sized vector. These are used first by a general FC layer and then fed to two sibling FC layers: a regression and a classification layer. b) Each window from the sliding window will be convoluted with the k rectangular, fixed sized, anchors.



(a) Reorganization Operation

(b) Structure of the output

Figure 2.13: Details on the YOLOv2 architecture. a) The reorganization operation takes every alternate voxel and places it in a different positon, reducing, in the example, the size of the input ot half and creating 4 new channels. b) The output of the network is dividided in a grid, and assigns a position and box-size vector $y = [y_1, y_2, y_3, y_4]$, as well as a probability distribution over each class for a set of n anchors (similar to Fig. 2.12(b)), to each entry.

2.2.6 *U-Net* and other Segmentation Networks

2.2.6.A Before the *U-Net*

In the context of computer vision, there have been multiple approaches to image segmentation with deep learning models. Early work in the field used patch-wise CNN [23], which are architectures that look at a small rectangular neighbourhood (often square) in the image, and output a classification for that region's central pixel. By running this operation for all the regions in an image, each pixel in the image is associated with a label. The specific approach in [23] also used other techniques such as *superpixels* and *segmentations trees*. Following the large breakthrough in image classification with deep learning models [47, 89], a fully convolutional network was also developed for this task with a structured

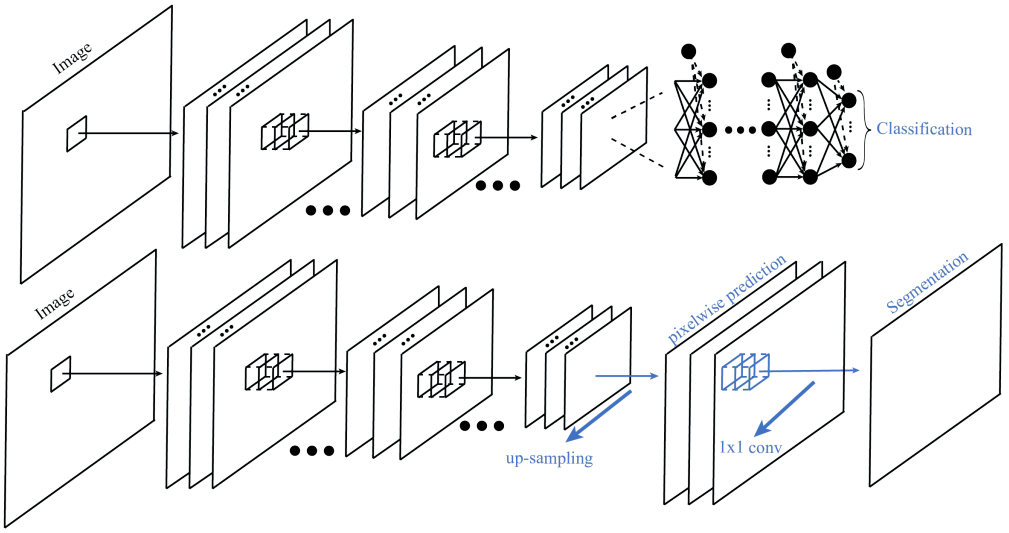


Figure 2.14: Comparison between a typical classification CNN (on top) and a fully convolutional network (below). As depicted, both architectures share a feature extraction sub-network. The changes to the fully connected network are the substitution of the fully connected layers by a deconvolution and a 1×1 convolution.

output [60]. This network relied on a pre-trained CNN (originally AlexNet [47], VGG-16 [89] and the first version of the InceptionNet [94] were tested), with two major modifications to turn it into a segmentation CNN:

1. The last layer was changed to a 1×1 convolutional layer, therefore allowing the network to output a spatial map (*i.e.*, a tensor that assigns a label to each pixel), instead of a classification label vector.

2. In order to ensure that the spatial map has the same dimensions as the input image, an up-sampling layer was also added (also called deconvolution layer), which implements the deconvolution operation, or more accurately the transposed convolution (Fig. 2.15).

The work in [60] also uses residual connections, which will be described in Section 4.1.1.

2.2.6.B *U-Net* and its application to medical imaging

The *U-Net* model [83] was built upon the previous fully convolutional network [60]. It is also a fully convolutional network, but it is trained from scratch and can be divided in two sub-components: an encoder (a down-sampling path that produces a high-density low-resolution feature map) and a decoder (an up-sampling path, which translates the features into the final spatial segmentation map) (Fig. 2.16). The down-sampling operation is the already presented max-pooling (Fig. 2.7), while the up-sampling operation is the deconvolution (Fig. 2.15).

(a) Non-strided **(b)** Strided

Figure 2.15: Examples of non-strided (a) and strided (b) deconvolution. The output of the operation is represented on the top, in light blue, while on the bottom, the input for the operation is represented in dark-blue. The black pixel is obtained by convolving the pixels inside the dashed square in the input, with a 3×3 kernel.

In order to increase the feature content in the decoder path, at each up-sampling step, a concatenation is performed with feature maps from the encoder path of corresponding size. This model is currently the state-of-the-art for several image segmentation tasks, being widely used in medical imaging [87]. One major adaptation of this model for anatomical and lesion segmentation in medical images was the *nnU-Net* (*no-new-U-Net*) [42], which, without changing any of the core components of the *U-Net*, was able to achieve state-of-the-art results in several medical segmentation tasks, one being lung tumour segmentation. It relies on extensive data curation techniques, as well as on a cascade scheme which combines two *U-Net* models in sequence, allowing the use of a 3D architecture [16] without overflowing the GPU memory.

In parallel, another U-Net-based architecture was developed in order to tackle the lung lesion segmentation task. This work introduced a *central focused double branched CNN*, or *CF-CNN*, which relied on a novel pooling able to pool features mostly from the center of the feature map [101]. The goal of this approach was to have the model focused in the center of the input image, assuming a VOI around each lesion was already detected. That work was able to achieve state-of-the-art results, at the cost of huge amounts of training time, as well as an extremely high computational requirement.

Recently, residual connections have also been added to the U-Net [4, 18], with positive results, but, at the time of writing, no *U-Net* with added residual connections was found to have been applied to lung lesion segmentation.

2.2.7 Loss Function and Model Training

As mentioned in Section 2.2.2, the goal of a neural network is to aproximate an unknown function F with a model $F(\mathbf{X}; \theta)$. In the case of supervised learning, the set of paremeters $\theta = \{\mathbf{W}^k, \mathbf{b}^k\}$ for all layers k that define F are learned from the training data, a set of labelled pairs of observations and

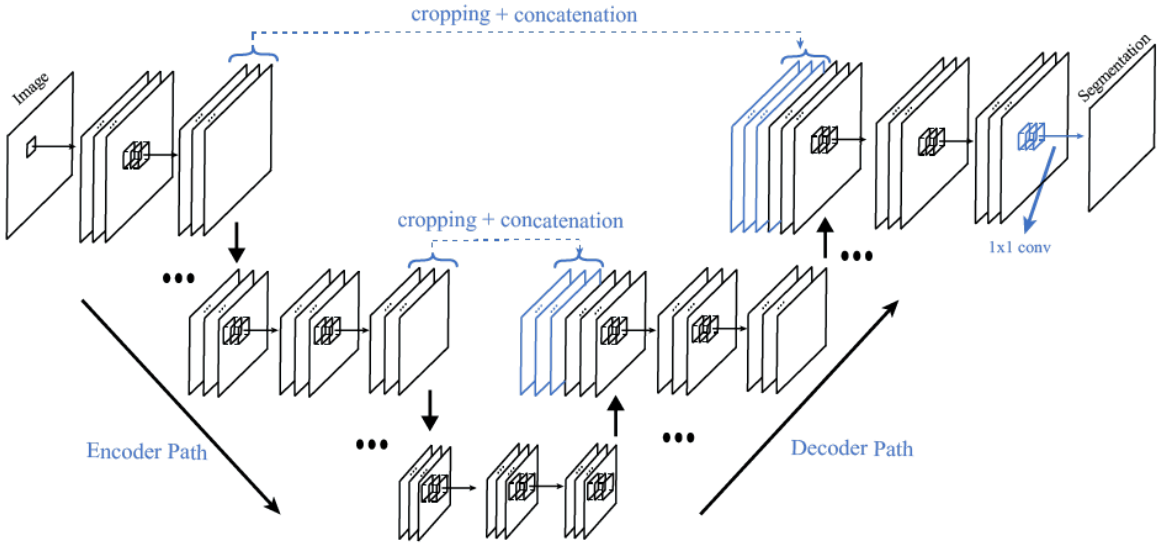


Figure 2.16: Descending arrows in the encoder path denote a 2×2 max-pooling operation, while ascending arrows in the decoder path denote a 2×2 up-convolutional operation. Every convolution uses a 3×3 kernel. Blue dashed arrows represent the cropping of the feature map from encoding path and concatenation to the feature map of the decoding path. The original *U-Net* architecture has four down-sampling and up-sampling operations, yielding five levels. These levels were omitted for representation purposes, therefore only showing three levels.

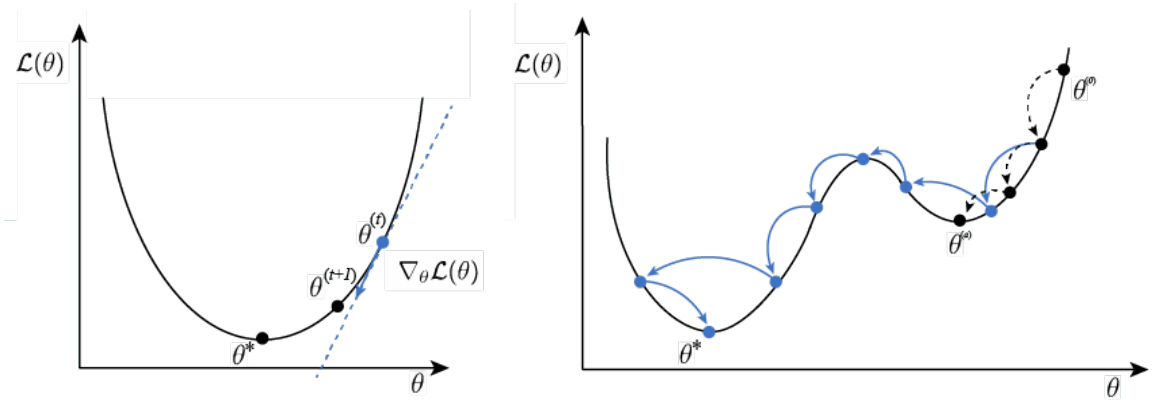
outcomes, $\mathcal{D} = \{(\mathbf{X}^1, Y^1), ..., (\mathbf{X}^M, Y^M)\}$. The goal will be then to find the best set of parameters θ, that will minimize some measure of difference between the true output in the training data Y, and the estimated $\hat{Y}$, given its corresponding observation X:

$$\theta = \arg\min_{\theta} \mathcal{L}(Y, F(\mathbf{X}, \theta)) = \arg\min_{\theta} \mathcal{L}(\theta), \tag{2.18}$$

where is the function to be minimized, and that will be defined in the context of the specific problems at hand. The goal is to have this function to match as close as possible the ideal metric that F is trying to minimize. We will call this function *loss function*, but it may also be called objective function, cost function, or error function. The goal will be then to ideally find the θ^* that minimizes $\mathcal{L}$, such that, for any $\theta \in \mathbb{R}^d$, $\mathcal{L}(\theta^*) \leq \mathcal{L}(\theta)$ [11].

2.2.7.A Gradient Descent

The estimation of θ is made using iterative descent methods that proceed in small steps in the optimal direction, until a stopping criterion is met. At the core of the methods used is the gradient descent that updates the parameters at each iteration t as follows: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \lambda_{(t)} \nabla \mathcal{L}(\theta^{(t)})$ [11, 30], and its intuition is presented in Fig 2.17(a). Additionally, as the method is applied to the full training dataset



(a) Gradient Descent

(b) Gradient Descent with and without momentum

Figure 2.17: The goal of the gradient descent is to iteratively converge from an initial value of the loss function $\mathcal{L}(\theta)$ to the optimal θ^*. a) Indication of the use of the gradient estimate at $\theta^{(t)}$ to find a value of the new iteration $\theta^{(t+1)}$. b)Comparison of GD with and without momentum. If the loss function has a local minimum, it might get stuck (black arrows). The usage of momentum is mimmicking a ball descending down a hill: as it accumulates momentum it rolls faster enabling it to descend quicker. If it has enough momentum it might roll up a hill, thus being able to descend to lower minimum.

$\mathcal{D} = \{(\mathbf{X}^1, \mathbf{Y}^1), ..., (\mathbf{X}^M, \mathbf{Y}^M)\}$, with M examples, the loss function needs to be defined as:

$$\mathcal{L}(\theta) = \frac{1}{M} \sum_{i=1}^{M} L(F^*(\mathbf{X}^i; \theta), Y^i) = \frac{1}{M} \sum_{i=1}^{M} L_i(\theta). \tag{2.19}$$

And the gradient of the loss function is then:

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{M} \sum_{i=1}^{M} \nabla_\theta L_i(\theta). \tag{2.20}$$

Unfortunately, finding such set of parameters is extremely hard in the context of deep learning as the number of model parameters, d, is extremely large (within the range of the millions), and the loss function is non-convex, with a huge number of saddle points and local *minima* [30]. Another major limitation to the gradient descent method is that it requires a full pass by all the pairs of examples in the training set, when computing $\nabla_\theta \mathcal{L}(\theta)$, before updating the parameters in each iteration. For large datasets this task takes a long time [12, 30] and requires a large amount of memory [52], therefore becoming impractical.

2.2.7.B Stochastic Gradient Descent

The *stochastic gradient descent* (SGD) method updates the gradient estimation with a single training example by sampling it uniformly (choosing a random $r \in [1, ..., M]$) [12, 30]:

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{M} \sum_{i=1}^{M} \nabla_\theta L_i(\theta) \tag{2.21}$$

$$\backsimeq \nabla_\theta L_r(\theta) \tag{2.22}$$

The result is an unbiased, but extremely noisy estimate of the gradient, as the examples are chosen uniformly at random. [12, 30]. In practice a small amount of training examples can be used, the so called batch, which will yield a less noisy (more examples lead to a better estimation of the gradient), but still unbiased estimate of the gradient [30, 52]:

$$\nabla_\theta \mathcal{L}(\theta) \backsimeq \frac{1}{|B|} \sum_{i \in B} \nabla_\theta L_i(\theta) \tag{2.23}$$

where the batch, B, is defined as a set of $|B|$ randomly sampled examples $\{r_1, ..., r_B\}$, with $|B| \ll N$. The term stochastic here refers to the fact that an estimation of the gradient is being computed, which will be a noisy sampling of the average gradient computed over the full training dataset [52]. Even though this is a noisy estimate, the approach typically is shown to result in faster convergence towards a good set of parameters [54]. The update rule for the weights can be finally defined:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \lambda^{(t)} \frac{1}{|B|} \sum_{i \in B} \nabla_\theta L_i(\theta), \tag{2.24}$$

where $\lambda^{(t)}$is called the *learning rate*.

It is now possible to iteratively estimate the weights of the model, but several questions arise. Will the SGD algorithm be able to minimize such a highly non-convex function as the loss function in hand? Can we efficiently calculate and update the parameters of both fully connected layers and convolutional layers? We will begin by answering the first question, and will after give some key intuition on the second one in the next sub-section.

2.2.7.C The Adam Algorithm

As previously mentioned the optimization problem being solved is extremely hard, therefore, the guarantee of convergence to a global minimum might be impossible to achieve. Even though SGD is a good first approach to finding the parameters θ, it is highly dependent on initialization, and has several shortcomings [12, 30]. Instead of using the plain SGD, most recent works use other iterative descent

methods, with one of the most used being the *adaptive moment estimation* method [45], or *Adam*, which will be the one adopted in this work. This model combines the usage of two major concepts: (1) a moving average of the gradient instead of the gradient itself (as is the case in SGD); (2) an adaptive learning rate taking into account information of the first and second moments of the gradient, which correspond to the mean and the uncentered variance of the gradient.

Before discussing these two concepts, lets us first define the concept of *momentum*. First-order gradient methods, like SGD, have trouble navigating "ravines" (i.e. areas where the level curves of $\mathcal{L}(\theta)$ curve steeply in one of the dimensions), usually getting stuck in local minimuma [30]. In order to deal with this, some algorithms add a small fraction γ of the previous update vector,v_{t-1} to the current update vector, v_t. This is called the *momentum term*, and it is given by

$$v^{(t+1)} = \gamma v_t + \lambda \nabla_\theta \mathcal{L}(\theta) \tag{2.25}$$

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - v^{(t+1)} \tag{2.26}$$

Some intuition on the momentum term is given comparing gradient descent to a ball rolling down a hill. With enough momentum, it is able to avoid getting stuck in local minima. (Fig 2.17(b)). This also allows the gradient to converge with less oscillations. As a result, it is possible to converge faster, and ultimately achieve a better solution to Equation 2.18 [30, 74]. Adam uses a similar mechanism by storing a decaying average of past gradients, as well as an exponentially decaying average of past squared gradients [45]:

$$m^{(t+1)} = \beta_1 m^t + (1-\beta_1)\nabla_\theta \mathcal{L}(\theta) \tag{2.27}$$

$$v^{(t+1)} = \beta_2 v^t + (1-\beta_2)(\nabla_\theta \mathcal{L}(\theta))^2, \tag{2.28}$$

where, β_1 and β_2 are two hyperparameters that need to be tuned. The update rule for Adam is then [45]

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \frac{\lambda}{\sqrt{\hat{v}^{(t)}} + \epsilon} \hat{m}^{(t)}, \tag{2.29}$$

where, $\hat{v}^{(t)}$ and $\hat{m}^{(t)}$ are the bias-corrected first and second moment estimates, and ϵ is a term to avoid a zero in the denominator (usually set to 10^{-8}). Details on applications and limitation of the Adam algorithm are given in [30], and the pseudo-code for the algorithm can be found in [45]. Note that more recent versions of the Adam algorithm have been developed [20, 107], but weren't considered in this work.

