University Institute of Lisbon

Department of Information Science and Technology

Pulmonary Nodule Segmentation in Computed Tomography with Deep Learning

João Henriques Oliveira Gomes

A Dissertation presented in partial fulfillment of the Requirements for the Degree of

Master in Computer Science

Supervisor:
PhD, João Pedro Afonso Oliveira da Silva, Assistant Professor,
ISCTE - University Institute of Lisbon

September 2017

Resumo

A deteção do cancro do pulmão numa fase inicial é essencial para o tratamento da doença. Sistemas de segmentação de nódulos pulmonares, usados em junção com sistemas de Deteção Assistida por Computador (DAC), podem ajudar médicos a diagnosticar e gerir o cancro do pulmão. Neste trabalho propomos um sistema de segmentação de nódulos pulmonares, recorrendo a técnicas de aprendizagem profunda. Aprendizagem profunda é um sub-campo de aprendizagem automática, responsável por vários resultados estado da arte em datasets de segmentação de imagem, como o PASCAL VOC 2012. O nosso modelo final é uma 3D U-Net modificada, treinada no dataset LIDC-IDRI, usando interseção sobre união como função de custo. Mostramos que o nosso modelo final funciona com vários tipos de nódulos pulmonares. O nosso modelo consegue resultados estado da arte no LIDC test set, usando nódulos anotados pelo menos por 3 radiologistas, com uma verdade consensual de 50%.

Palavras-chave: visão computacional, imagiologia médica, nódulos pulmonares, segmentação de imagem, aprendizagem profunda

Abstract

Early detection of lung cancer is essential for treating the disease. Lung nodule segmentation systems can be used together with Computer-Aided Detection (CAD) systems, and help doctors diagnose and manage lung cancer. In this work, we create a lung nodule segmentation system based on deep learning. Deep learning is a sub-field of machine learning responsible for state-of-the-art results in several segmentation datasets such as the PASCAL VOC 2012. Our model is a modified 3D U-Net, trained on the LIDC-IDRI dataset, using the intersection over union (IOU) loss function. We show our model works for multiple types of lung nodules. Our model achieves state-of-the-art performance on the LIDC test set, using nodules annotated by at least 3 radiologists and with a consensus truth of 50%.

Keywords: computer vision, medical imaging, pulmonary nodules, image segmentation, deep learning

Acknowledgements

This work was made possible by the support of many people.

First, I must thank my advisor, Prof. Dr. João Pedro Afonso Oliveira da Silva, for always being available when I needed him the most, and for providing valuable advice. Also, without my advisor, I wouldn't have access to the server where the foundation for this work was built upon. Thank you for setting everything up, and for your support.

Also, I'd like to thank ISCTE, the institution where I worked on my master's degree for the last two years. To the teachers and faculty, thank you for your professionalism, and for the respect you always treated me with.

A huge thank you to my parents, for always supporting me, always wanting what's best for me, and believing in me when no one else would.

This work wouldn't exist if not for the LIDC-IDRI dataset. I'd like to acknowledge the great work done by all the people involved in creating the dataset, and the Cancer Imaging Archive for making the dataset publicly available.

Finally, an honorable mention goes out to the Google Cloud's free tier program, for allowing us to train deep neural networks using advanced GPUs.

Contents

List of Tables

List of Figures

Dedicated to our future AI overlords

Chapter 1

Introduction

In this introductory chapter, we talk about lung nodule segmentation systems, and the role they play in fighting lung cancer. We briefly mention deep learning, and how it relates to image analysis systems. We discuss our objectives, and give a broad overview of this document.

1.1 Lung Cancer and Lung Nodules

Lung cancer is the leading cause of cancer deaths worldwide (World Health Organization, 2017). In the United States of America, lung cancer accounts for approximately 27% of cancer-related deaths (American Cancer Society, 2016). One of the reasons why this disease is so deadly is because symptoms only appear at an advanced, non-curable stage. Not only that, but radiologists may misdiagnose the disease by pointing out abnormalities that turn out not to be cancer (false positives). This may lead to additional tests such as more CT (Computed Tomography) scans, needle biopsies or even surgery (National Cancer Institute, 2016).