2.2.7.D Backpropagation algorithm

Finally, an efficient way to calculate $\nabla_\theta L(F^*(\mathbf{X};\theta), Y)$ is needed. To do so, the gradient backpropagation method is used [11, 54]. This algorithm mainly relies on the application of the chain rule of derivation to each layer. The core of this algorithm is that, at each of the hidden units, the derivative of the loss function $\mathcal{L}(\theta)$ [52] is calculated with respect to the input of the following unit. The way that the algorithm efficiently implements the derivatives is by dividing this process into two steps: the first stage is the forward pass (already discussed in Sub-section 2.2.2), where the outputs of each hidden unit are computed according to Equations 2.5 and 2.6. The second stage is called the backward pass, and starts by computing the derivative of the loss function with respect to the output, and then applies the chain rule of derivatives at each hidden layer. The mathematical details are shown in Appendix A.

2.3 Generalization and Overfitting

In this section, the concepts of model generalization, overfitting, and underfitting are reviewed. Section 2.3.2 will describe how to evaluate generalization error, focusing on DL in the medical imaging field, which will be followed by the introduction of several changes to both the network architecture (residual connections), as well as the training of the model (regularization and dropout), that will allow for better generalization.

2.3.1 Overfitting, underfitting and the generalization error

Until now, we have talked about building a model $F(\mathbf{X};\theta)$ through the use of a trainingset, by minimizing a loss function $\mathcal{L}(\theta)$. Yet, the ultimate goal in machine learning is to perform accurately in an **unseen** set of inputs $\{\tilde{\mathbf{X}}_1, ..., \tilde{\mathbf{X}}_{K_0}\}$, usually called the *test set* [30, 65]. This goal can be summarized as minimizing the so called *generalization error*, which amounts to the expected error, computed using the predefined metric, and averaged over future data [65]. This can be estimated by computing the error in an external set of examples, separated from the training examples, which its called *test error*.

The comparison of the generalization error and the training-error yields the concepts of underfitting and overfitting (Fig. 2.18), as functions of the so-called model capacity. Figure 2.18 illustrates te relationship: generally, as the model capacity is increased, there is an improvement of the results in the training-set, but the performance in test-set is going to decrease. This is caused by an increase in the sensitivity of the model to small fluctuations in the training-set, that are unlikely to be due to variations of interest [65]. The opposite is expected to occur as the model's capacity is decreased, as the model loses sensitivity to any fluctuations in the data, becoming unable to learn its task. In specific, one can prove that the discrepancy between training error and generalization error is bound from above by a quantity

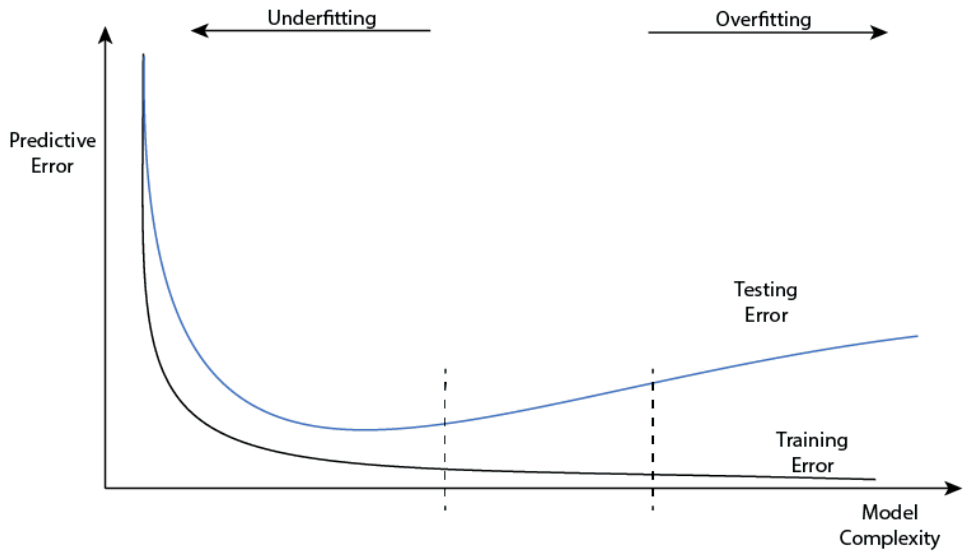


Figure 2.18: Relationship between capacity and error. The goal is to determine the optimal capacity regime which is confined between the underfitting and overfitting regime.

that grows as the model capacity grows by shrinking as the number of training examples increase [99].

Note that this classical perspective is currently being reshaped to a ”double descent” risk curve [10], in order to address some of the recent observations in modern machine learning practice. Yet, as no exhaustive study has been carried out in the context of CNNs and deep networks in non-benign datasets, such as the case of medical imaging, and therefore this work will address the problem of minimizing the generalization error according to the classical perspective.

2.3.2 Generalization error in medical imaging

In the field of medical imaging, evaluating the generalization error is sometimes a difficult task. This is due to the large size of current deep learning models [33]. Medical images have features that are shared within a single patient, but that are not relevant for the task. Nonetheless, due to its high capacity, these will be captured by the model, increasing its performance in the training-set, without encoding any relevant information [28, 59]. Therefore, slice-wise separation of the dataset within a patient is not suitable for a thorough analysis of the generalization ability of the model. Accordingly, a clean patient-wise example separation between training and test set is of paramount importance to transition the models to clinical practice [59]. Furthermore, the sheer amount of variation between acquired images [64], due to the inherent disparity in the acquisition process, as well as the multiple possible machines used for the acquisition [28, 64], lead to a high probability of overfitting to specific features that are not shared by all images. Ultimately, a good practice in this field is to have an external dataset used for the

test set, preferably from a different institution than the ones which provided the data for the training-set, in order to effectively assess the generalization of the model [70].

2.3.2.A No free lunch theorem

However, even through the use of extensive testing in cleanly separated, or external datasets, the best model that is chosen is never a universal best model, as this hypothetical model does not exist - this is called the *no free lunch theorem* [103]. This theorem states that "averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points" [30]. At the core of this statement is the set of the assumptions that are chosen for each task, which have no certainty of working as well in other domains. This leads to the need for working towards a set of models and algorithms that best solve each problem [65]. In this case, a set of NN architectures that are developed for the problems at hand (Equations 2.2 and 2.4), specified for the medical imaging domain.

2.3.2.B Validation set

When trying to assess the generalization error during training, it is good practice to use a different dataset than the external test set, as by assumption it isn't accessible during training [65]. This dataset, called *validation set*, and is used to fine-tune any hyperparameter of the model, which includes its capacity and complexity, as well as the already mentioned learning rate, and any other parameter not related to θ [30]. If the model complexity was chosen by looking directly at the generalization error of the test set, the model choice would be biased, as there would exist an over-optimistic performance estimation of the model by overfitting of the test data [11].

2.3.2.C Limiting the Model's Capacity

The model capacity can be controlled by limiting the *hypothesis space* of the model - the set of functions that the learning algorithm will be restricted to learn [65]. In the context of NN, this can be achieved primarily by reducing both the number of hidden layers and the number of hidden units (or the size and number of kernels, in the case of CNN) [30]. Note that, due to the highly complex problem that is being tackled, in fact, what is achieved is not the representational capacity of the model, but its effective capacity, defined by the parameters θ to which the learning algorithm converges to [30]. Therefore, another away to effectively control the capacity of the model is to tune the learning algorithm in order to find a model with an effective capacity that is within the optimal capacity regime. The easiest way to do so is to either stop the iterative learning algorithm when the generalization error starts increasing, or save several models during the training process and select the best one *a posteriori*.

Currently, in medical imaging, the application of early stoping is quite common when dealing with complex models. However, by stopping the training early, or through the application of model complexity restrictions, which include both the reduction of the number of layers and the number of hidden units in the network, the NN might not be able to learn enough from the training set to fit any relevant model. There is then a trade-off between accuracy and generalization that is hard to tackle.

Currently, in the field of medical imaging, choosing a model that does not underfit is quite easy, simply by both increasing the number of parameters and letting the model increase its complexity during the training. However, by stopping the model too early and/or imposing too strict complexity restrictions - there is a risk that the NN will not be able to learn a good model from the data. There is then a trade-off between accuracy in the training set and generalization, which is hard to tackle [65]. In the rest of this chapter, several alternatives will be presented to address this issue.

2.3.3 Parameter Regularization

One of the most common approaches to limit overfitting is to control the effective complexity of the model by adding a parameter penalization term $\Omega(\theta)$ to the loss function, which limits the capacity of the model [11, 30]. The resulting regularized loss function can be written as such

$$\mathcal{L}_{\text{reg}}(\theta) = \mathcal{L}(\theta) + \alpha\Omega(\theta), \tag{2.30}$$

where $\lambda \in \mathbb{R}_+$ is a hyperparameter that weighs the relative contribution of the parameter penalization term, $\Omega(\theta)$, with respect to the loss function, $\mathcal{L}(\theta)$. In practice, only the weights of the affine transformation at each layer are penalised, since the biases usually require less data to accurately fit. This leads to less induced variance when leaving the bias unregularized [30]. Due to its already well-tested advantages when working with neural networks, these regularization methods will be extensively used throughout this work.

2.3.3.A ℓ_2 regularization

One common choice term used is ℓ_2 regularization, which drives the weights closer to the origin by defining $\Omega(\tilde{\theta})$ as the squared ℓ_2 norm of all the weights. This type of regularization is commonly known in the context of neural networks as *weight decay*, but can also be called ridge regression or Tikhonov regularization [30] . With this regularization, the loss function can be described as

$$\mathcal{L}_{\text{reg}}(\theta) = \mathcal{L}(\theta) + \frac{\alpha}{2} w^\top w, \tag{2.31}$$

where w is the vector containing all the weights in the network. The gradient of $\mathcal{L}_{\text{reg}}$ with respect to w is

$$\nabla_w \mathcal{L}_{\text{reg}}(\theta) = \nabla_w \mathcal{L}(\theta) + \alpha w \tag{2.32}$$

Lets analyze this regularization, by considering a single gradient step to update the weights w:

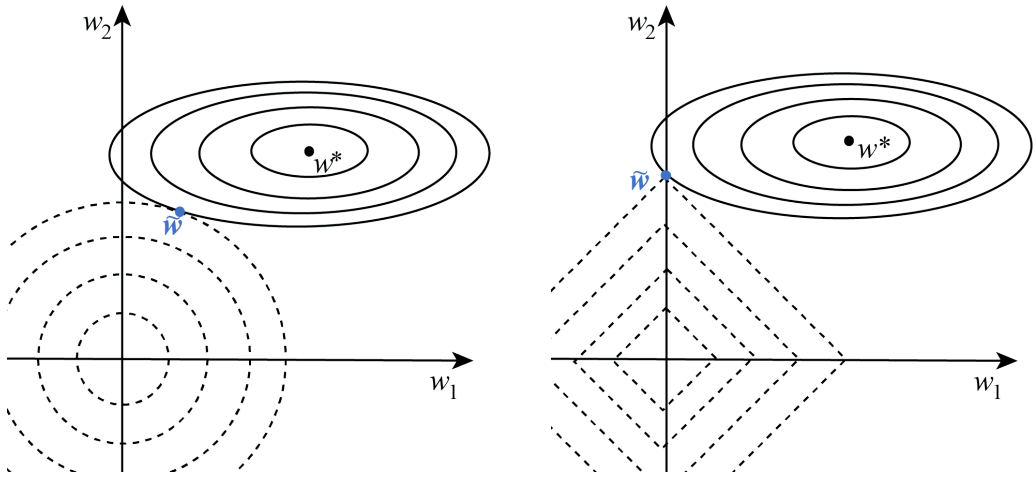
$$w \longleftarrow w - \lambda(\alpha w + \nabla_w \mathcal{L}(\theta)) \tag{2.33}$$

$$w \longleftarrow (1 - \lambda\alpha)w - \lambda \nabla_w \mathcal{L}(\theta)), \tag{2.34}$$

where λ is the learning rate that defines the update step as seen in Subsection 2.2.7.B. Therefore, at each update step the term $(1 - \lambda\alpha)$ will multiplicatively decrease the weights $\mathbf{w}$. Taking this analysis one step further, it is possible to define a quadratic approximation of the loss function from Equation 2.31 in the neighborhood of the optimal solution of $w^* = \arg\min_w \mathcal{L}(w)$:

$$\mathcal{L}(\theta) = \mathcal{L}(\theta^* + \frac{1}{2}(w - w^*)^\top \mathbf{H}(w - w^*), \tag{2.35}$$

where $\mathbf{H}$ is the Hessian matrix of $\mathcal{L}$ with respect to the $\mathbf{w}$ computed in $\mathbf{w}^*$. The missing-first order term is due to the fact that the gradient will be zero at the minimum w^* [11]. The effect of the weight decay is that w^* will be rescaled along the axis defined by the eigenvectors of $\mathbf{H}$ [30]. A graphical interpretation of this effect is shown in Figure 2.19(a), where the directions of the eigenvalues that are large are less affected by the regularization [30].



(a) ℓ_2-regularization **(b)** ℓ_1-regularization

Figure 2.19: Illustration on the effect of ℓ_2 and ℓ_1 regularization on the optimal $\mathbf{w}$. The dashed circles and dashed lines represent contours of equal value of the ℓ_2 and ℓ_1 regularizers respectively, whereas the solid elipses denote countours of equal valued unregularized loss function around the optimal weights values, $\mathbf{w}^*$. The blue dot represents the weight values $\tilde{\mathbf{w}}$ were the two competing terms of the loss function arrive at an equilibrium, for each regularizer.

2.3.3.B ℓ_1 regularization

An alternative to the ℓ_2 regularization, is the ℓ_1, which is a type of weight decay that is sometimes called *lasso*, in reference to the least absolute shrinkage and selection operator for feature selection [97] used in feature selection [30]. Instead of quadratically penalizing the weights, this regularization usesthe sum of absolute values of the individual weights [11]:

$$\|w\|_1 = \sum_i |w_i|, \tag{2.36}$$

This induces sparsity in the parameters, which can be translated as the optimal solution having some parameters at zero, which greatly reduces the complexity of the model [30]. With this regularization, the loss function is then given by

$$\mathcal{L}_{\text{reg}}(\theta) = \mathcal{L}(\theta) + \alpha\|\mathbf{w}\|_1, \tag{2.37}$$

with the corresponding sub-gradient:

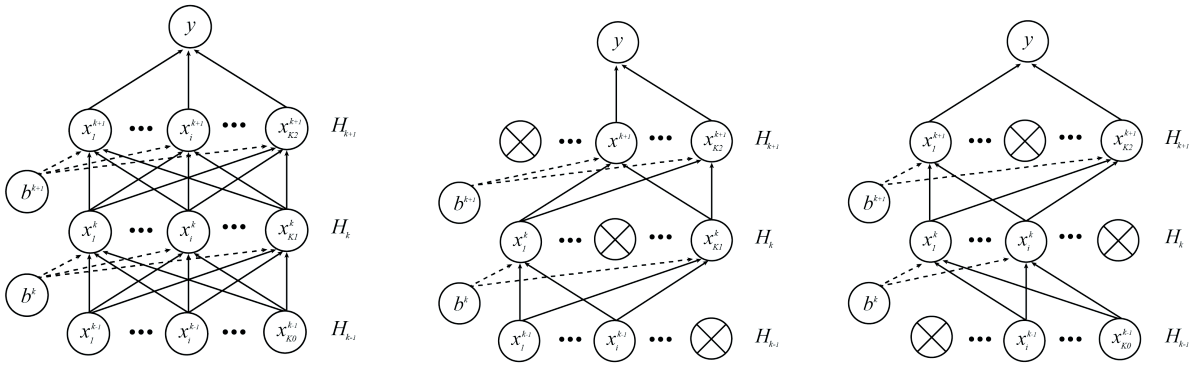
$$\nabla_w \mathcal{L}_{\text{reg}}(\theta) = \nabla_w \mathcal{L}(\theta) + \alpha\,\text{sign}(\mathbf{w}), \tag{2.38}$$

where $\text{sign}(\mathbf{w})$ is simply the element-wise application of the sign to $\mathbf{w}$. A graphical interpretation of this regularization is still possible, as can be seen in Figure 2.19(b), where the previous ball seen in Figure 2.19(a), is now replaced by a region delimited by the ℓ_1 norm.

2.3.4 Dropout

One recent regularization technique extensively used in a wide range of NN architectures is named *dropout*. Dropout works by randomly dropping hidden units, h_i and their incoming ($\mathbf{W}_i^k$) and outgoing ($\mathbf{W}_i^{k+1}$) connections, at each iteration of the training phase [30,91]. The unit's retaining rate, p, is usually called *dropout rate*. After training, when applying the network to the test set, the weights of the network are downscaled by a factor equal to p, which can be seen as a computationally efficient approximation of the average prediction of all the sub-networks [91].

Dropout can be seen as analogous to the training of an ensemble of sub-networks that share parameters, but where each model inherits a separate sub-set of parameters from the original network. A visual representation is presented in Figure 2.20, where two possible sub-networks arise from the application of dropout with a dropout rate of $\frac{1}{3}$ to three successive layers. Alternatively, dropout can be thought of as a sort of regularization to each hidden units that forces each feature to be good in different contexts [30], by limiting the co-adaptation of the units to the training data, dropout prevents overfitting [91], therefore increasing the generalization power of the network as each hidden unit is forced to



(a) Network without dropout **(b)** Sub-network 1 **(c)** Sub-network 2

Figure 2.20: Region proposal mechanism. a) In a sliding window fashion, patches of the feature map are extracted, which are convoluted with k predefined anchors, to generate a fixed sized vector. These are used first by a general FC layer and then fed to two sibling FC layers: a regression and a classification layer. b) Each window from the sliding window will be convoluted with the k rectangular, fixed sized, anchors.

redundantly encode its information [30].

Finally, one of the major advantages of this method is that it works well with nearly any model that uses a distributed representation, while still allowing the learning algorithm to rely on stochastic gradient descent methods (such as Adam) [30], and therefore is well suited for regularization in the models developed in the context of this work.

2.3.5 Data Augmentation

One obvious way to improve the generalization ability of a model is to train it with more data [30]. This is often not possible, as the amount of available data is limited. However, artificial ways to increase the number of examples exist, and these are known as *data augmentation*. In image classification tasks, these methods are reasonably straightforward: a classifier can be improved through data augmentation techniques that exploit its invariance to several transformations [30, 73]. Therefore, it is possible to generate a new set of examples by applying label preserving transformation to the existing $\mathbf{X}$ in the training-set, which include geometric augmentations, such as scaling, translations, and rotations, as well as colour augmentations [30].

When applying these techniques, one important consideration is that the transformations need to generate new examples, without corrupting any significant features in the original ones, as this could result in changes to the class [30].

Due to the inherently limited number of examples in the medical field, and the huge need for examples when training CNNs, data augmentation is extraordinarily useful in this context [41], and will be largely used in this work. One recent augmentation method relied on *generative adversarial networks* (GANs)

[31], a type of generative model based on differentiable generator networks, and which was used to perform unsupervised generation of new images for training, with powerful results in data augmentation for image recognition models [63, 73]. In medical imaging classification, this technique has also allowed for improved performance in comparison to traditional data augmentation techniques [13, 24, 35].

2.3.6 Class Imbalance and Example Sampling

Another issue deeply ingrained in classification tasks is that of data imbalance, which corresponds to the existence of one (or more) classes with a vastly smaller amount of examples, in comparison to others [30]. The main problem is that the imbalanced example distribution will lead to overfitting to the most represented classes [11, 30]. In the context of DL, usually two major approaches to deal with this problem are used: artificially re-balance the data, and/or use of an appropriate loss function that takes into account the imbalance depending on the amount of data imbalance after the re-balancing techniques are applied.

Methods dedicated to data engineering include applying data augmentation techniques only to the minority classes [30], which carry the same limitations of the approaches already seen in Subsection 2.3.5, or sampling more examples from the under-expressed classes, when preparing the training-set. This is usually called *oversampling*, and may lead to good results depending on how unevenly distributed the classes are in the dataset is, and on the number of examples. If the training set is extremely small, reducing the number of examples even further will negatively impact the performance of the model, yet in cases where this is not the situation, it might lead to better results [39]. Usually, the loss function for image classification tasks is the softmax (Eq. 2.17). However, it is possible to reshape the loss function to down-weight the loss assigned to well-classified examples, which corresponds to the so-called focal loss [58]. One of the major reported issues with the focal loss is its recall, which is is one of the most relevant qualities in medical applications. This issue has recently been explored in an application for satellite images, but no work has been developed specifically for medical imaging [86].

2.3.7 Architectural Changes for Rotation Invariance

As already explained, invariance to rotation and shifts is extremely important to avoid overfitting. If the model isn't able to generalize to objects in different poses and positions, it will underperform when shown the same object rotated or shifted from the position it was in the examples of the training set [30]. As previously presented in Section 2.2.3, by incorporating a pooling layer in CNNs and, more specifically, by using a max-pooling layer, it is possible to grant the network some invariance to small shifts and rotations [30, 47]. Another common way to limit the overfitting of the network is to use data augmentation to the training set with specific augmentation techniques that focus in delivering new examples that

simulate objects in different positions, such as random rotations, mirroring, and shifts [41, 73].

Another recent architectural change to the models, is the use of capsule structures [84, 85], which allow data representation in a CNN to take into account important spatial hierarchies, and that have had an impact in increasing models' generalization ability [84, 104]. These are based on the concept of inverse graphics that brings one key idea: representation of objects should not depend on view angle [84]. There are already several successful applications of these architectures in the medical imaging domain [3, 48], and its impact on performance is only significantly noticeable for a training-set with a lower amount and imbalanced set of examples [43]. Yet, there are still several limitations when bringing these models to practice as current programming frameworks have not yet been optimized for this type of networks [25].

2.3.8 Transfer Learning

Outside the field of medical imaging, it is possible to have datasets comprising millions of images. These enable the use of models with extremely high capacity with limited risk of overfitting [30]. One recent technique enabled the use of these large models by taking pre-trained weights of a neural net trained on some similar and more comprehensive data, and fine-tune certain parameters to best solve a more specific problem [52]. To do so, only the last layers of the model is re-trained in the application-specific training set, effectively re-learning the weights of the classifier sub-network and/or some previous convolutional layers, and leaving all other "frozen" [30, 59].

This allows networks extensively trained in extraordinarily large datasets, such as the *ImageNet* dataset - comprised of more than 1.2 million images from 1000 different categories, to be used in problems where such data availability would be impossible. Yet, for large scale medical tasks, little benefit when using transfer learning has been shown, and simple lightweight models have a comparative performance to models trained using ImageNet [75]. Nonetheless, this technique still offers a viable application where larger datasets are impossible to attain.

3

Data and Tools

Contents

3.1 Used Data

3.1.1 The LIDC-IDRI Dataset

The Lung Image Database Consortium - Image Database Resource Initiative (LIDC-IDRI) is a public and free dataset that contains 1018 thoracic CT scans, both standard and low-dose CT scans, all annotated by four experienced radiologists in a two-phase blind annotation process. [8]. Seven academic centers and eight medical imaging companies collaborated to identify and ultimately resolve all the inherent technical, organizational, and clinical issues of building such a large dataset. The dataset has a total of 2669 lesions that were marked as a nodule with a diameter larger than 3 mm, by at least one radiologist, from which 34.7% (928 lesions) were marked by all four radiologists.

To increase the quality of the examples, all scans with slice thickness greater than 3 mm, inconsistent slice spacing, or missing slices, were excluded, therefore yielding 888 scans. Taking into account recent metrics for the follow-up of detected pulmonary lesions, only nodules larger than 6 mm were included [9]. The final distribution of the nodule size is shown in Figure 3.1, where the diameter of the lesions ranges from 6 mm to 32 mm with a large proportion os smaller nodules. This dataset was the one used to train and evaluate the detection model of the hybrid DL architecture by dividing it into three parts: 80% of the scans constitute the training set and the remaining were equally split into validation and test sets. From each scan, a variable number of 64×64 patches was extracted for training and validation, maintaining a balanced distribution of classes, and totalling 1908 patches from 704 scans for the training set, and 344 patches from 88 scans for the validation set. Patches for the test set were sampled following a sliding-window approach from a total of 88 scans, with each CT scan producing 1156 patches per slice with a highly unbalanced class distribution.

3.1.2 Decathlon Competition Dataset

The medical Decathlon competition was a recent effort to develop a machine learning model that could generalize to segmentation tasks from a wide range of clinically relevant anatomies. This led to the development of a large collection of annotation medical datasets with high-quality imaging data that was released as a free open-source platform. The lung task of the Decathlon competition encompassed the segmentation of lung tumours from thin-section CT scans. The dataset contains 64 labelled CT scans with singular non-small lesions annotated by an expert thoracic radiologist on a representative cross-section [90]. In the context of this work, this was the dataset used to train and evaluate the segmentation model in the hybrid DL architecture. As the segmentation of pulmonary lesions is relevant in the context of the radiomics pipeline only for larger ones [28], tumours smaller than 25mm were excluded. The final dataset totalled 32 scans, and a variable number of 64$\times$64 patches was extracted from the VOI around each lesion. Each VOI had the same number of slices with and without lesion, the latter being equally

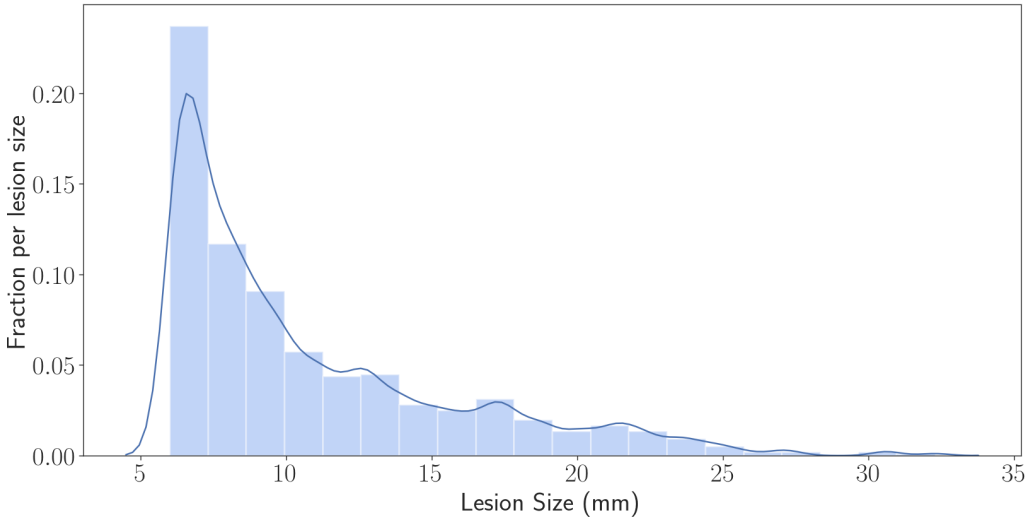


Figure 3.1: Final size distribution of the lesions in the training set of the detection network.

distributed above and below it. Finally, 5-fold cross-validation (CV) was used, with 21 scans used to train the model, and the rest split by assigning 6 examples to the validation set and 7 to the test set.

3.2 Image Pre-processing

In machine learning, one of the most important steps for high-performance models, is to implement a series of steps that improve the quality of the data. The steps adopted in the processing methodology are presented next. These are partially outlined in Figure 3.2.

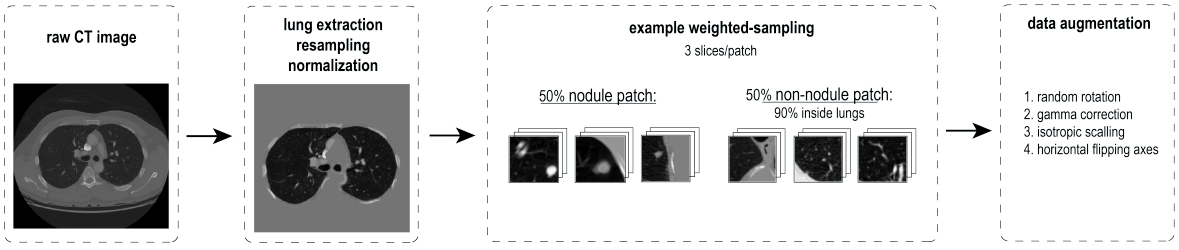


Figure 3.2: Depiction of the preprocessing and data curation steps used to prepare the training set for the detection model. The raw thoracic CT image first preprocessed by having the lungs extracted, followed by resolution resampling to the median of the training-set voxel size and histogram-based normalization. Sets of 3 patches were extracted, maintaining a balanced class distribution and focusing on highly informative patches. The training was optimized by using several data augmentation techniques.

3.2.1 Lungs Extraction

One of the main issues in the automatic detection of lung lesions is the number of false positives outside of the lung region. To avoid this issue, using the lung masks provided in the *LIDC-IDRI* dataset, the lungs were extracted from all images. A small dilation was also applied to the masks beforehand to allow the accurate detection of *juxta-pleural nodules* – nodules attached to the side-wall of the lung. Due to the hybrid approach to the detection and segmentation tasks, which will be further described in Section 4.4), only the CT scans used for lesion detection will have the lungs removed. This effort also serves the purpose of reducing the amount of information that needs to be encoded by the model, as all the regions outside of the lung will be easily detected as not containing any lesion.
As the accuracy of the lung segmentation is not a factor for the deep learning model performance (as long as within certain quality criteria that guarantees all the lung region is included in the mask), the lung extraction could have been performed by other non-supervised methods such as [5, 67].

3.2.2 Resampling

In general, DCNNS are not prepared to handle the voxel spacing heterogeneity of CT images, which arises from different scanners and different acquisition protocols. To overcome this issue, resampling was applied to all scans by first estimating a common image resolution through a median of the dataset voxel size, and then performing a second-order B-spline interpolation to this size. The resampling is reported to improve the system's performance by close to 20% in other 2.5D CNN based systems [7].

3.2.3 Normalization

Finally, as previously stated in Section 2.1.3, the CT scans have an absolute intensity scale arising from the estimation of the attenuation coefficients. This yields a scale that ranges from -1000 to 1000, and which may introduce instability when training a DCNN, as it usually performs better with values in a lower range [51]. Therefore, the final input images for both networks were volume-level normalized by using a histogram-based normalization, following a clipping to the $[1, 99]$ intensity value percentiles.

3.3 Data Curation

Due to the limited amount of training data in both tasks, extensive data curation of training examples was performed. This also consisted of implementing extensive data augmentation techniques to increase the amount of data available in the training-set.

3.3.1 Example Resampling

As explained in Section 2.3.6, in the case of imbalanced datasets, one of the first attempts should be to artificially balance the data through oversampling strategies. Therefore, weighted sampling of significant examples was applied, with a balanced sampling of examples for training the detection network: equal numbers of patches with and without nodules were extracted from each scan. Due to the extraction of the lungs from the image, learning the false examples from outside the lungs is a trivial task for the network, so 90% of the negative examples were chosen from within the lungs. A similar strategy was applied to train the segmentation network with examples selected through weighted sampling where the sampling likelihood of each voxel, and window around it, is proportional to its class frequency.

3.3.2 Data Augmentation

Finally, several data augmentation schemes were applied to both the final 1908 patches that constituted the training-set of the detection model, and to the 24 VOI that comprised the training-set for the segmentation model. Several extensions of the transformations to 3D input, as well as specific data augmentation strategies for medical imaging, which try to capture and replicate relevant data statistics, were used.

Nonetheless, the data augmentation methodology used in this work was defined with a focus on techniques that retain relevant features for the tasks at hand, specifically anisotropic deformations that could distort texture features were avoided, such as shearing and elastic deformations, as well as any techniques involving the addition of noise. Common augmentation techniques in this context include [41]:

- *flipping* the image with respect to its axis;
- *rotations* around one of the three dimensional axis;
- *scaling* by an affine transformation $\mathbf{AX}$

$$\mathbf{A} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix}, \tag{3.1}$$

  and s_x,s_y, and s_z are scaling factors.

- *mirroring*, by flipping with respect to a specific axis..

Both the detection and the segmentation datasets were augmented using random isotropic scaling, and horizontal mirroring; however, random rotations and vertical mirroring was only used for the segmentation, as orientation may encode relevant features for the detection task.

3.4 Implementation and Hardware Details

Several programming frameworks are currently available for the efficient implementation of DL architectures, with the majority of them being developed either with a *TensorFlow* [2] or a *PyTorch* backend [71]. In this work, *NiftyNet* was used. This is a TensorFlow-based platform, which provides a modular DL pipeline with components dedicated to data loading, data augmentation, network architectures, loss functions, and evaluation metrics, all tailored for medical imaging tasks [27]. Some of the advantages of using the *NiftyNet* infrastructure include its ability to rapidly develop and test DL solutions for segmentation and classification within the context of medical imaging, as well as the existence of several models already implemented and ready for use. Finally, experiments were run on a machine with an *Intel Xeon-1620 CPU* and two *NVidia Geforce GTX 1080* GPUs (totalling 22 GB on-board memory), which enabled the models to be trained with reasonably high batch size, whilst maintaining reasonable training times.

4

DL Network Architectures

Contents

4.1 Components of the Architecture

In this work, two architectures were developed: one for the detection task, and another for the segmentation task. The main goal of the work was to produce high-performance models, while still focusing on reducing their final capacity, and thus enabling them with good chances of not overfitting. During the process of developing each model, two notable additions were made: the inclusion o *residual connections*, and the use of *leaky-ReLUs* as activation functions; both components will be addressed in this section.

4.1.1 Residual Connections

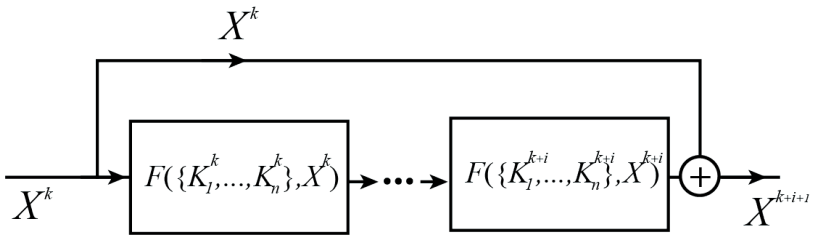


Figure 4.1: Representation of a residual block, where the input X^k is depicted skipping a set of i layers and added to layer $k+1$.

Residual connections, also called skip-connections, were developed for refined training of deep networks, and, even though they do not directly tackle the problem of overfitting, by enabling the reduction of the size of the architectures it allows the reduction of the model's capacity [36]. The key idea of a skip-connection is to define an identity mapping to bypass one or more parameterised layers in a network, through the addition of the input of a residual block to its output (Fig. 4.1) [36, 57]. Specifically, let a convolutional layer H_k with a set of kernels $\{K_1^k, .., K_n^k\}$ and an input X^k, as defined in Equation 2.11, be denoted by $X^{k+1} = F(\{K_1^k, ..., K_n^k\}, X^{k-1})$. Then, the output of a residual block with i layers can be described as

$$X^{k+i+1} = X^k + F(\{K_1^{k+i}, ..., K_n^{k+i}\}, F(...(F(\{K_1^k, ..., K_n^k\}, X^k))...). \tag{4.1}$$

Another major advantage of this technique is that it helps to deal with the vanishing gradient problem, an existing difficulty in gradient-based learning. This problem occurs when the gradient becomes extremely smaller, preventing the weights from being effectively updated, and in the worst case, it might lead to completely stopping the training [30, 37]. The skip-connection between layers reduces the length of the backward paths, from the output of each layer, thus mitigating gradient problems [92]. It was

also shown that using residual connections in a CNN smooths the loss landscape, and produces loss functions that are easier to train [56].

Recently, it has been found that residual networks operate similarly to an ensemble of relatively shallow networks [100]. Models without residual connections will have, for each single unit x_i^k, a fixed number of input units in X that affect it - the commonly named *receptive field* of an unit [30]. However, training a network with n residual blocks will use 2^n different paths, and therefore features can be learned by a varied set of different receptive fields. Experiments in medical imaging tasks suggest that residual networks reduce the effective receptive field of the network [57]. These also indicate that it is possible to reduce the number of parameters by one order of magnitude, while comparing favourably against other models.

4.1.2 Activation Function

As mentioned in Section 2.2.2, any neural network first relies on the use of a activation function σ, as presented in Equation 2.14 for the non-linear characteristics of the model. The most commonly used activation function is the *rectifier linear unit* (ReLU) [52, 76], which has already been shown to improve training comparatively to other previously widely used activation functions [29], such as the *logistic sigmoid* and the *hyperbolic tangent*. The *ReLU* activation function is simply defined as

$$\sigma(z) = \text{ReLu}(z) = \max(0, z) \tag{4.2}$$

One important advantage of this activation function is that it has an extremely computationally efficient derivative, which allows it to be calculated much more easily than other activation functions. Recently, several variants have arised, such as the *noisy-ReLU*, the parametric *ReLU*, and the activation function that was mostly used in this work: the *leaky-ReLU*. This specific activation function allows a small, positive gradient when the unit is not active and was proven to avoid the vanishing gradient problem [62, 106]. The *leaky-ReLU* is defined as

$$\sigma(z) = \text{leaky-ReLu}(z) = \begin{cases} z & \text{if } z > 0 \\ \alpha z & \text{otherwise.} \end{cases} \tag{4.3}$$

where $\alpha > 0$ is a small constant, typical values are $\alpha \simeq 0.01$. Throughout all this work, the leaky-ReLU was used in all layers of the detection architectures, except in the last layer, where a softmax is always used.