Despite all, there have been important breakthroughs in the fight against lung cancer. Screening trials, for example, have shown that lung cancer screening can help prevent fatalities associated with the disease. In particular, the National Lung Screening Trial (or NLST) showed that: "... participants who received low-dose helical CT scans had a 15 to 20 percent lower risk of dying from lung cancer than participants who received standard chest X-rays. This is equivalent to approximately three fewer deaths per 1,000 people screened in the CT group compared to the chest X-ray group over a period of about 7 years of observation (17.6 per 1,000 versus 20.7 per 1,000, respectively)" (National Cancer Institute, 2014). As a result of the NLST and later studies, the U.S. is currently implementing lung cancer screening programs using low-dose CT. Other countries will likely implement similar lung screening trials in the near future. There is, however, one

significant challenge associated with these trials: the massive amount of CT images needed to be analyzed by radiologists. And that's where Computer Aided Detection comes into play.

A Computer Aided Detection (CAD) system is a piece of software that receives medical images as input, identifies suspicious features on the image, and brings them to the attention of the physicians. CAD systems can potentially decrease observational oversights (e.g. reduce false negative rates), and, in general, facilitate the work of physicians who interpret medical images.

While the main goal of this dissertation is not to build a fully functional CAD system, a lung nodule segmentation system is a key component in CAD systems used for screening, diagnosing and managing lung cancer.

A lung nodule is a lesion in the lungs which potentially represents lung cancer. The diameter and volume of a nodule are both useful metrics for assessing its malignancy (de Hoop et al., 2010; Eisenhauer et al., 2009). Lung nodule segmentation systems help provide these measures.

Lung nodule segmentation systems traditionally rely on hand crafted lung features (a process known as "feature-engineering"). Features can be anything from the pixel intesity in a certain area, to measuring the sphericity of certain objects. Traditional segmentation systems also rely on hand written segmentation rules, such as: if a pixel is above a certain threshold, label it as "lung nodule pixel" (Farag, Munim, Graham, & Farag, 2013). However, our final system will not make use of either handcrafted features of segmentation rules. Our final model will be based on deep learning, which is a sub-field of machine learning centered around neural networks (LeCun, Bengio, & Hinton, 2015). Deep learning removes the necessity for handcrafted features and segmentation rules.

Deep learning systems achieve state-of-the-art results in several image analysis tasks, like image segmentation (Everingham et al., 2015). Deep learning systems are also known to outperform humans in some image classification tasks (He et al., 2015). A lung nodule segmentation system built on deep learning can potentially allow doctors evaluate more patients, in a more efficient and effective way (compared to traditional lung nodule segmentation systems).

1.2 Objectives

The main goal of this dissertation is to develop a pulmonary nodule segmentation system, based entirely on deep learning. A system for pulmonary nodule segmentation can be used for measuring whether a nodule is malignant or not. This is usually done in two ways: by measuring the diameter of a nodule, or by measuring the volume. Nodule

mass ($volume \times density$) can also be used (de Hoop et al., 2010). Volume and mass measurements require nodule segmentation masks. Diameter measurements can also be made using nodule segmentation masks.

The final segmentation system has to be fast, given that manual segmentation of a single nodule can take up to 10 minutes (de Hoop et al., 2010). If a CT scan has multiple nodules (as they often do), the segmentation process can take way more than just 10 minutes. Besides speed, the system has to be consistent, by producing the same results for the same scans. In other words, the final system should always produce the same output for a given input. Result consistency can be achieved with software based segmentation systems. However, in real life, result consistency can be an issue, since physicians tend to segment nodules differently (Revel et al., 2004).

Finally, and perhaps most importantly, the produced nodule segmentation masks have to be accurate. We will measure our system's accuracy in accordance with the literature on lung nodule segmentation (Messay et al., 2015):

- The intersection over union (IOU) metric will be used. The IOU metric is discussed in Section 2.2.1.
- Lung nodules from the original LIDC dataset (McNitt-Gray et al., 2007) will be used as test set.

1.3 Contributions

We build a lung nodule segmentation system based on deep learning techniques. Our deep learning model has a modified 3D U-Net architecture (Çiçek et al., 2016). The 3D U-Net is a deep learning architecture used to segment 3D images (more information on Chapter 3).

In the original 3D U-Net paper (Çiçek et al., 2016), the authors trained their model using a softmax with weighted cross-entropy loss function. Our model, on the other hand, was trained with the intersection over union (IOU) loss (Rahman & Wang, 2016) (more information on loss functions and performance metrics is available in Section 2.2.1). We show deep learning segmentation models can be trained with the IOU loss, and still achieve state-of-the-art performance.