4.2 Detection Network

4.2.1 Architectures

In the context of building this network an effort was made to minimize the complexity of the model. This attempt was loosely guided by the principle of parsimonony, Occam's razor, which states:

> *"Among competing hypotheses that explain known observations equally well, one should choose the simplest."*

For the problem described in this work, this means that when taking into account the problem of finding a function $F(X, \theta)$ that maps $\mathbf{X}$ to the set of nodules centroid coordinates, and bounding-box sizes,D, the goal should be to choose the simplest model F that achieves a feasible performance within the context of the clinical problem. This premise is also well adjusted within the framework that was described in Section 2.3.2.C, where in order to minimize the overfitting of a model, its capacity should be reduced.

In this chapter, the general approach for nodule detection will be described, which will be followed by the details on the architectures and training schemes that were tested for this task.

4.2.1.A Rewriting detection as a patch classification problem

To more effectively tackle this problem, the lesion detection model that was first described in Equation 2.2 was written as a patch classification problem. This meant first transforming the input CT scan X into a set of patches $P = \{p_1, ..., p_n\}$, where $p_i \in \mathbb{R}^{N \times N \times Z}$ is a two-dimensional or three-dimensional patch, depending on Z being equal or larger than 1. In this work, N was set to 64, to include the larger sized nodule in the dataset, and at the same time to be able to deal with input size restrictions of the U-Net architecture. The problem could then be rephrased as one of binary-classification, where each patch is classified as having, or not, a lesion. Let us then redefine the detection model as a function F_1 that maps from a single patch p_i to its corresponding classification c_i:

$$F_1 : \mathbb{R}^{N \times N \times Z} \longrightarrow \{0, 1\}. \tag{4.4}$$

Through this reformulation of the problem, it is possible to restrict the lesion detection task to a much simpler binary classification problem, instead of having to determine a set of values $\{y_1, y_2, y_3, y_4, y_5, y_6\}$ for each lesion that is required to be detected. The integration of this detection model into the full lesion detection and segmentation model will be made clear in Section 4.4.

Following this definition of F_1, the goal will then be to approximate this function with a DL model by learning a set of weights and bias $\theta = \{W^k, b^k\}$ as previously specified in Section 2.2.7, for all layers in the architecture. In this work, several architectures were tested, with a focus on using a small set

of parameters. To guarantee the validity of the each added element of the network, several tests were performed starting from the simplest fully connected model, and finalizing with a model with several state-of-the-art components that was able to achieve competitive performance.

4.2.1.B NaiveNet

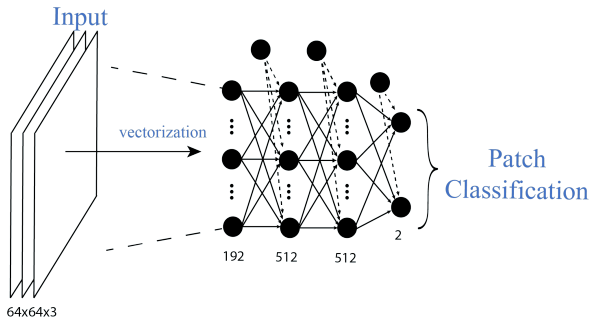


Figure 4.2: Depiction of the *NaiveNet* architecture. The layout follows the previously shown iconographic representation of FC layers.

Based on the first iterations of NN architectures that were applied to image classification tasks [53], the earliest sketch of the architectures that were tested amounted to a simple multilayer perceptron. This architecture, which we called *NaiveNet*, is depicted in Figure 4.2. The focus of this first approach with an architecture containing only FC layers, was to establish a lower bound on what was possible to achieve in this problem using NN. Several numbers of hidden units and layers were tested; the final NN was composed of two FC layers, each with 512 hidden units, and using a *leaky-ReLU* as the activation function, with $\alpha = 10^{-2}$. The last set of hidden units was connected to two outputs, with a binary-softmax as the activation function. All details regarding the *NaiveNet* architecture are described in Table 4.1.

Table 4.1: Architecture details of the *NaiveNet*.

Layer	**FC layers**	
	number of units	activation function
FC 1	512	leaky-ReLU
FC 2	512	softmax

4.2.1.C NaiveConvNet

The next NN architecture that was tested, besides the FC layers that were present in the *NaiveNet*, also included convolutional layers, which loosely followed the description of the first layers of the VGG-16 [89]. The final architecture was empirically obtained by testing various NN variations to optimize the number of parameters of both the FC layers and the convolutional layers, as well as the number

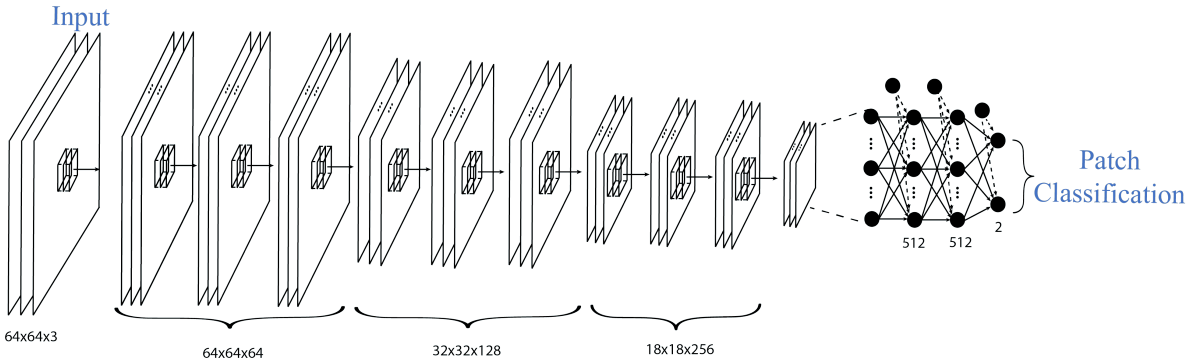


Figure 4.3: Scheme of the *NaiveConvNet* architecture. In this case, the input was assumed to a three slice patch. The FC and convolutional layers follow the pictorial representation introduced in 2.

of layers and kernels in the convolutional layers. This architecture was named *NaiveConvNet* and it has three convolutional layer blocks that extract features from the data and two fully connected layers that perform classification, as shown in Figure 4.3. All the convolutional layers used a 3×3 kernel, which were grouped into sets of 3, before the application of a max-pooling. Convolutions and pooling are all applied with appropriate padding, with convolutions having stride 1, and max-pooling using 2x2 kernels and stride 2. The FC layers used followed the same specifications of the *NaiveNet*, and the full description of the network is indicated in Table 4.2.

Table 4.2: Architecture details of the *NaiveConvNet* network divided into *convolutional* layers and *fully connected* layers.

Layer	**convolutional layers**		
	kernel size	number of kernels	stride
Conv 1	3×3	64	1
Conv 2	3×3	64	1
Conv 3	3×3	64	1
Max Pooling 1	2×2	-	2
Conv 4	3×3	128	1
Conv 5	3×3	128	1
Conv 6	3×3	128	1
Max Pooling 2	2×2	-	2
Conv 7	3×3	256	1
Conv 8	3×3	256	1
Conv 9	3×3	256	1
Max Pooling 3	2×2	-	2
	fully connected layers		
	number of units	activation function	
FC 1	512	leaky-ReLU	
FC 2	512	softmax	

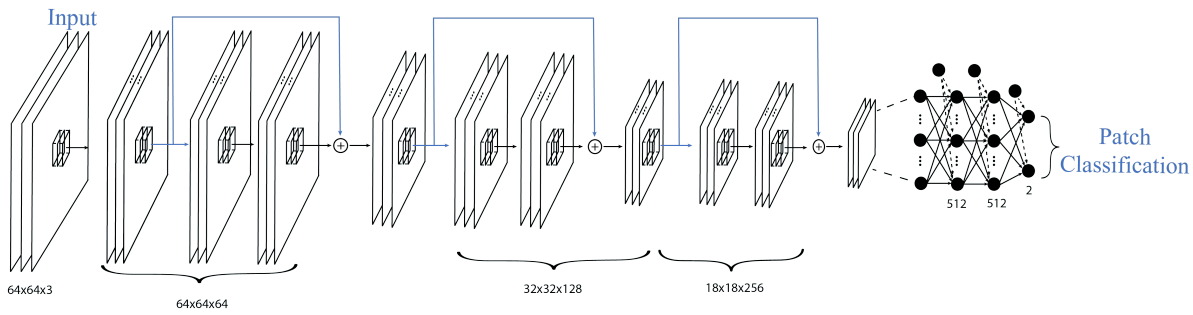


Figure 4.4: Depiction of the *DetectionNet*, which constitutes a novel *residual DCNN* model developed for the detection of lung lesions in CT images. The input was assumed to be a three slice patch. All component's representations follow the sketchs introduced in Section 2.

4.2.1.D Residual CNN

Finally, the last iteration of the NN was completely based in the architecture of the *NaiveConvNet*, but also included residual connections, following the description in Section 4.1.1. This addition was implemented with a skip connection being completed between the first and the third layer of each block of three convolutional layers, and preceding the max-pooling. All other details of the architecture follow the description regarding the *NaiveConvNet*, including the number of paremeters and kernels, as well as its strides. These again are provided in Table 4.2. This deep CNN constitutes a novel *residual DCNN* model for the detection of lung lesions. Its architecture is shown in Figure 4.4,

4.2.2 Training

4.2.2.A Loss Function

The loss function chosen was used to simultaneously train all the NN that were presented above. In classification tasks, the most commonly used loss function is the cross-entropy [30], which is only outperformed by the focal loss [58], when the dataset at hand is heavily unbalanced. As in this work several strategies were implemented to deal with unbalanced distribution of the examples (Sec. 2.3.6), such as oversampling and data augmentation, there was no need to implement the focal loss. The loss function is then defined as

$$\mathcal{L}(\theta) = \frac{1}{|B|} \sum_{i \in B} L_{\text{ce}}(\hat{c}_i, c_i), \tag{4.5}$$

where B is the training set batch, and the cross-entropy loss function, L_{ce}, for binary classification, is given by

$$L_{\text{ce}}(\hat{c}_i, c_i) = -(c_i \log(\hat{c}_i) + (1 - c_i) \log(1 - \hat{c}_i)), \tag{4.6}$$

and $\hat{c}_i$ is the estimated classification probability from the binary-softmax output, given the 64x64xC patch, and $c_i \in \{0, 1\}$ is the corresponding label (lesion or no lesion) on the middle slice of the patch.

The next step is to add parameter regularization terms to the loss function, following the description in Section 2.3.3, to deal with overfitting. Therefore, the loss function is the rewritten as

$$\mathcal{L}(\theta) = \frac{1}{|B|} \sum_{i=1}^{B} L_{\text{ce}}(\hat{c}_i, c_i) + \alpha_1 \|\mathbf{w}\|_1 + \alpha_2 \|\mathbf{w}\|_2, \tag{4.7}$$

where w is a subvector of θ that contains the weights, but not the biases, the first added term is the ℓ_1-norm introduced in Section 2.3.3.B, and α_1 is its weight, whereas the second term is the ℓ_2-norm introduced in Section2.3.3.A, and α_2 is its weight.

4.2.2.B Optimization

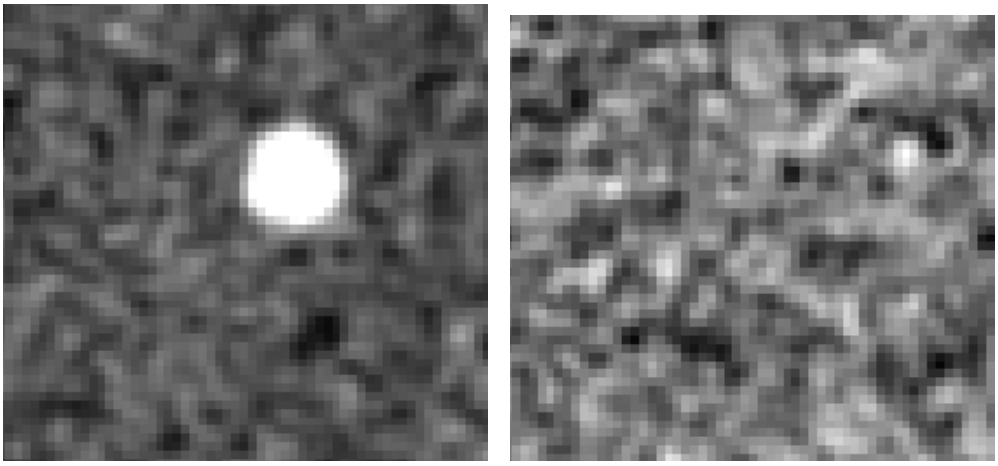
The optimization of all hyperparameters was performed empirically, with the learning rate λ defined as 10^{-5} for the *NaiveNet* and as 3×10^{-4} for the other two models. In order to balance the effect of increasing the generalization of the model when the batch size is large, and the overfitting that occurs when this number gets to high [30], B was set to 300. Finally, extensive regularization was implemented through penalization terms added to the loss function, taking into account the specifications of each model. For the *NaiveNet* only ℓ_2-regularization was applied, with the $\alpha\ell_1$ set to zero. Regarding the other two networks, due to their larger size, the parameter penalization strategy was defined in order to benefit from both terms. In the first 500 iterations, α_1 was set to 0.1, whilst α_2 was set to zero, in order to select the weights that contribute the most to the improvement in performance, and attempting to leave the others as zero. After this, α_1 was set to zero, and α_2 was set to 0.1. These non-conservative values were chosen in order to compensate for the inherent high tendency of the models to overfit in this small dataset.

4.2.3 Final Model Choice

In order to evaluate each one of the models, an experiment in a synthetic dataset was performed. The main goal was to assess the performance and stability of each model, in a much more benign, and controlled environment.

4.2.3.A Synthetic dataset

When developing the synthetic dataset, special attention was given to limit the variability of the images, while still striving the keep the noisy nature of thoracic CT images. Therefore, any representation of possible confounding objects, such as vascular structures, was rejected. This lead to only one clearly



(a) Patch with positive class **(b)** Patch with negative class

Figure 4.5: Example of synthetic patches used to evaluate the three types of detection architectures.

detectable object per positive patch. The lesions were simulated through elliptical shapes. In order to provide a similar object size distribution to the one present in the LIDC-IDRI dataset (Fig. 3.1), the distribution of the maximum diameter in each ellipse of the synthetic dataset was set following a gamma distribution, with the shape parameter, and the scale parameter, respectively set to 3 and 2. Each patch was then corrupted with Gaussian noise, followed by the application of a Gaussian kernel to reduce the edge contrast, as commonly seen in CT images. The dataset was composed of a total of 5300 patches. Note, that the full dataset was designed following a balanced distribution of classes, where the number of positive and negative examples was the same. Examples of synthetic images are shown in Figure 4.5, one for each class.

4.2.3.B Tests Performed

As there was no hyperparameter tuning during this experiment, the dataset was split into a training set, composed of 80% of dataset, and a test set. Each architecture was then trained 100 times for 1000 iterations, yielding, from each trial, four models taken from iterations 100, 200, 500 and 1000. These were then evaluated in both the training set and the test set. The results for this experiment are summarized in Appendix C, in Figure C.1. As expected, the performance of the Naive-Net is already extremely high, with values of accuracy above 0.90 for models from the iteration 100. The addition of convolutional layers in the NaiveConv-Net leads to some improvements in performance, which are then just marginally surpassed by the residual DCNN. Nonetheless, the final architecture used in the LIDC-IDRI dataset was the residual DCNN.

4.3 Segmentation Network

4.3.1 Architectures

The main approach relied in the previously shown *U-Net* (Section 2.2.6.B), with a new version focusing on reducing the number of parameters. Due to the recent successes in the field, relying in models using residual connections, besides the *vanilla U-Net*, a version with residual blocks was also tested [21].

4.3.1.A U-Net

In order to adapt the original *U-Net* architecture to the segmentation of 64×64 patches, its size was reduced. This change aimed at limiting the effective maximum capacity of the model, and, therefore, lessen overfitting. Incidentally, the number of levels in both the descending and ascending paths of the network was scaled down from four blocks to three blocks. Also, in order to better tackle the vanishing gradient problem, a *leaky-ReLU* with a slope of 10^{-2} was defined as the activation function for all the layers except for the last one where softmax was used. Following the original description, 3×3 convolutional kernels were implemented with a stride of 1, as well as 2×2 deconvolutional kernels with a stride of 2. Each pooling layer was set with a 2×2 kernel with a stride of 2. Similarly, the number of channels per layer also followed the original work. Finally, convolutions, deconvolutions and pooling are all applied with appropriate padding. The architecture details can be seen in Table 4.3.

As stated, a *residual U-Net* was also tested. In this case, a skip connection was added in each block of the encoding and decoding path, between the first and the second layer. It is important to notice that this model didn't have any other changes in comparison to the vanilla U-Net architecture. The final *residual U-Net* architecture is depicted in Figure 4.6.