The training, validation and test sets used in this work were all obtained with the pylidc library (Matthew C. Hancock, 2016). Using a 3rd party library to create our sets of data ensures future reproducibility, by providing a reliable and independent source of data. In other words, if someone wishes to recreate our results, they will be able to use the exact same data we used, thanks to the pylidc library. To our knowledge, this work is the first to allow this kind of reproducibility.

Finally, our system achieves state-of-the-art performance on the LIDC test set, using nodules annotated by at least 3 radiologists and with a consensus truth of 50%. More information on the performance of our model can be found in Section 3.4.

1.4 Document Structure

In Chapter 2, we discuss the theory on which our final segmentation system is based on. First, we provide an overview of techniques typically used in lung nodule segmentation. Then, we briefly discuss existing methods for performing image segmentation using deep learning. Finally, we combine the two, and give a broad overview of how deep learning can be used for lung nodule segmentation.

Chapter 3 provides a roadmap of all the experiments made, and detailed information on how to recreate them. We give information on how to pre-process the data, set up and train the deep learning model, among others. The final system's performance is discussed and a comparison is made with existing systems.

Finally, in Chapter 4 we conclude this dissertation by sharing our final thoughts, and discuss possible future work in this area.

Chapter 2

State of the Art

What is a lung nodule? This is not an easy question, and there are different (sometimes conflicting) definitions (Armato et al., 2009; van Ginneken et al., 2010). A nodule can be anything from "primary lung cancer, metastatic disease, a noncancerous process, or indeterminate in nature" (Armato et al., 2011). Sometimes, the term "lesion" may be used to denote a lung nodule. It is true that lung nodules are lesions. However, there are certain lung lesions that do not classify as nodules (also known as non-nodular lesions), such as: bronchiolitis lesions, pleural or fissural lesions, and apical scars (Armato et al., 2011). As an apart, the term lesion can also be misleading, as certain lesions like apical scars, despite being considered "lesions", can sometimes be harmless.

Considering all of the above, we will define a lung nodule as a lesion in the lungs which potentially represents lung cancer. However, determining if a lung nodule is cancerous is not a trivial task. A typical process involves the following two steps:

1. Nodule identification: after a lung CT (Computed Tomography) scan is taken, a physician examines it, and identifies possible nodules. Nodules are classified by their shape, pixel intensities, texture and size. Nodules can be classified, for example, as positive (cancerous), negative or indeterminate.

2. Nodule management: This is a very broad process, with extensive guidelines defined by several institutions, such as the Fleischner Society (MacMahon et al., 2017). After identifying nodules in the first CT scan, a follow-up CT scan is performed. The Fleischner Society proposes a follow-up scan in 3-12 months to measure the nodules' growth rate. For indeterminate nodules, if the follow-up CT scan shows an increase in diameter or volume, then there is a high chance the nodule is malignant.

The are two typical lung nodule measurements a physician can make: diameter and volume measurements. The current standard for measuring tumor response, Response

Evaluation Criteria in Solid Tumors (RECIST) (Eisenhauer et al., 2009), is based on measurements of the tumor diameter.

However, despite nodule diameter being the standard, nodule volume has been shown to be more reliable (de Hoop et al., 2010; Heuvelmans et al., 2013). In fact, the Nederlands-Leuvens Longkanker Screenings Onderzoek (NELSON) trial, the largest European lung cancer screening trial, relied on nodule volumetric measurements.

Nodule segmentation systems can be used to measure both nodule volume and diameter. These systems provide nodule segmentation masks with consistency and speed, making it easier for physicians to screen, diagnose and manage lung cancer. Consistent results are important, because in real life, physicians can show substantial inter-observer variability when measuring lung nodules (Revel et al., 2004). Speed is also crucial, because a physician can take up to 10 minutes do produce a nodule segmentation mask (de Hoop et al., 2010).

This chapter provides the theoretical background necessary to build a lung nodule segmentation system using deep learning. First, we give an overview on traditional lung nodule segmentation techniques, like region growing and morphology based segmentation. As we'll see, these techniques rely heavily on hand written segmentation rules. We'll explain why and how hand written rules can fail in some cases.

Finally, in this chapter, we provide some technical background on segmentation via deep learning. Some commonly used deep learning architectures are discussed, along with the theoretical foundation to fully understand them.