4.3.2 Training

4.3.2.A Loss Function

The *dice-score*, also called Sørensen-Dice coefficient, or F-1 score, is a common metric that is used to evaluate segmentations in the field of medical imaging [110] by evaluating the similarity of the ground-truth segmentation and the estimated segmentation. It is given by the expression

$$\text{dice-score} = \frac{2|\mathbf{S} \cdot \hat{\mathbf{S}}|}{|\mathbf{S}|^2 + |\hat{\mathbf{S}}|^2}, \tag{4.8}$$

where $\mathbf{S}$ is the ground-truth segmentation mask, $\hat{\mathbf{S}}$ is the estimation that is obtained from the model at the output of the binary-softmax. Several variations of loss functions based on the *dice-score* have been recently proposed [21, 42, 93]. The basis for the loss function that was used in this work was the

Table 4.3: Architecture details of the detection network

Layer	kernel size	activation function	number of kernels	stride
		Contracting path		
Conv 1	3×3	leaky-ReLU	64	1
Conv 2	3×3	leaky-ReLU	64	1
Max Pooling 1	2×2	-	-	2
Conv 4	3×3	leaky-ReLU	128	1
Conv 5	3×3	leaky-ReLU	128	1
Max Pooling 2	2×2	-	-	2
Conv 6	3×3	leaky-ReLU	256	1
Conv 7	3×3	leaky-ReLU	256	1
Max Pooling 3	2×2	-	-	2
		Bridge		
Conv 7	3×3	leaky-ReLU	512	1
Conv 8	3×3	leaky-ReLU	512	1
		Expanding path		
Deconvolution 1	2×2	-	-	2
Conv 9	3×3	leaky-ReLU	256	1
Conv 10	3×3	leaky-ReLU	256	1
Deconvolution 2	2×2	-	-	2
Conv 11	3×3	leaky-ReLU	128	1
Conv 12	3×3	leaky-ReLU	128	1
Deconvolution 3	2×2	-	-	2
Conv 11	3×3	leaky-ReLU	64	1
Conv 12	3×3	leaky-ReLU	64	1
Conv 13	1×1	softmax	2	1

dice plus cross-entropy loss function that previously achieved state-of-the-art results in the Decathlon segmentation competition [42],

$$\mathcal{L}(\theta) = \frac{1}{|B|}\sum_{i=1}^{|B|} L_{\text{dice}}(\hat{\mathbf{S}}^i, \mathbf{S}^i) + \frac{1}{|B|}\frac{1}{m}\sum_{i=1}^{|B|}\sum_{j=1}^{m} L_{\text{ce}}(\hat{s}_j^i, s_j^i), \tag{4.9}$$

where s_i^j is the j-th voxel of the i-th ground segmentation mask, $\mathbf{S}^i$, in the training batch, L_{ce} is the *cross-entropy* loss function previously defined in Equation 4.6, L_{dice} is the *dice* loss function, defined as

$$L_{\text{dice}}(\hat{\mathbf{S}}^i, \mathbf{S}^i) = -\frac{\sum_{j=1}^{m} \hat{s}_i^j s_i^j + \epsilon}{\sum_{j=1}^{m} \hat{s}_i^j + \sum_{j=1}^{m} s_i^j + \epsilon}, \tag{4.10}$$

where ϵ is an added term to avoid underflow. The definition of the *dice* loss function can be extended to include all elements of the batch, which is equivalent to considering the samples in the batch as a pseudo-volume, avoiding the need to average the *dice* loss over the batch. This leads to writing Equation 4.9 as

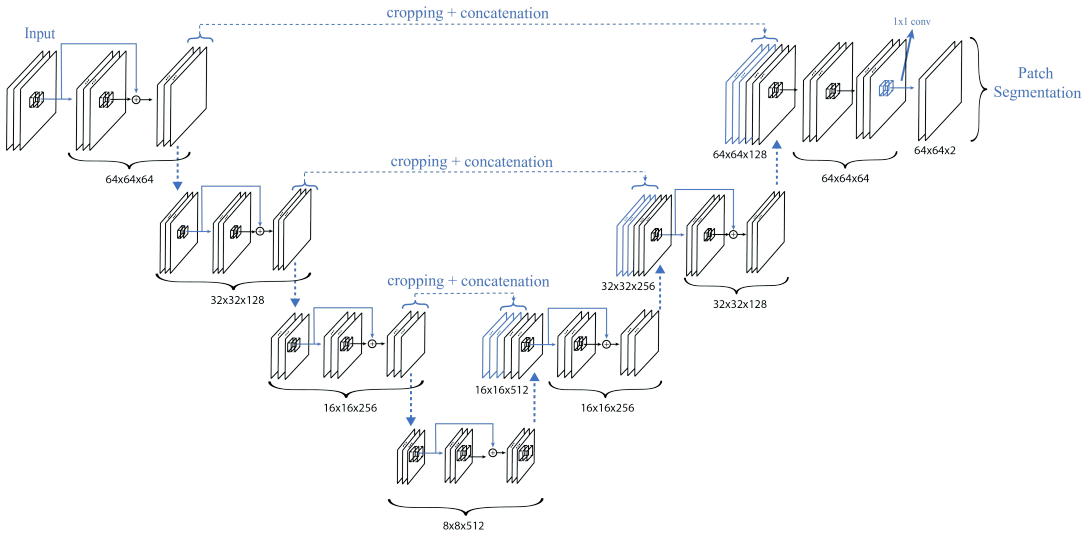


Figure 4.6: Schematic depiction of the final *residual U-Net* architecture that was used to build the *segmentation model*. All notation and illustrative representations of components was already presented in Chapter 2. Note that changes to the original *U-Net* model include the reduced number of layers in both the encoding and the decoding path, as well as the inclusion of the residual connections in each convolutional block.

$$\mathcal{L}(\theta) = \frac{1}{|B|}\frac{1}{m}\sum_{i=1}^{|B|}\sum_{j=1}^{m} L_{\text{ce}}(\hat{s}_j^i, s_j^i) + L_{\text{dice}}(\hat{V}, V), \tag{4.11}$$

where $\hat{V}$ is the set of voxels that are estimated by applying the network to each Z^i, and V is the corresponding set of voxels from the ground-truth segmentation. This forces the *dice* loss function to be re-written as:

$$L_{\text{dice}}(\hat{V}, V) = -\frac{\sum_{\hat{v}_i \in \hat{V}, v_i \in V} \hat{v}_i v_i + \epsilon}{\sum_{\hat{v}_i \in \hat{V}} \hat{v}_i + \sum_{v_i \in V} v_i + \epsilon}. \tag{4.12}$$

As the U-Net usually struggles to segment objects occupying small fractions of the total number of voxels/pixels of the input image, as is usually the case for the the target segmentation of this task: lung lesions. To overcome this hurdle, a novel loss-function was introduced that weights in differently the contribution of the two classes. Such loss-function was named *weigthed dice* (w-dice) and changes Equation 4.12 to

$$L_{\text{w-dice}}(\hat{V}, V) = -\sum_{k \in \{0,1\}} w_k \frac{\sum_{\hat{v}_i \in \hat{V}_k, v_i \in V_k} \hat{v}_i v_i + \epsilon}{\sum_{\hat{v}_i \in \hat{V}_k} \hat{v}_i + \sum_{v_i \in V_k} v_i + \epsilon}, \tag{4.13}$$

where the set of voxels $\hat{V}$ and V were each separated into two subsets, with each k respective subset $\hat{V}_k$ and V_k corresponding to one class; the weights w_0 and w_1 control the contribution of each class to the loss, with $w_0 < w_1$, thus down-weighting the background contribution. In this work, w_0 was set 0.4 and w_1 was set to 0.6.

Similarly to the case of the detection network, a parameter penalization term is added to the loss function. In this case, only ℓ_2-regularization was used. The final expression for the loss function is then

$$\mathcal{L}(\theta) = L_{\text{w-dice}}(\hat{V}, V) + \frac{1}{|B|}\frac{1}{m}\sum_{i=1}^{|B|}\sum_{j=1}^{m} L_{\text{ce}}(\hat{s}_j^i, s_j^i) + \alpha_2\|\mathbf{w}\|_2, \tag{4.14}$$

where α_2 is the weight of the regularization term.

4.3.2.B Optimization

A large number of tests was performed to empirically determine the hyperparameters. This included first optimizing the learning rate, which was set to 10^{-3} for the *U-Net*-based models. Due to the high memory requirements of these two networks, the number of examples in the training batch was extremely limited. In this case, the maximum batch-size, was set to 12, maximizing the memory usage. Finally, a slight amount of parameter penalization was applied, with α_2 being set to 10^{-4}, as a noticeable decay in performance was found for higher values of regularization.

4.4 LungSD-Net

In order to tackle recent limitations in current lung lesion and detection models, an hybrid detection and segmentation network is proposed, following a novel integrated approach, where the detection of lung lesions is connected to its segmentation. The network is composed by the detection network and the segmentation network, which were, respectively, presented in Section 2.2.5 and Section 2.2.6. The details of this new design will be shown next.

4.4.1 Sliding Window Approach and Final Model

As previously stated, for the detection model, the approach used was based on first defining a set of patches, $P = \{p_1, ..., p_n\}$, from the original input scan X, which leads to both a simpler task and a simpler model. The *detection model* can then be incorporated into a sliding-window approach, meaning that each patch p_i will be sampled from X by using a window that will run through each full slice of the image, with a fixed step size, s, when a new CT scan is being inferred. Similarly, the *segmentation model* is integrated into this final architecture by taking as an input each patch that is positively classified

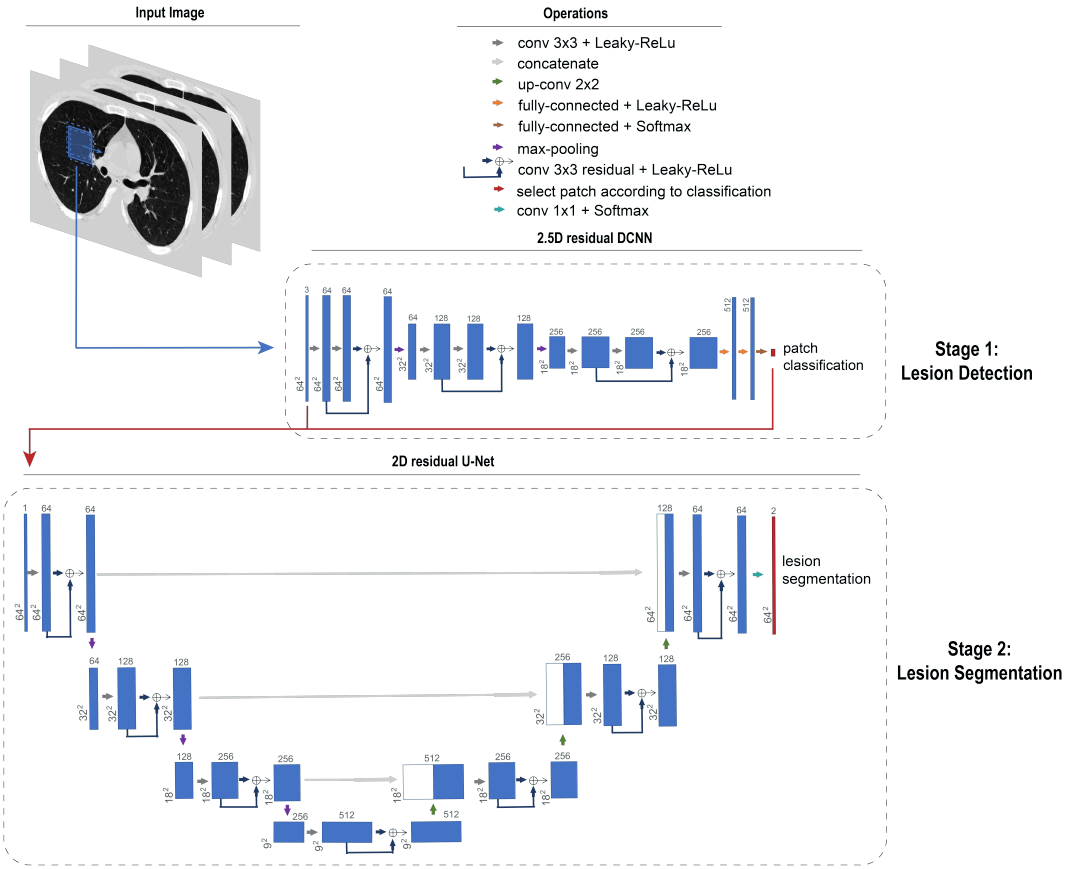


Figure 4.7: Overview of the *LungSD-Net* architecture, including a representation of the sliding-window approach. One 64×64×3 patch was extracted with a step of 15 voxels, and is used as input for the detection model, the 2.5D *residual DCNN*, that classifies the middle slice as having or not a lesion. The patches positively classified are used as input for the second step of the model, the 2D *residual U-Net*, that yields a final segmentation

as having a lesion. This model will be called *LungSD-Net*, as it performs the segmentation of lesions subsequently to its detection. A depiction of the model can be seen in Figure 4.7. In this work, a 64×64×3 sliding-window was defined with s set to 15, to guarantee that each lesion would not be sliced in any of the steps.

5

Results and Discussion

Contents

5.1 Performed Tests and Evaluation Metrics

In order to assess the proposed LungSD-Net, separate evaluations were made for each of its sub-networks' models. The performance of the detection model was estimated in a hold-out methodology, using the already presented detection test-set. The goal of this test-set is to simulate the inference on full CT images, using the described sliding-window approach. On the other hand, the segmentation model was evaluated assuming pre-detected lesions, and using a 5-fold cross validation, due to the limited number of examples available. Both networks were also compared to the best performing models for both tasks, which were introduced in Section 2.2.5.C. Furthermore, an effort to evaluate most of the key contributions of this work was made. Therefore, a set of experiments was defined, where several models were trained with a selection of features that were chosen as potentially being crucial to the final model performance. This group of tests was mostly guided towards assessing the performance of the detection model due to its lack of dependence on previous work. These included an evaluation of the impact of both weight penalization, and drop-out in the ability of the model to generalize, as well as estimating the effect of data curation and data preparation. The final experiment, still concerning the detection model, dealt with the impact of using a 2.5D approach, instead of using a single slice as a input to the network. Nonetheless, the *segmentation model* was also tested for the impact of adding residual blocks, by comparing it to the *vanilla U-Net*. Note that for all experiments, the best iteration of the model was chosen as that with the best detection accuracy, or best dice score for the segmentation task, both on the validation dataset. This was performed in order to minimise overfitting through early stopping.

The overfitting of each model was evaluated for both the training set and the validation-set, as well as through the comparison of the performance of the model in the test-set and training set. This assessment is possible, first, through the computation of the loss function at each training iteration, enabling the assessment of the convergence of the model for the training and validation sets; and then, by assessing how well the model performs in the test-set, through several suitable metrics specific to the tasks at hand. In the case of the detection task, the accuracy was used to evaluate the performance of the model during training. Yet, this metric is not applicable in the case of the test-set, as the example distribution is extremely unbalanced, and would lead to an overoptimistic performance estimation. Therefore, for the test-set, the *receiver operating characteristic* (ROC) curve was computed, using the probability distribution provided by the output of the softmax. From here, it was possible to compute the *area under the curve* (AUC). In addition to this metric, the confusion matrix was also calculated, and other metrics were derived. Namely, the *true positive rate* (TPR), also called sensitivity, or recall, and the *true negative rate* (TNR), also commonly known as specificity. The AUC and the TNR were patch-wise evaluated, in order to have an estimate of the performance of the sliding-window approach, whereas the TPR was computed slice-wise, as the fraction of true positives and the total number of lesions. Note, that the

assessment of the TPR and TNR of the models was done by performing an *arg-max* on the output of the softmax. This is equivalent to using a threshold of 0.5. It is also important to mention that the TPR was chosen in accordance to common practices in the literature [72], and complemented with the AUC and the TNR order to have a better appraisal of the novel approach. On the other hand, in the case of the segmentation models, only the loss function was used to assess the models performance during training, as it was derived directly from the most common metric of evaluation for medical segmentation: the dice-score. The final performance of the model was assessed in the test-set, using the dice-score, which was comprised of variably sized VOI, extracted around a lesion. Notice that this approach intends to mimic the behaviour of the full hybrid LungSD-Net. Finally, it is relevant to observe that the *Jaccard*-index was not included, due its equivalence to the dice coefficient by performing: $D = 2J/(1+J)$, where the D represents the dice coefficient and J represents the Jaccard index.

5.2 Detection Task

The analysis of the detection network will focus on evaluating the final 2.5D *residual DCNN*, first showcasing three ablation tests, and then comparing the best model with results from the literature. Nonetheless, the empirical assessment of the *NaiveNet*, and of the *NaiveConvNet* on the LIDC-IDRI dataset, showed the superior performance of the 2.5D *residual DNN*. Note that all the figures regarding the results of the following set of experiments are compiled in Appendix D. A table with a summary of all results (Tab. D.1) can also be found in Appendix D.

5.2.1 Impact of regularization

The first ablation test performed intended to assess the contribution of the weight penalization design and the addition of drop-out during training, therefore, three models were compared:

1. *detection model 1*: 2D residual DCNN with no weight penalization, and no drop-out;
2. *detection model 2*: 2D residual DCNN with both ℓ_1 and ℓ_2 regularization, but no drop-out;
3. *detection model 3*: 2D residual DCNN with weight penalization, and drop-out.

Table 5.1: Comparison of the models used to assess the impact of regularization to the model. Evaluation metrics are AUC from the ROC curve, true positive rate (TPR), and true negative rate (TNR).

Model	**Evaluation Metrics**		
	AUC	TNR	TPR
detection model 1	0.66	0.645	0.710
detection model 2	0.78	0.734	0.846
detection model 3	**0.80**	**0.766**	**0.871**

In Figure D.2, in Appendix D, it is possible to compare the impact of the regularization in all models. It is clear that the weight penalization term highly increases the value of the loss during training, and both regularization strategies make the model converge slower. In specific, the loss function is the same for *the detection model 1*, as the the regularization weights are zero. The values of the accuracy during training, which are visible in Figure D.1, also in Appendix D, already showcase the impact of these added methodologies, where the added regularization leads to an increase of the accuracy in the validation-set. In addition, the evaluation of the models in the test-set (Tab. 5.1) help strengthen this conclusion, as all the performance metrics assessed - AUC, TPR and TNR - increase from the *detection model 1* to the *detection model 2*, and then to the *detection model 3*. Furthermore, it is possible to note that the added weight penalization terms contribute significantly to the increase in performance, with a relevant benefit in the decrease of false positives.

5.2.2 Impact of data curation and augmentation

Table 5.2: Comparison of the models used to assess the impact of regularization to the model. Evaluation metrics are AUC from the ROC curve, true positive rate (TPR), and true negative rate (TNR).

Model	**Evaluation Metrics**		
	AUC	TNR	TPR
detection model 3	**0.80**	**0.766**	**0.871**
detection model 4	0.56	0.548	0.597
detection model 5	0.74	0.718	0.806

In order to estimate the impact of data curation and augmentation, two different models were trained and evaluated for the second experiment performed. These are:

1. *detection model 4*: 2D residual CNN trained with only raw images and no data augmentation;
2. *detection model 5*: 2D residual DCNN trained with pre-processed images, but no data augmentation.