2.1 Lung Nodule Segmentation

Segmentation consists in dividing an image in multiple coherent parts (Ho, 2011). A "coherent part" can be anything from an image part that belongs to one specific object, a part with similar color, texture, etc. In machine learning based segmentation, there are two main tasks:

1. Semantic Segmentation — Classify every single pixel in an image.
2. Instance Segmentation — Identify instances of an object in an image, and segment each instance separately.

Figure 2.1 shows the difference between the two kinds of machine learning segmentation tasks. Most work in image segmentation is in the area of semantic segmentation. In fact, the terms "semantic segmentation" and "segmentation" are sometimes used interchangeably. This work will focus on semantic segmentation. The two pixel classes we'll be working with are: nodule and not nodule (or background).

(a) Object Detection (b) Semantic Segmentation (c) Instance Segmentation

Figure 2.1: Difference between object detection, semantic segmentation and instance segmentation. Object Detection is the task of finding bounding boxes, rectangles, circles, or other shapes that best fit certain objects of interest. Semantic segmentation consists of labeling every pixel in an image with a class (in this case, the label is "cat'). Instance segmentation, unlike semantic segmentation, provides two different segmentation masks, one for each cat instance. Original cat picture by payaovat (2017), distributed with the Pixabay License.

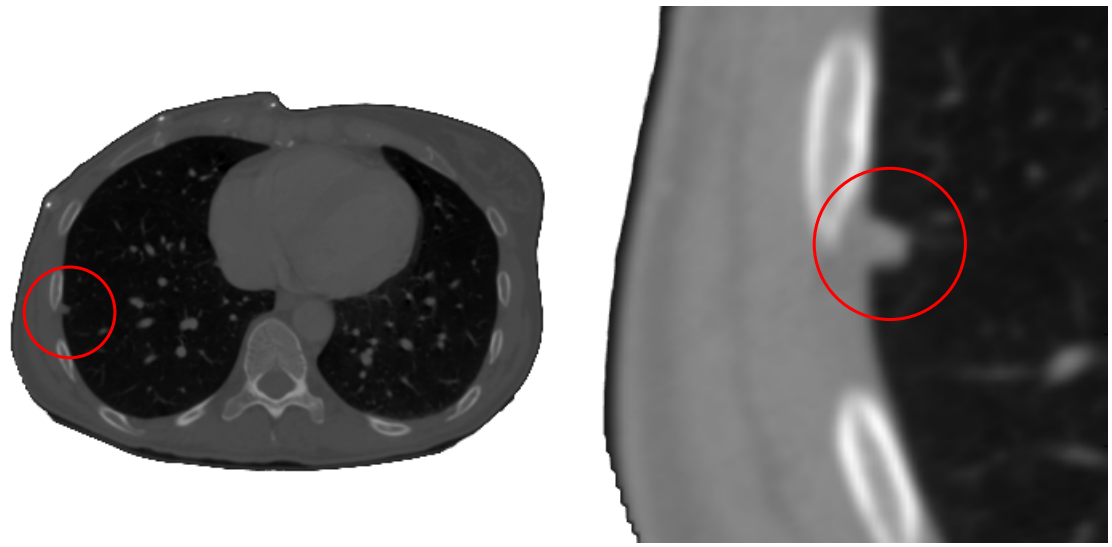
Segmentation should not be confused with detection. Detection (or object detection) consists in creating bounding boxes (sometimes bounding rectangles, circles, or other shapes) around objects of interest. Lung Nodule Detection is an area where deep learning has made significant contributions in the past two years. These contributions can be seen in the LUNA16 challenge, where the majority of solutions involve deep learning (Setio et al., 2017).

Historically, automatic lung nodule segmentation has been a difficult task. Mainly due to lung nodules' variability in shape, size, position inside the lungs, among others. The nodules that present the biggest challenge are:

1. Nodules attached to structures inside the lung, e.g. nodes attached to the walls of the lungs, also known as Juxta-Pleural Nodules, or JPN (Li et al., 2016). See Figure 2.2 for an example of a JPN.

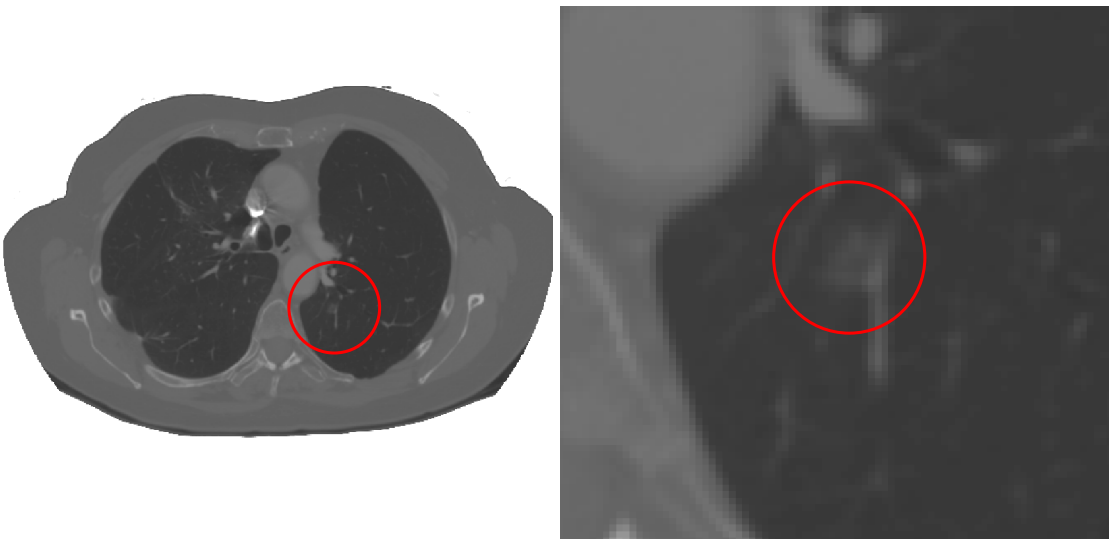
2. Subsolid nodules with heterogeneous texture and weak edges. There are two types of subsolid nodules: part-solid and non-solid nodules, also known as ground-glass opacity nodules, or GGON (Lassen, Jacobs, Kuhnigk, van Ginneken, & van Rikxoort, 2015). See Figure 2.3 for an example of a GGON.

Subsolid nodules, while being a challenge to both segment and detect, also tend to be more malignant compared to other nodules (Henschke et al., 2002). Developing a segmentation framework that works for all types of nodules is a challenging task (Farag et al., 2013). As such, literature on lung nodule segmentation can sometimes focus on just one type of nodules: subsolid nodules (Lassen et al., 2015), nodules attached to other



(a) Scan with a Juxta-Pleural Nodule (b) Zoomed in Juxta-Pleural Nodule

Figure 2.2: Example of a Juxta-Pleural Nodule (JPN). Scan taken from the LIDC-IDRI dataset (Armato et al., 2011).



(a) Scan with a Ground-Glass Opacity Nodule (b) Zoomed in Ground-Glass Opacity Nodule

Figure 2.3: Example of a Ground-Glass Opacity Nodule (GGON). The nodule we see here is almost invisible, but there were even more subtle GGON nodules in the LIDC-IDRI dataset. Scan taken from the LIDC-IDRI dataset (Armato et al., 2011).

lung structures (Diciotti, Lombardo, Falchini, Picozzi, & Mascalchi, 2011), etc. The deep learning approach presented in Section 2.2 is designed to work for all kinds of nodules.

Some of the traditional segmentation and detection methods require pre or post removal of certain lung structures like the chest wall, bones and vessels. This facilitates the segmentation/detection of the nodules. With deep learning, the removal of certain lung structures could help, but it is not strictly necessary.

Most segmentation methods are semi-automatic, requiring variable amount of user input. For example, traditional region growing (discussed in the next section) methods require the user to define an initial seed point. Other methods may require a greater amount of user input. There are also fully automatic methods, which require no user input. As an aside, the definitions of "semi-automatic" and "fully automatic" may vary from author to author. For example, in Messay et al. (2015), the described fully automatic segmentation method requires a "cue point" from the user.

In the next sections, we describe commonly used segmentation methods, like region growing and morphology based segmentation. We will discuss the strengths and weaknesses of these methods, and show why they may fail for certain types of nodules.

2.1.1 Region Growing

Region growing is a segmentation technique used not only in lung nodule segmentation, but in other segmentation tasks as well (Adams & Bischof, 1994). In its simplest form, region growing is a thresholding technique that involves the following two steps:

1. Start the segmentation mask in one point, also known as seed point. This point is usually chosen manually by the physician, but can also be chosen by a Computer Aided Detection system. This seed point is typically somewhere inside a lung nodule.
2. Grow the segmentation mask from the seed point in all directions. As the segmentation mask grows, only include new points if they are similar to the seed point. Pixel similarity can be defined in many ways, such as whether or not a pixel's intensity is bellow a given threshold.