These were compared with the *detection model 3* already mentioned. For *detection model 4*, all the steps of resampling, lung extraction, and normalization were excluded, and the example sampling was redefined in accordance: the false examples were extracted with no biased sampling. The plots for the accuracy of the model during training are shown in Figure D.1, in Appendix D. As visible, for both cases, training is much more unstable, taking many more iterations to train, with the first model only converging after 7000 iterations, and not being able to generalize to the validation-set, and the second model eventually being able to converge after more than 15000 iterations, with a slight decrease in accuracy in the middle of the training.

The evaluation of the performance in the test-set also contributes to the conclusion that both data

augmentation and image pre-processing have an impact in the final models' performance. This information is summarized in Table D.1, where it is possible to observe the large decrease in performance when the both data curation, and image pre-processing are removed from the pipeline. Several factors may contribute to the inability of *detection model 4* to generalize in comparison to the model only lacking data augmentation. These can be the large range of values of the images' voxels, which are known cause instability when training NN [51], as well as the multiple steps to curate the examples for training the model [101].

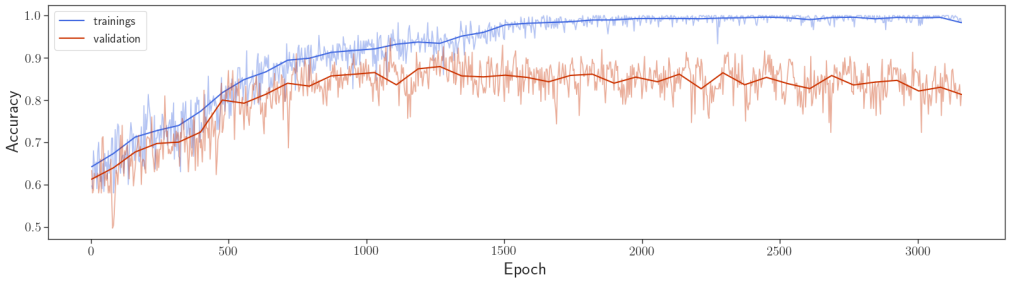
Also, when compared with the 2D *residual CNN* trained with fully pre-processed, curated, and augmented images, the results of the *detection model 4* fall slightly short. It is thus possible to claim that the augmentation techniques used also contribute, both to the training and the generalization of the final model.

5.2.3 Impact of a 2.5D approach

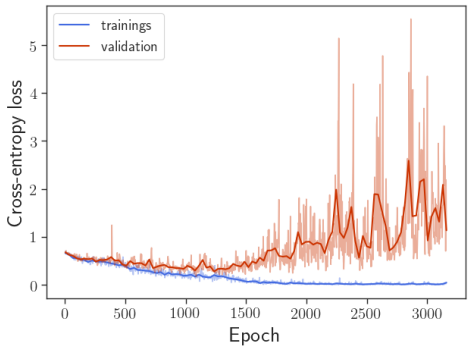
Table 5.3: Comparison of the models used to assess the impact of regularization to the model. Evaluation metrics are AUC from the ROC curve, true positive rate (TPR), and true negative rate (TNR).

Model	**Evaluation Metrics**		
	AUC	TNR	TPR
detection model 3	0.80	0.766	0.871
2.5D residual CNN	**0.87**	**0.830**	**0.901**

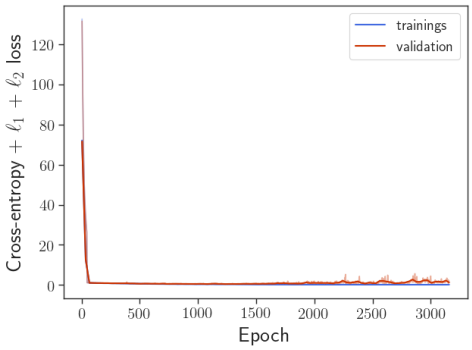
Finally, to evaluate the impact of the added slices it possible to compare the performance of the already shown 2D residual CNN - the *detection model 3*, and a 2.5D residual CNN, which has one slice above, and one slice below, in the axial plane - the one with the lowest spatial resolution. Note that in this test, both models were trained with the full pipeline, so the only difference is the number of input image slices. It is possible to have a better assessment of the performance in the test-set by looking at Table 5.3. Although training is slightly more unstable, the overall evaluation metrics still allow us to conclude that the model with added context outperforms the previous model. It is also relevant to note that decrease in performance in the 2D residual CNN is more noticeable with respect to the TNR. One possible explanation is that the added context allows the model to better detect common false positives, such as vascular structures, which are common confounders in the axial plane of CT scans [7]. This is supported by the fact that anatomical information, specially in the transverse direction, might encode relevant features for the detection of these structures,



(a) *Accuracy measured during training*



(b) *Cross-entropy loss during training*



(c) *Cross-entropy and weight penalization loss during training*

Figure 5.1: Metrics assessed during training for the final detection model.

5.2.4 Comparison with the state of the art

The evaluation of the model during training is showcased in Figure 5.1. From Figure 5.1(a), the final model was chosen by maximizing the accuracy in the validation-set; this corresponds roughly to the 1250-th iteration. Figure 5.1(b) shows that this model has a minimal cross-entropy loss in the validation-set, before it starts diverging due to overfitting.

From the ROC curve computed on the test-set, shown in Figure 5.2, an AUC of 0.87 was computed. Taking into account the many false examples in the test-set, it can be claimed that AUC = 0.87 is a meaningful value for a sliding-window approach. Nonetheless, a TNR of 0.83 leads to 196.52 false positives per slice, which is a number comparable to [105], but higher than the 15.0 candidates/slice, reported by [19]. Therefore, it is possible to suggest that the LungsSD-Net would greatly benefit from a false positive reduction model, such as those presented in [19, 105]. The analysis of a slice-wise nodule detection rate can be seen in Table 5.4, where we compare our results with the state of the art for lung lesion detection. Note that both the YOLO-based approach [26], and the RetinaNet model [6] were not presented, as no values for the detection task were found; a particle swarm optimisation based model [17] was also excluded from the comparison, as it didn't perform patient-wise separation when

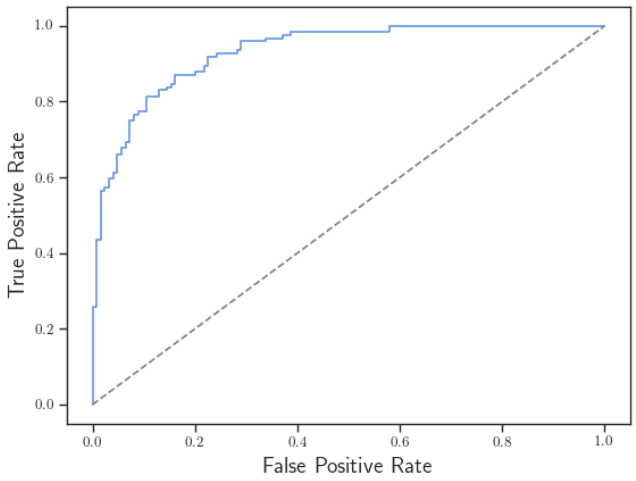


Figure 5.2: ROC of the lesion detection task (AUC = 0.87)

building the test-set.

Table 5.4: State-of-the-art 2D detection models comparison.

Model	Recall
3D dual path network R-CNN [109]	**0.958**
3D Deep CNN w/ lung extraction [34]	0.947
2.5D Faster RCNN w/ deconv layer [19]	0.946
YOLOv2 w/ InceptionV3 backbone [7]	0.890
triple 2D Faster RCNN w/ deconv layer and RPN ensemble [105]	0.864
2.5D residual DCNN (proposed)	0.902

In comparison to much more complex architectures and training methodologies, the proposed, much simpler, 2.5D residual DCNN achieves recall values on par with the state of the art. Arguably, regularization and data augmentation largely contribute to the good results herein reported. Also, reducing the problem to the inside of the lungs allowed to benefit from resampling of the training examples, which may contribute to the ability of the network to learn more relevant features from within the lungs. This may justify to the comparable results achieved, as none of the non-3D models performed lung extraction, and only one of the 3D models used it as pre-processing step. Moreover, it is important to mention that the 2.5D *Faster RCNN* was also helped by extensive data augmentation, as well as positive oversampling. Finally, it is important to note that a disparity in the size of lesions that are considered for analysis currently exists, but that this work followed the parameters set by the *Fleischner Society*.

In Figure 5.3, the comparison between the number of false negatives per lesion size, and the size distribution of examples in the training set is presented. Although the number of nodules drastically

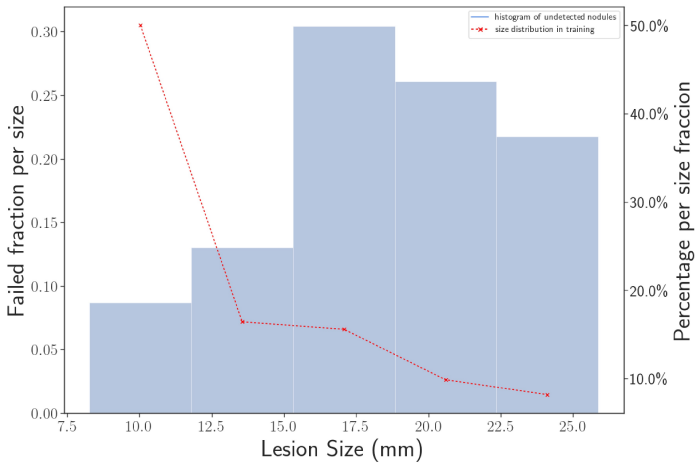


Figure 5.3: Comparison between false negatives count in the test-set and percentage of examples, both with respect to the lesion size. An inverse pattern trend is noticeable with an increase of false negative counts as the percentage of examples decrease and the size of the lesions increases.

decreases in the dataset, the number of positive patch examples decreases at a slower rate, as it is possible to extract a greater number of examples per lesion. As expected, due to the fewer amount of larger examples, most of the undetected lesions coincide with larger sized lesions. It is also possible to hypothesise that this trend is counterbalanced for larger nodules, as its sheer size might make them easier to detect, as presented in Figure 5.3. Another reason for the low amount of undetected small lesions, is the fact that a 2.5D approach is being used. This may have contributed to the detection of small lesions, as they become distinguishable from other similar structures, such as blood vessels, which are present in multiple slices. Nonetheless, a more comprehensive analysis of this detection model still needs to be performed using a larger, independent test-set.

5.3 Segmentation

5.3.1 Vanilla U-Net vs residual U-Net

The goal of this experiment was to compare the impact of residual connections in the task of lung lesion segmentation. For that, using the already established pre-processing pipeline, as well as the training methodology specified in Section 4.3.2, a *vanilla U-Net*, and a *residual U-Net*, both following the modified U-Net architecture described in Section 4.3.1.A.

These two models share an extremely similar pattern of loss convergence, with a possible indication of less overfitting in the *modified residual U-Net*. The final assessment of the models in the test-set also only shows marginal increase in the dice score of the second model, at the cost of a slight decrease in

Table 5.5: Comparison of the *vanilla U-Net*, and a *residual U-Net* models in a 5-fold cross validation. The used evaluation metric was the dice-score.

Model	dice score
modified vanilla U-Net	0.692±0.115
modified residual U-Net	**0.709±0.124**

deviation. This lead us to choose the *modified residual U-Net* for the rest of the work.

5.3.2 Comparison with the state of the art

As shown in Table 5.6, the *modified residual U-Net* achieved 0.709 ± 0.124 dice score (assessed by 5-fold CV). This score outperforms the best model [42] so far (Table 5.6), without using a 3D approach nor a cascaded network. It is also important to mention that the Central Focused double branched CNN [101] outperforms both models, with a dice-score of 0.802, tested in an external test-set, but is trained in a dataset which contains 893 nodules. A 2.5D approach was also tested, with training results comparable to the proposed model, but presenting worse testing performance. Arguably, for the specific case of the detected nodules, the added slices in a 2.5D approach only contribute to overfitting the model to the training set. An exploratory assessment of a recent 3D segmentation model was also performed. That network, called the *HighResNet* [57], allowed the improvement of multi-class segmentation of brain lesions, in MRI data. Nonetheless, the early tests performed for lung lesion in CT scans showed a failure of the model to fully segment the lesions.

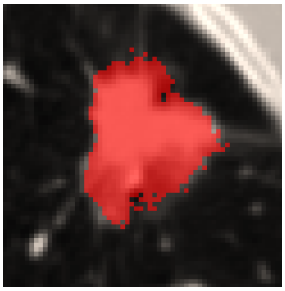
Finally, Figure 5.4 shows two examples of segmented lesions, depicting one of the highest scoring segmentations (Figure 5.4 (a) and (b)), and then one of lowest scoring segmentations (Figure 5.4 (c) and (d)). As visible in both examples, the network is able to produce accurate segmentations in solid nodules, but still struggles to provide a reliable segmentation of sub-solid nodules, as is the case of the second segmented lesion shown.

Table 5.6: Comparison of the dice score with state-of-the-art model in the Decathlon-Lung task for lesion segmentation.

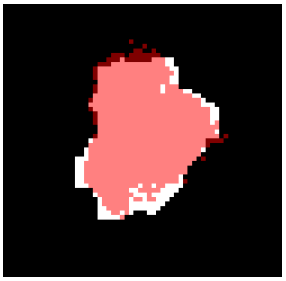
Model	Mean dice score
3D Cascade U-Net [42]	0.692
2D modified residual U-Net (proposed)	**0.709**

Another important remark is that this comparison validates the *LungSD-Net* as a viable approach for lesion segmentation. However, in order to achieve clinically relevant results, it is still mandatory to implement several improvements to its first stage, as these are directly connected to the segmnetation task. Nonetheless, the proposed architecture proves to be viable for the segmentation task at hand, without an extensive training setup, even when trained in a small dataset. These results illustrate the

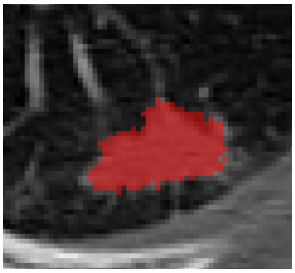
robustness of the model for the segmentation of lesions with challenging textures.



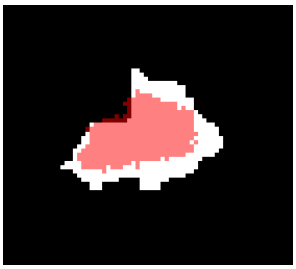
(a) Output mask overlaying the original scan



(b) Output mask overlaying the ground-truth segmentation



(c) Output mask overlaying the original scan



(d) Output mask overlaying the ground-truth segmentation

Figure 5.4: Example of segmentation with the *modified residual U-Net*. a) and b) present one of the segmentations with the best dice-score, whereas c) and d) show one of the lowest scoring segmentations.

6

Final Remarks

Contents

6.1 Conclusions

The main goal of this present work was to assess the viability of relatively simple deep learning models for the detection and segmentation of lung lesions in computed tomography (CT) images. Both problems encompass one major hurdle: the need to overcome the high tendency for overfitting that exists in the field of medical image segmentation. In this scenario, the problem at hand was approached through a thorough pre-processing pipeline, as well as careful data curation and data augmentation. This included lung extraction, which helped to deal with false positives from outside of the lungs' parenchyma, and both resampling and normalization. A novel region-based hybrid model for detection and segmentation of lung lesions on 3D CT scans, which is comprised of a new 2.5D residual *deep convolutional neural network* (DCNN) and a modified residual U-Net was also proposed. The results obtained in the detection task support the hypothesis that simpler DCNN architectures are able to achieve relevant results in medical imaging, through careful design of the training scheme, powerful data augmentation, and state-of-the-art architectural modifications, as this model was able to produce competitive results in comparison with other current state-of-the-art models. This simpler, but highly accurate, model has lower hardware requirements, thus is potentially of more widespread applicability. The results herein presented also provide evidence for the advantages of reducing the problem of segmentation to the *volume of interest* (VOI) around each detected lesion. This is a first step towards the creation of a fully automated segmentation tool based on deep learning that first relies on the detection of lesions in a sliding-window fashion.

6.2 Limitations and Future Work

In the field of medical imaging, the application of deep learning models involves two major problems. The first one is data availability, which due to the extensive requirements of deep learning models, leads to severe limitations when considering the costs, ethical hurdles, and very time-consuming annotation of medical data. We were fortunate to have been able to use the Lung Image Database Consortium - Image Database Resource Initiative (LIDC-IDRI) dataset, which provided a solid foundation for the training of the detection model. Nonetheless, a requirement for a clinically applicable model is to have it evaluated in an external test set, so this is an absolutely necessary step in the continuation of this work. This need for more data is even more essential in the case of the segmentation task, where the dataset was extremely reduced. In the future, it would also be interesting to tackle this problem with non-conventional data augmentation techniques, such as the use of generative models (such as generative adversarial networks) [13, 24, 35], to produce artificial images. Transfer learning is also an attractive technique to increase the efficiency of the available data, which becomes even more appealing with recent evidence that it might allow the use of light-weight models [75]. Currently, one of the frontiers

that has started to expand is the field of unsupervised learning in medical image [33]. This would also contribute to lessen the requirements on data annotation, and may be an attractive direction for future work. Interpretability is the second common issue that is brought up when dealing with deep learning models, which is of extreme importance within the clinical context. Therefore, we recommend the study of potential interpretability in this work through methods such us SHAP [61].

Regarding actual changes to the *LungSD-Net*, there is quite some room from improvement. Due to its sliding window approach, and the limited computational, and memory requirements, one clear improvement to the model is to expand it to 3D. Also, the addition of a false positive reduction sub-network between the detection model, and the segmentation model, could potentially drastically improve the TNR of the detection stage. Dense connections were never tried in this work, and could also lead to overall improved performance of the model.

Finally, as the ultimate goal of this project would be to have it applied in the context of clinical practice, we suggest the integration of the improved model into a CAD system, and then testing the potential gains when used by radiologists. A semi-supervised method for the segmentation task would also be expected to yield a more clinically valid mask of the VOI, through its iterative refinement by the radiologists. The integration of this work in a radiomics pipeline by using the radiomic features extracted from the VOI of each lesion, and then perform its subsequent analysis and classification, is also a logic next step.

6.3 Reproducibility

During this project, a large effort was made to assure its reproducibility. All the programming involved relied on open source tools, such as *Python* libraries, visualization tools, and deep learning packages, such as *NiftyNet*. Furthermore, all the code was structured as a library that provides several structured functions for all the steps in CT scan preprocessing, as well as data curation and augmentation for training machine learning models. The same library includes scripts that integrate all the function into pipelines for and end-to-end approach of prepraring the data. A separate module dedicated to synthetic images production was also included in the package, which could be used for exploratory analysis of machine learning models.

These tools are fully documented and will enable the reproduction of this work. This resource can be accessed through this link: https://github.com/JoaoCarv/LungSDNet.

Bibliography

[1] Who cancer fact sheet 2018. https://www.who.int/news-room/fact-sheets/detail/cancer.

[2] ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G. S., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., GOODFELLOW, I., HARP, A., IRVING, G., ISARD, M., JIA, Y., JOZEFOWICZ, R., KAISER, L., KUDLUR, M., LEVENBERG, J., MANÉ, D., MONGA, R., MOORE, S., MURRAY, D., OLAH, C., SCHUSTER, M., SHLENS, J., STEINER, B., SUTSKEVER, I., TALWAR, K., TUCKER, P., VANHOUCKE, V., VASUDEVAN, V., VIÉGAS, F., VINYALS, O., WARDEN, P., WATTENBERG, M., WICKE, M., YU, Y., AND ZHENG, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] AFSHAR, P., MOHAMMADI, A., AND PLATANIOTIS, K. N. Brain tumor type classification via capsule networks. In *2018 25th IEEE International Conference on Image Processing (ICIP)* (2018), IEEE, pp. 3129–3133.

[4] ALOM, M. Z., HASAN, M., YAKOPCIC, C., TAHA, T. M., AND ASARI, V. K. Recurrent residual convolutional neural network based on u-net (R2U-net) for medical image segmentation. *arXiv preprint arXiv:1802.06955* (2018).

[5] ALOM, M. Z., HASAN, M., YAKOPCIC, C., TAHA, T. M., AND ASARI, V. K. Recurrent residual convolutional neural network based on u-net (R2U-net) for medical image segmentation. *arXiv preprint arXiv:1802.06955* (2018).

[6] ARDILA, D., KIRALY, A. P., BHARADWAJ, S., CHOI, B., REICHER, J. J., PENG, L., TSE, D., ETEMADI, M., YE, W., CORRADO, G., ET AL. End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. *Nature medicine 25*, 6 (2019), 954.

[7] ARESTA, G., ARAÚJO, T., JACOBS, C., VAN GINNEKEN, B., CUNHA, A., RAMOS, I., AND CAMPILHO, A. Towards an automatic lung cancer screening system in low dose computed tomography. In *Image Analysis for Moving Organ, Breast, and Thoracic Images*. Springer, 2018, pp. 310–318.

[8] ARMATO III, S. G., MCLENNAN, G., BIDAUT, L., MCNITT-GRAY, M. F., MEYER, C. R., REEVES, A. P., ZHAO, B., ABERLE, D. R., HENSCHKE, C. I., HOFFMAN, E. A., ET AL. The lung image database consortium (LIDC) and image database resource initiative (IDRI): a completed reference database of lung nodules on ct scans. *Medical physics 38*, 2 (2011), 915–931.

[9] BANKIER, A. A., MACMAHON, H., GOO, J. M., RUBIN, G. D., SCHAEFER-PROKOP, C. M., AND NAIDICH, D. P. Recommendations for measuring pulmonary nodules at CT: a statement from the Fleischner Society. *Radiology 285*, 2 (2017), 584–600.

[10] BELKIN, M., HSU, D., MA, S., AND MANDAL, S. Reconciling modern machine learning and the bias-variance trade-off. *arXiv preprint arXiv:1812.11118* (2018).

[11] BISHOP, C. M. *Pattern recognition and machine learning*. springer, 2006.

[12] BOTTOU, L. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

[13] BOWLES, C., CHEN, L., GUERRERO, R., BENTLEY, P., GUNN, R., HAMMERS, A., DICKIE, D. A., HERNÁNDEZ, M. V., WARDLAW, J., AND RUECKERT, D. GAN augmentation: augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863* (2018).

[14] BRENNER, D. J., AND HALL, E. J. Computed tomography—an increasing source of radiation exposure. *New England Journal of Medicine 357*, 22 (2007), 2277–2284.

[15] CHEN, M., SHI, X., ZHANG, Y., WU, D., AND GUIZANI, M. Deep features learning for medical image analysis with convolutional autoencoder neural network. *IEEE Transactions on Big Data* (2017).

[16] ÇIÇEK, Ö., ABDULKADIR, A., LIENKAMP, S. S., BROX, T., AND RONNEBERGER, O. 3D U-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention* (2016), Springer, pp. 424–432.

[17] DA SILVA, G. L. F., VALENTE, T. L. A., SILVA, A. C., DE PAIVA, A. C., AND GATTASS, M. Convolutional neural network-based PSO for lung nodule false positive reduction on CT images. *Computer methods and programs in biomedicine 162* (2018), 109–118.

[18] DEVALLA, S. K., RENUKANAND, P. K., SREEDHAR, B. K., SUBRAMANIAN, G., ZHANG, L., PERERA, S., MARI, J.-M., CHIN, K. S., TUN, T. A., STROUTHIDIS, N. G., ET AL. DRUNET: a dilated-residual u-net deep learning network to segment optic nerve head tissues in optical coherence tomography images. *Biomedical optics express 9*, 7 (2018), 3244–3265.

[19] DING, J., LI, A., HU, Z., AND WANG, L. Accurate pulmonary nodule detection in computed tomography images using deep convolutional neural networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2017), Springer, pp. 559–567.

[20] DOZAT, T. Incorporating nesterov momentum into adam.

[21] DROZDZAL, M., VORONTSOV, E., CHARTRAND, G., KADOURY, S., AND PAL, C. The importance of skip connections in biomedical image segmentation. In *Deep Learning and Data Labeling for Medical Applications*. Springer, 2016, pp. 179–187.

[22] ERFANI, S. M., RAJASEGARAR, S., KARUNASEKERA, S., AND LECKIE, C. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognition 58* (2016), 121–134.

[23] FARABET, C., COUPRIE, C., NAJMAN, L., AND LECUN, Y. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence 35*, 8 (2012), 1915–1929.

[24] FRID-ADAR, M., DIAMANT, I., KLANG, E., AMITAI, M., GOLDBERGER, J., AND GREENSPAN, H. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing 321* (2018), 321–331.

[25] FROSST, N., SABOUR, S., AND HINTON, G. DARCCC: Detecting adversaries by reconstruction from class conditional capsules. *arXiv preprint arXiv:1811.06969* (2018).

[26] GEORGE, J., SKARIA, S., VARUN, V., ET AL. Using YOLO based deep learning network for real time detection and localization of lung nodules from low dose CT scans. In *Medical Imaging 2018: Computer-Aided Diagnosis* (2018), vol. 10575, International Society for Optics and Photonics, p. 105751I.

[27] GIBSON, E., LI, W., SUDRE, C., FIDON, L., SHAKIR, D. I., WANG, G., EATON-ROSEN, Z., GRAY, R., DOEL, T., HU, Y., ET AL. NiftyNet: a deep-learning platform for medical imaging. *Computer methods and programs in biomedicine 158* (2018), 113–122.

[28] GILLIES, R. J., KINAHAN, P. E., AND HRICAK, H. Radiomics: Images Are More than Pictures, They Are Data.

[29] GLOROT, X., BORDES, A., AND BENGIO, Y. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (2011), pp. 315–323.

[30] GOODFELLOW, I., BENGIO, Y., AND COURVILLE, A. *Deep learning*. MIT press, 2016.

[31] GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M., XU, B., WARDE-FARLEY, D., OZAIR, S., COURVILLE, A., AND BENGIO, Y. Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680.

[32] GRAY, H. *Anatomy of the human body*, vol. 8. Lea & Febiger, 1878.

[33] GREENSPAN, H., VAN GINNEKEN, B., AND SUMMERS, R. M. Deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging 35*, 5 (2016), 1153–1159.

[34] GRUETZEMACHER, R., GUPTA, A., AND PARADICE, D. 3D deep learning for detecting pulmonary nodules in CT scans. *Journal of the American Medical Informatics Association 25*, 10 (2018), 1301–1310.

[35] HAN, C., HAYASHI, H., RUNDO, L., ARAKI, R., SHIMODA, W., MURAMATSU, S., FURUKAWA, Y., MAURI, G., AND NAKAYAMA, H. GAN-based synthetic brain MR image generation. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)* (2018), IEEE, pp. 734–738.

[36] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[37] HOCHREITER, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6*, 02 (1998), 107–116.

[38] HORNIK, K., STINCHCOMBE, M., AND WHITE, H. Multilayer feedforward networks are universal approximators. *Neural networks 2*, 5 (1989), 359–366.

[39] HUANG, C., LI, Y., CHANGE LOY, C., AND TANG, X. Learning deep representation for imbalanced classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 5375–5384.

[40] HUANG, J., RATHOD, V., SUN, C., ZHU, M., KORATTIKARA, A., FATHI, A., FISCHER, I., WOJNA, Z., SONG, Y., GUADARRAMA, S., ET AL. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 7310–7311.

[41] HUSSAIN, Z., GIMENEZ, F., YI, D., AND RUBIN, D. Differential data augmentation techniques for medical imaging classification tasks. In *AMIA Annual Symposium Proceedings* (2017), vol. 2017, American Medical Informatics Association, p. 979.

[42] ISENSEE, F., PETERSEN, J., KLEIN, A., ZIMMERER, D., JAEGER, P. F., KOHL, S., WASSERTHAL, J., KOEHLER, G., NORAJITRA, T., WIRKERT, S., ET AL. nnu-net: Self-adapting framework for u-net-based medical image segmentation. *arXiv preprint arXiv:1809.10486* (2018).

[43] JIMÉNEZ-SÁNCHEZ, A., ALBARQOUNI, S., AND MATEUS, D. Capsule networks against medical imaging data challenges. In *Intravascular Imaging and Computer Assisted Stenting and Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*. Springer, 2018, pp. 150–160.

[44] KAK, A. C., SLANEY, M., AND WANG, G. Principles of computerized tomographic imaging. *Medical Physics 29*, 1 (2002), 107–107.

[45] KINGMA, D. P., AND BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[46] KONTSCHIEDER, P., FITERAU, M., CRIMINISI, A., AND ROTA BULO, S. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1467–1475.

[47] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105.

[48] KRUTHIKA, K., MAHESHAPPA, H., INITIATIVE, A. D. N., ET AL. CBIR system using Capsule Networks and 3d CNN for Alzheimer's disease diagnosis. *Informatics in Medicine Unlocked 14* (2019), 59–68.

[49] KUO, C.-C. J. Understanding convolutional neural networks with a mathematical model. *Journal of Visual Communication and Image Representation 41* (2016), 406–413.

[50] KURIHARA, Y., MATSUOKA, S., YAMASHIRO, T., FUJIKAWA, A., MATSUSHITA, S., YAGIHASHI, K., AND NAKAJIMA, Y. MRI of pulmonary nodules. *American journal of roentgenology 202*, 3 (2014), W210–W216.

[51] LAPEDES, A. S., AND FARBER, R. M. How neural nets work. In *Neural information processing systems* (1988), pp. 442–456.

[52] LECUN, Y., BENGIO, Y., AND HINTON, G. Deep learning. *nature 521*, 7553 (2015), 436.

[53] LECUN, Y., BOSER, B. E., DENKER, J. S., HENDERSON, D., HOWARD, R. E., HUBBARD, W. E., AND JACKEL, L. D. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems* (1990), pp. 396–404.

[54] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[55] Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning* (2009), ACM, pp. 609–616.

[56] Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems* (2018), pp. 6389–6399.

[57] Li, W., Wang, G., Fidon, L., Ourselin, S., Cardoso, M. J., and Vercauteren, T. On the compactness, efficiency, and representation of 3D convolutional networks: brain parcellation as a pretext task. In *International Conference on Information Processing in Medical Imaging* (2017), Springer, pp. 348–360.

[58] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 2980–2988.

[59] Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B., and Sánchez, C. I. A survey on deep learning in medical image analysis. *Medical image analysis 42* (2017), 60–88.

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

[61] Lundberg, S. M., and Lee, S.-I. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774.

[62] Maas, A. L., Hannun, A. Y., and Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml* (2013), vol. 30, p. 3.

[63] Marchesi, M. Megapixel size image creation using generative adversarial networks. *arXiv preprint arXiv:1706.00082* (2017).

[64] Mazurowski, M. A., Buda, M., Saha, A., and Bashir, M. R. Deep learning in radiology: An overview of the concepts and a survey of the state of the art with focus on MRI. *Journal of Magnetic Resonance Imaging 49*, 4 (2019), 939–954.

[65] MURPHY, K. P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[66] NAIR, A., BARTLETT, E. C., WALSH, S. L., WELLS, A. U., NAVANI, N., HARDAVELLA, G., BHALLA, S., CALANDRIELLO, L., DEVARAJ, A., GOO, J. M., ET AL. Variable radiological lung nodule evaluation leads to divergent management recommendations. *European Respiratory Journal 52*, 6 (2018), 1801359.

[67] NEGAHDAR, M., BEYMER, D., AND SYEDA-MAHMOOD, T. Automated volumetric lung segmentation of thoracic CT images using fully convolutional neural network. In *Medical Imaging 2018: Computer-Aided Diagnosis* (2018), vol. 10575, International Society for Optics and Photonics, p. 105751J.

[68] OPENSTAX. *Anatomy and Physiology*. OpenStax CNX, 2016. http://cnx.org/contents/14fb4ad7-39a1-4eee-ab6e-3ef2482e3e22@8.24.

[69] OUDKERK, M., DEVARAJ, A., VLIEGENTHART, R., HENZLER, T., PROSCH, H., HEUSSEL, C. P., BASTARRIKA, G., SVERZELLATI, N., MASCALCHI, M., DELORME, S., ET AL. European position statement on lung cancer screening. *The Lancet Oncology 18*, 12 (2017), e754–e766.

[70] PAPANIKOLAOU, N., AND SANTINHA, J. An introduction to radiomics: Capturing tumour biology in space and time. *Hellenic Journal of Radiology 3*, 1 (2018).

[71] PASZKE, A., GROSS, S., CHINTALA, S., CHANAN, G., YANG, E., DEVITO, Z., LIN, Z., DESMAISON, A., ANTIGA, L., AND LERER, A. Automatic differentiation in PyTorch. In *NIPS-W* (2017).

[72] PEHRSON, L. M., NIELSEN, M. B., AND AMMITZBØL LAURIDSEN, C. Automatic pulmonary nodule detection applying deep learning or machine learning algorithms to the LIDC-IDRI database: A systematic review. *Diagnostics 9*, 1 (2019), 29.

[73] PEREZ, L., AND WANG, J. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621* (2017).

[74] QIAN, N. On the momentum term in gradient descent learning algorithms. *Neural networks 12*, 1 (1999), 145–151.

[75] RAGHU, M., ZHANG, C., KLEINBERG, J. M., AND BENGIO, S. Transfusion: Understanding transfer learning for medical imaging.

[76] RAMACHANDRAN, P., ZOPH, B., AND LE, Q. V. Searching for activation functions. *arXiv preprint arXiv:1710.05941* (2017).

[77] RAWAT, W., AND WANG, Z. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation 29*, 9 (2017), 2352–2449.

[78] RAZZAK, M. I., NAZ, S., AND ZAIB, A. Deep learning for medical image processing: Overview, challenges and the future. In *Classification in BioApps*. Springer, 2018, pp. 323–350.

[79] REDMON, J., DIVVALA, S., GIRSHICK, R., AND FARHADI, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 779–788.

[80] REDMON, J., AND FARHADI, A. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 7263–7271.

[81] REN, S., HE, K., GIRSHICK, R., AND SUN, J. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (2015), pp. 91–99.

[82] REN, S., HE, K., GIRSHICK, R., ZHANG, X., AND SUN, J. Object detection networks on convolutional feature maps. *IEEE transactions on pattern analysis and machine intelligence 39*, 7 (2016), 1476–1481.

[83] RONNEBERGER, O., FISCHER, P., AND BROX, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (2015), Springer, pp. 234–241.

[84] SABOUR, S., FROSST, N., AND HINTON, G. Matrix capsules with EM routing. In *6th International Conference on Learning Representations, ICLR* (2018).

[85] SABOUR, S., FROSST, N., AND HINTON, G. E. Dynamic routing between capsules. In *Advances in neural information processing systems* (2017), pp. 3856–3866.

[86] SERGIEVSKIY, N., AND PONAMAREV, A. Reduced focal loss: 1st place solution to xview object detection in satellite imagery. *arXiv preprint arXiv:1903.01347* (2019).

[87] SHEN, D., WU, G., AND SUK, H.-I. Deep learning in medical image analysis. *Annual review of biomedical engineering 19* (2017), 221–248.

[88] SIEGEL, R. L., MILLER, K. D., AND JEMAL, A. Cancer statistics, 2019. *CA: a cancer journal for clinicians 69*, 1 (2019), 7–34.

[89] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[90] SIMPSON, A. L., ANTONELLI, M., BAKAS, S., BILELLO, M., FARAHANI, K., VAN GINNEKEN, B., KOPP-SCHNEIDER, A., LANDMAN, B. A., LITJENS, G., MENZE, B., ET AL. A large annotated

medical image dataset for the development and evaluation of segmentation algorithms. *arXiv preprint arXiv:1902.09063* (2019).

[91] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I., AND SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research 15*, 1 (2014), 1929–1958.

[92] SRIVASTAVA, R. K., GREFF, K., AND SCHMIDHUBER, J. Highway networks. *arXiv preprint arXiv:1505.00387* (2015).

[93] SUDRE, C. H., LI, W., VERCAUTEREN, T., OURSELIN, S., AND CARDOSO, M. J. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 2017, pp. 240–248.

[94] SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V., AND RABINOVICH, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 1–9.

[95] TANG, Y. Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239* (2013).

[96] TEAM, N. L. S. T. R. Reduced lung-cancer mortality with low-dose computed tomographic screening. *New England Journal of Medicine 365*, 5 (2011), 395–409.

[97] TIBSHIRANI, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological) 58*, 1 (1996), 267–288.