When lung nodules are attached to lung structures (e.g. wall of the lungs), region growing can mistakenly select points belonging to said lung structures as part of the final segmentation mask. This failure happens when lung structures have similar pixel intensities to the attached lung nodules. An example of this phenomenon can be seen in Figure 2.4.

However, there are region growing methods that fix such problematic cases. In Dehmeshki, Amin, Valdivieso, and Ye (2008), the authors begin by roughly separat-

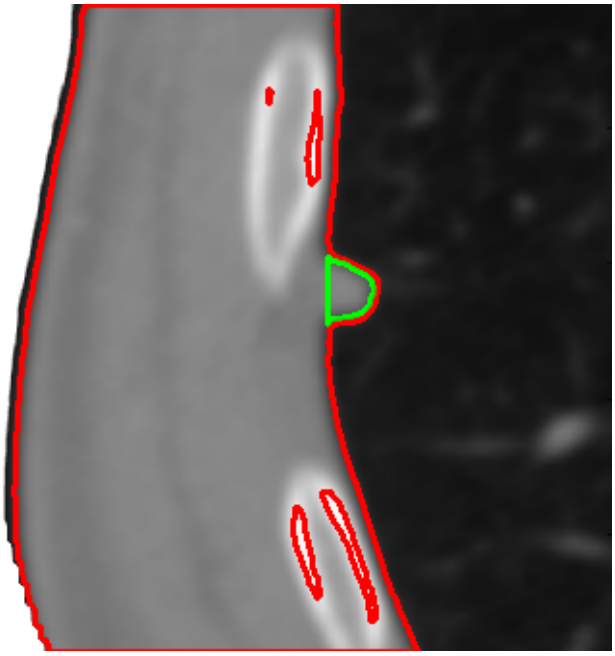


Figure 2.4: A Juxta-Pleural Nodule (JPN). Good segmentation (in green) compared to a naive, bad segmentation (in red). The bad segmentation may result from an improper implementation of the region growing segmentation algorithm. A regular version of this nodule can be seen in Figure 2.2. Scan taken from the LIDC-IDRI dataset (Armato et al., 2011).

ing foreground (parts of interest inside the lung) and background (non interesting parts of the lung). Then, they create a rough, initial segmentation mask by refining the foreground. Once the segmentation mask is created, each point is assigned a value that represents how likely they belong to the same object as the seed point. This is done for every point in the segmentation mask, and creates a so called fuzzy connectivity map. The authors then apply region growing on this map, and are able to correctly segment different types of lung nodules.

Region growing techniques can be performed both slice by slice (2D) and voxel by voxel (3D or voxel-wise). Other region growing based solutions for lung nodule segmentation include Diciotti et al. (2011); Kubota, Jerebko, Dewan, Salganicoff, and Krishnan (2011). These methods often combine region growing and morphology based segmentation, which we will discuss next.

2.1.2 Morphology Based

Morphology based segmentation makes prior assumptions on the shapes of the objects to be segmented (Diciotti et al., 2011; Kubota et al., 2011; Kuhnigk et al., 2006). For example, if we know nodules tend to have round shape, and the final segmentation mask

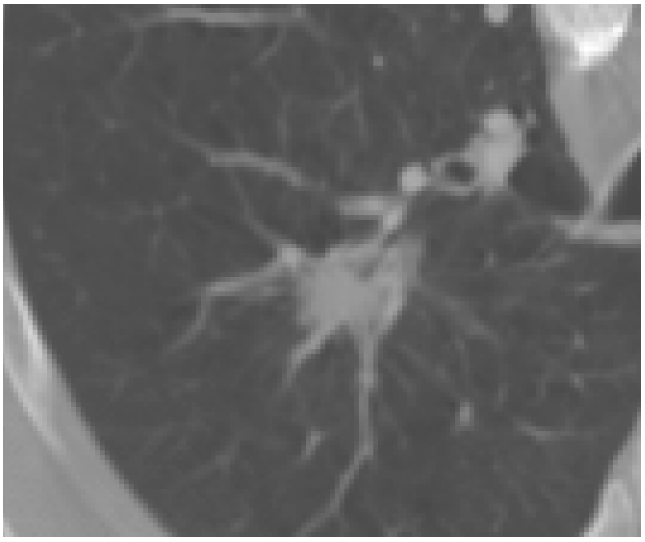


Figure 2.5: Example of a nodule connected to vascular structures, also known as Juxta-Vascular Nodule, or JVN. Scan taken from the LIDC-IDRI dataset (Armato et al., 2011)

.

has a square shape, then it may be a good idea to instead look for a rounder shape. These methods fall in the category of knowledge-based methods, given that they rely on domain specific knowledge.