[98] VAN GINNEKEN, B., ARMATO III, S. G., DE HOOP, B., VAN AMELSVOORT-VAN DE VORST, S., DUINDAM, T., NIEMEIJER, M., MURPHY, K., SCHILHAM, A., RETICO, A., FANTACCI, M. E., ET AL. Comparing and combining algorithms for computer-aided detection of pulmonary nodules in computed tomography scans: the ANODE09 study. *Medical image analysis 14*, 6 (2010), 707–722.

[99] VAPNIK, V. *The nature of statistical learning theory*. Springer science & business media, 2013.

[100] VEIT, A., WILBER, M., AND BELONGIE, S. Residual networks are exponential ensembles of relatively shallow networks. *arXiv preprint arXiv:1605.06431 1*, 2 (2016), 3.

[101] WANG, S., ZHOU, M., LIU, Z., LIU, Z., GU, D., ZANG, Y., DONG, D., GEVAERT, O., AND TIAN, J. Central focused convolutional neural networks: Developing a data-driven model for lung nodule segmentation. *Medical image analysis 40* (2017), 172–183.

[102] WIELPÜTZ, M., AND KAUCZOR, H.-U. MRI of the lung: state of the art. *Diagnostic and interventional radiology 18*, 4 (2012), 344.

[103] WOLPERT, D. H., MACREADY, W. G., ET AL. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation 1*, 1 (1997), 67–82.

[104] XI, E., BING, S., AND JIN, Y. Capsule network performance on complex data. *arXiv preprint arXiv:1712.03480* (2017).

[105] XIE, H., YANG, D., SUN, N., CHEN, Z., AND ZHANG, Y. Automated pulmonary nodule detection in CT images using deep convolutional neural networks. *Pattern Recognition 85* (2019), 109–119.

[106] XU, B., WANG, N., CHEN, T., AND LI, M. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853* (2015).

[107] ZHANG, Z., MA, L., LI, Z., AND WU, C. Normalized direction-preserving adam. *arXiv preprint arXiv:1709.04546* (2017).

[108] ZHAO, Z.-Q., ZHENG, P., XU, S.-T., AND WU, X. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems* (2019).

[109] ZHU, W., LIU, C., FAN, W., AND XIE, X. Deeplung: Deep 3D dual path nets for automated pulmonary nodule detection and classification. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2018), IEEE, pp. 673–681.

[110] ZOU, K. H., WARFIELD, S. K., BHARATHA, A., TEMPANY, C. M., KAUS, M. R., HAKER, S. J., WELLS III, W. M., JOLESZ, F. A., AND KIKINIS, R. Statistical validation of image segmentation quality based on a spatial overlap index1: scientific reports. *Academic radiology 11*, 2 (2004), 178–189.

A

Backpropagation Algorithm: Mathematical Details

Firstly, in regards to Equation 2.5, a derivation is performed with respect to a generic hidden layer vector with K units $\mathbf{x}^k = \{x_1^k, ..., x_K^k\}$, of the previous layer:

$$\frac{\partial L(F^*(\mathbf{X};\theta), Y)}{\partial x_j^k} = \sum_i \frac{\partial L(F^*(\mathbf{X};\theta), Y)}{\partial z_i^{k+1}} \frac{\partial z_i^{k+1}}{\partial x_j^k} \tag{A.1}$$

$$= \sum_i \frac{\partial L(F^*(\mathbf{X};\theta), Y)}{\partial z_i^{k+1}} W_{i,k}^{k+1}, \tag{A.2}$$

where $W_{i,k}^{k+1}$ is obtained by deriving Equation 2.5 with respect to x^k. This can be generalized to vectors as:

$$\nabla_{x^k} L(F^*(\mathbf{X};\theta), Y) = W^{(k+1)^\top} \nabla_{z^{k+1}} L(F^*(\mathbf{X};\theta), Y). \tag{A.3}$$

The next step is to apply the same rational to the pre-activation term in the same layer(Eq. 2.6),

remembering that in this case the activation function is defined as $g : \mathbb{R} \longrightarrow \mathbb{R}$. This is given by

$$\frac{\partial L(F^*(\mathbf{X};\theta),Y)}{\partial z_j^k} = \frac{\partial L(F^*(\mathbf{X};\theta),Y)}{\partial x_j^k}\frac{\partial x_j^k}{\partial z_j^k} \tag{A.4}$$

$$= \frac{\partial L(F^*(\mathbf{X};\theta),Y)}{\partial x_i^k} g'(z_j^k), \tag{A.5}$$

where $g'(z_j^k)$ is the derivative of the activation function. Similarly, this equation can be simplified by using the activation function defined in Equation 2.14, and its respective derivative $g'(z^k)$:

$$\nabla_{z^k} L(F^*(\mathbf{X};\theta),Y) = \nabla_{x^k} L(F^*(\mathbf{X};\theta),Y) \odot g'(z^k), \tag{A.6}$$

where $\odot$ is the element-wise multiplication. Here we ascertain an important requirement when building a NN architecture: in order to apply the backpropagation algorithm, $g'(z^k)$ needs to exist, which means that $g \in C^1$. Finally, the last step is to calculate the parameter gradient of the same hidden layer:

$$\frac{\partial L(F^*(\mathbf{X};\theta),Y)}{\partial W_{i,j}^k} = \frac{\partial L(F^*(\mathbf{X};\theta),Y)}{\partial z_i^k}\frac{\partial z_i^k}{\partial W_{i,j}^k} \tag{A.7}$$

$$= \frac{\partial L(F^*(\mathbf{X};\theta),Y)}{\partial z_i^k} h_j^{k-1}. \tag{A.8}$$

Further simplification yields:

$$\nabla_{W^k} L(F^*(\mathbf{X};\theta),Y) = \nabla_{z^k}.L(F^*(\mathbf{X};\theta),Y){h^{k-1}}^\top \tag{A.9}$$

Similarly, for the bias term b, a simplified equation is attained:

$$\nabla_{b^k} L(F^*(\mathbf{X};\theta),Y) = \nabla_{z^k} L(F^*(\mathbf{X};\theta),Y). \tag{A.10}$$

By applying these computations to all layers, the gradients for each hidden unit (including activations, pre-activations, weights and bias) are computed. The backpropagation algorithm connects all these calculations by beginning in the foremost layer's activation function, and computing the previous layers gradients from activation, to pre-activation, and finally weights and bias, until it reaches the first layer [52].

B

Sub-gradient

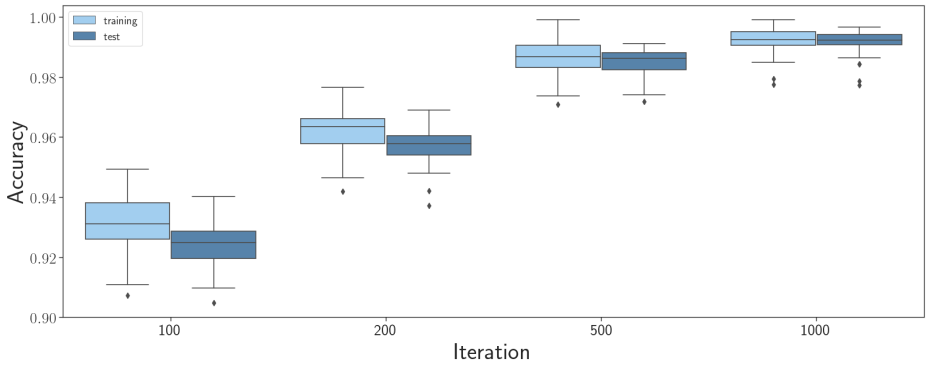
The definition of a subgradient allows is here defined in order to have a solution for the non differentiability of the *ReLu* (and the *leaky-ReLu*) at $z = 0$.

Let $s \in \mathbb{R}^K$ be defined as a subgradient of σ at $\mathbf{z} \in \mathbb{R}^K$ if $\forall \mathrm{z}' \in \mathbb{R}^K, \sigma\left(\mathrm{z}'\right) \geq \sigma(\mathrm{z}) + \mathrm{s}^T\left(\mathrm{z}' - \mathrm{x}\right)$. The subset of all subgradients of σ at $\mathbf{z}$ is then called its *subdifferential*. Now, with a slight generalization of the SGD to allow subgradients, is possible to compute the update for the loss function, even if the architecture has a non differentiable function such us the *ReLu*. Note that in both the cases of the *ReLu* and the *leaky-ReLu*, the subgradient for $z = 0$ is commonly set to zero.

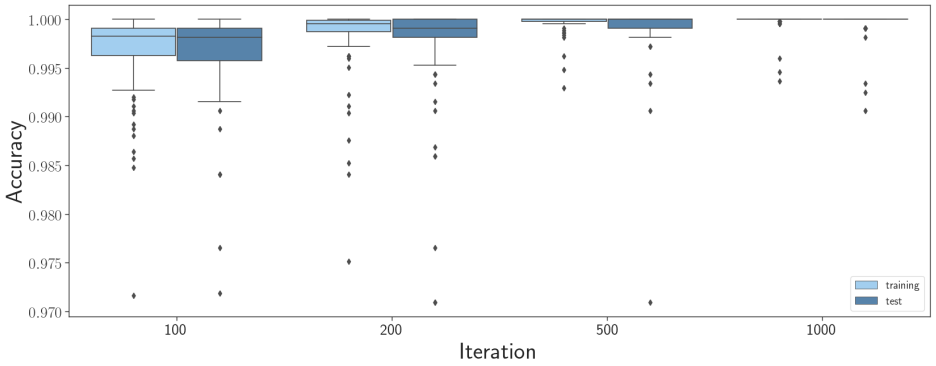
C

Detection Ablation Study

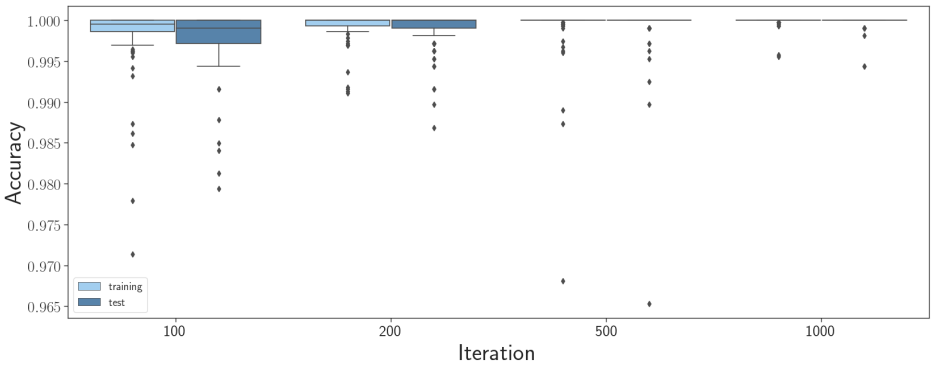
In AppendixC the results of the ablation study for the detection network are presented. The box plots presented in Figure C.1 summarize the results of the 100 trials for each one of the models.



(a) Box-plots for the Naive-Net experiment



(b) Box-plots for the NaiveCOnv-Net experiment



(c) Box-plots of the residual DCNN experiment

Figure C.1: Box-plots for the experiments performed using the three different architectures. Each figure summarizes the experiment, condensing the results for each iteration.

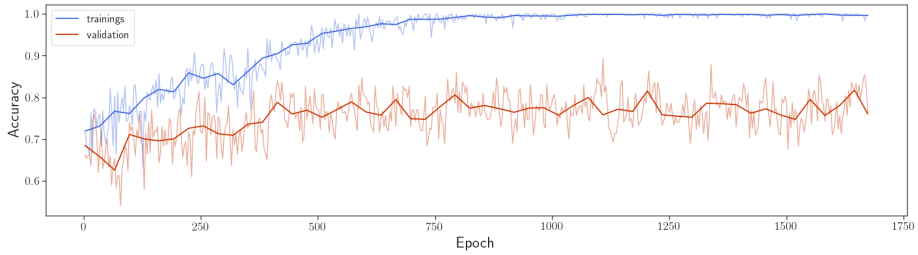
D

Detection Experiments

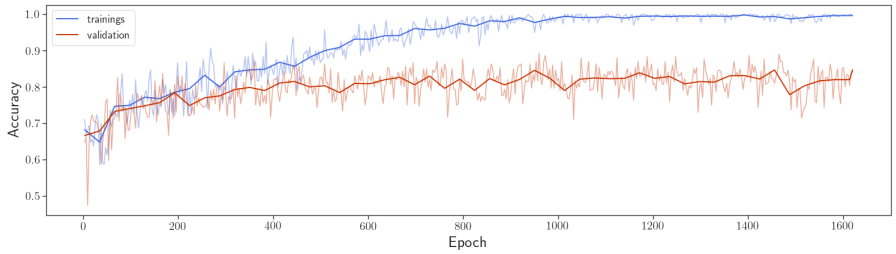
Table D.1 summarizes the results for all experiments. In comparison to all other models compared, the final model, which integrates the full pipeline of data preparation with the *2.5D residual CNN*, presents the best score for all metrics. The evaluation of the model convergence is also presented in Figure D.1, where we can see the impact of the regularization in the overfitting for Figures D.1(a), D.1(b) and D.1(c), as well as the incresed instability of the model's training in Figures D.1(d) and D.1(e), when the pre-processing and augmentation pipelines are removed.

Table D.1: Summary of the all the models performance in the test-set. Evaluation metrics are AUC from the ROC curve, true positive rate (TPR), and true negative rate (TNR).

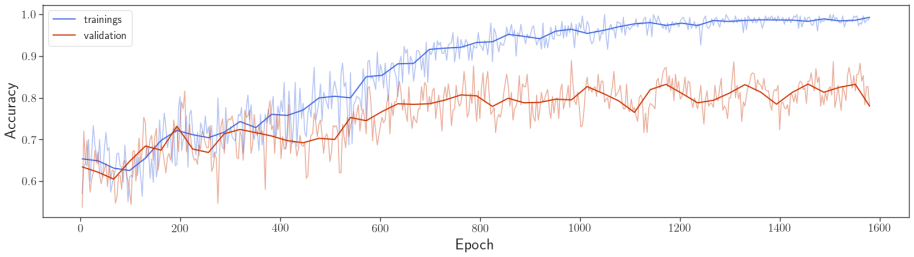
Model	**Evaluation Metrics**		
	AUC	TNR	TPR
detection model 1	0.66	0.645	0.710
detection model 2	0.78	0.734	0.846
detection model 3	0.80	0.766	0.871
detection model 4	0.56	0.548	0.597
detection model 5	0.74	0.718	0.806
final model	0.87	0.830	0.901



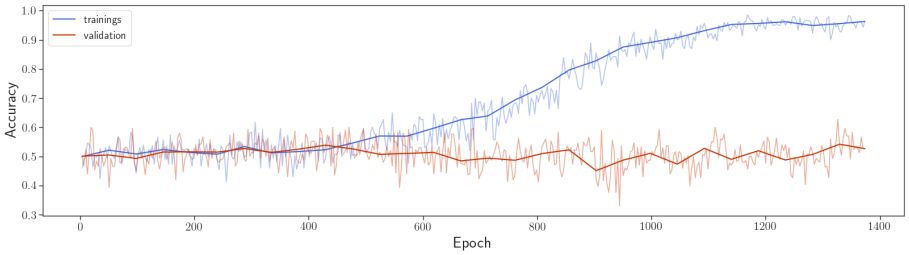
(a) *Detection model 1*



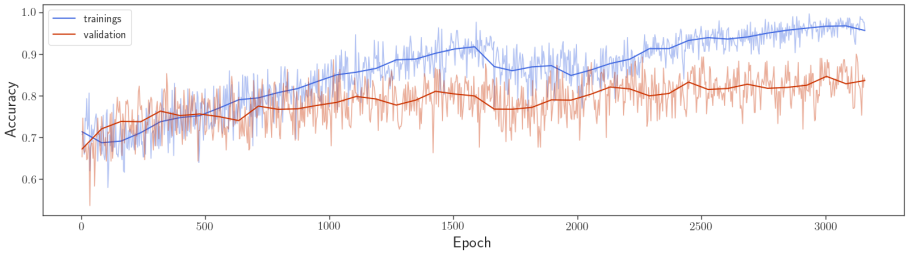
(b) *Detection model 2*



(c) *Detection model 3*

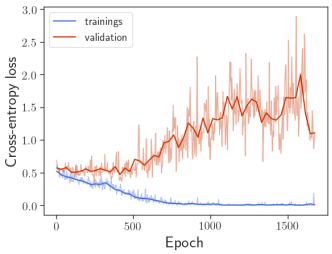


(d) *Detection model 4*

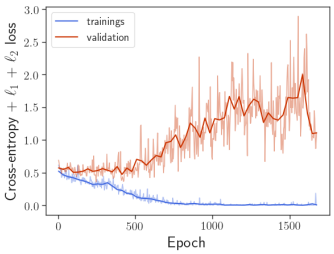


(e) *Detection model 5*

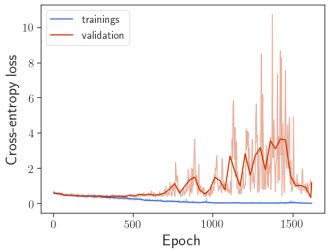
Figure D.1: Accuracy plots during training of all the models used to assess impact of regularization, data curation and augmentation, as well addition of context to the input.



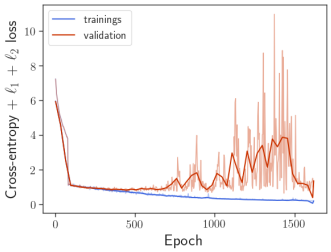
(a) *Detection model 1 classification loss*



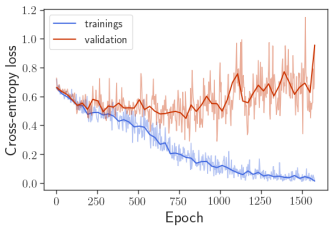
(b) *Detection model 1 classification and regularization loss*



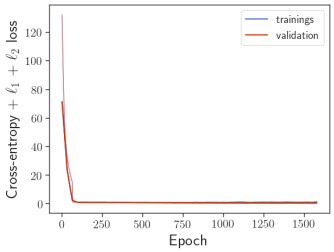
(c) *Detection model 2 classification loss*



(d) *Detection model 2 classification and regularization loss*



(e) *Detection model 3 classification loss*



(f) *Detection model 3 classification and regularization loss*

Figure D.2: Plots of the loss function during training of the three the models used to assess impact of regularization. Classification loss corresponds only to the cross-entropy loss, whereas the classification and regularization loss also includes the weight penalization.