In Kuhnigk et al. (2006), the authors begin by creating a rudimentary segmentation mask via region growing. In order remove the lungs from the mask, they assume the lungs are "mostly convex". A convex hull operation is then used to separate nodules from the lung wall. To accurately separate nodules connected to vascular structures (see Figure 2.5), the authors assume that the vascular structures are thinner than the nodules. Relying on this fact, the authors perform "morphological opening', which consists of erosion, removing the possibly attached "thinner" vascular structures, followed by dilation (Kuhnigk et al., 2006).

The main drawback of morphology based segmentation is the fact lung nodules have variable sizes, shapes, and degrees of connectivity to other lung structures. And so, assuming a pre-defined shape for all lung nodules is a strategy that will fail sooner or later, as nodules with shapes that are beyond our imagination may appear in real life.

2.1.3 Other Methods

There are many different ways to segment an image. A comprehensive list of available segmentation methods can be found in Pei-Gee Ho (Ho, 2011).

Besides the lung nodule segmentation methods described so far, many more have been proposed (Diciotti et al., 2011). There are two methods that, while not as common as

region growing or morphology based segmentation, are sometimes used in lung nodule segmentation literature: Active Contour Models (ACM), and feature based methods.

Active Contour Models

An Active Contour Model (also known as ACM or Snakes) is a framework that can be used for image segmentation. ACMs make use of deformable splines that are adjusted to shapes in an image. ACMs rely on energy minimization techniques to find the best fit between deformable splines and shapes inside an image. Active Contour Models can be used for multiple image segmentation tasks, including lung nodule segmentation (Li et al., 2016; Way et al., 2006).

Feature based methods

Feature based methods rely on the extraction of features from CT scans or existing segmentation masks. These methods are typically used to refine the final segmentation results. Features like segmentation volume and pixel intensity information are extracted, and fed into a previously trained machine learning model. The model then outputs the probability of a given segmentation mask being a lung nodule (true positive) or not (false positive).

One downside of this approach is the fact that the features need to be previously defined by a human, usually in a rule-based fashion (Messay et al., 2015). Most of these methods rely on domain specific knowledge in order to extract the best possible features. Also, the more features the machine learning model receives as input, the longer it will take to execute, as feature extraction can be computationally expensive (Campos, Simões, Ramos, & Campilho, 2014; Messay et al., 2015).

2.2 Deep Learning

Deep learning is a subset of machine learning, focused mainly on training multilayered neural networks (LeCun et al., 2015). To be precise, a deep learning model can be any machine learning model composed of multiple processing layers, and able to learn representations of data with multiple levels of abstraction. A multilayer neural network is a machine learning model that achieves both: it is composed of multiple layers, and is able to learn representations of data with multiple levels of abstraction. This means multilayered neural networks are fed with raw data, and automatically discover the representations needed for image classification and object detection. For example, when using a neural net to identify cats in images, the first neural net layers may learn low level representations like small edges at particular orientations and locations in the image. Layers higher up

in the neural net may learn more abstract representations, like cat eyes or cat ears. These features (small edges, cat eyes, etc.) are not designed by humans. Rather, neural networks are able to learn features from data using general-purpose learning techniques (LeCun et al., 2015). This is called Representation Learning, or Feature Learning, and is one of the main ideas behind deep learning. Deep learning can be applied to multiple tasks, including the task at hand: lung nodule segmentation.

Deep learning methods have produced state-of-the-art results in applications such as speech recognition (Hinton et al., 2012) and machine translation (Sutskever, Vinyals, & Le, 2014). They have been shown to outperform humans in certain tasks like image classification (He et al., 2015). Recently, a deep learning approach was used to beat one of the top players in Go, a task many thought to be impossible (Silver et al., 2016). All of these deep learning success stories have neural networks at their core. Specifically, they involve neural network architectures with many layers, also known as deep neural networks.

Note that not all multilayer neural networks are considered deep neural networks. A neural network can be considered "shallow" or "deep", depending on the number of layers it has. According to Juergen Schmidhuber, there is no consensus on the number of layers necessary for a neural network to be considered "deep" (Schmidhuber, 2014). However, some researchers consider a neural network with 3 or more hidden layers to be a deep neural network. A hidden layer is any layer located between the input layer and output layer of a neural network.

In theory, deep learning models don't need to rely on neural networks, as long as they have multiple processing layers and achieve representation learning (LeCun et al., 2015). However, in practice, the majority of deep learning research and applications rely on neural networks. This work is no exception.

2.2.1 Problem Formulation

Before we go further into neural networks and deep learning, we need to formalize some basic concepts. Let $\mathbf{X}$ denote a 3D matrix with dimensions $40 \times 40 \times 40$, where each element inside the matrix represents a $1mm \times 1mm \times 1mm$ voxel (or 3D pixel). In our case, the $\mathbf{X}$ matrix represents a region of interest (ROI) inside a CT scan, containing a lung nodule.

Let f be a function that maps $\mathbf{X}$ to $\hat{\mathbf{Y}}$,

$$f : \mathbf{X} \rightarrow \hat{\mathbf{Y}}, \tag{2.1}$$

where $\hat{\mathbf{Y}}$ is a $40 \times 40 \times 40$ with $1mm \times 1mm \times 1mm$ voxel elements (same dimensions as $\mathbf{X}$). A voxel is a 3-dimensional pixel. The terms pixel and voxel are used interchangeably

in this document.

We will call each element in the output matrix $\hat{\mathbf{Y}}$ a score, and f will be called a score function. For the task at hand (lung nodule segmentation), a $\hat{\mathbf{Y}}$ score in position (x,y,z) represents how likely a voxel from $\mathbf{X}$ in the same position (x,y,z) belongs to a certain class. For the lung nodule segmentation task, two classes are used: 0 (non nodule pixel) and 1 (nodule pixel).

Let the score function f be a linear classifier, given by

$$f(\mathbf{X}) = \mathbf{MX} + \mathbf{B}, \tag{2.2}$$

where $\mathbf{M}$ is a weight matrix (i.e. the "slope" in a linear function), and $\mathbf{B}$ is a bias matrix (i.e. the "intercept" in a linear function). Linear classifiers play a central role in neural networks, as we'll see in Section 2.2.2.

For simplicity, we will merge the matrices $\mathbf{B}$ and $\mathbf{M}$ into a new matrix $\mathbf{W}$, according to

$$\mathbf{MX} + \mathbf{B} = \underbrace{\begin{bmatrix}\mathbf{M} & \mathbf{B}\end{bmatrix}}_{\mathbf{W}} \cdot \begin{bmatrix}\mathbf{X} \\ \mathbf{I}\end{bmatrix}, \tag{2.3}$$

where $\mathbf{I}$ is the identity matrix.

The goal of this work is to find a score function f that produces the best possible segmentation score matrices, $\hat{\mathbf{Y}}$. In order to measure the fitness of a particular $\hat{\mathbf{Y}}$, we need to first choose a performance metric.

Image Segmentation Metrics

Metrics measure the performance of our score function f. For an output matrix $\hat{\mathbf{Y}}_i$, if there exists a truth matrix $\mathbf{Y}_i$ with the desired score values, we can compare the two with

$$\frac{\sum_{p=1}^{P} [\hat{\mathbf{Y}}_{i,p} = \mathbf{Y}_{i,p}]}{P}, \tag{2.4}$$

where the statement inside the brackets $[...]$ is evaluated, resulting in 1 if the statement is true, or 0 otherwise. This notation if called the Iverson bracket notation. P is the total number of elements in $\mathbf{Y}$ (and, consequently, $\hat{\mathbf{Y}}$, given both matrices have the same shape), and p represents a matrix element in a specific position (x,y,z). The sum should include all p positions from each matrix. The metric in Equation 2.4 is called Pixel Accuracy. This metric counts the number of pixels score function f scored correctly, divided by the total number of pixels in matrix $\mathbf{Y}$ (or $\hat{\mathbf{Y}}$).

Another metric used in segmentation tasks is the dice coefficient, given by

$$Dice(\hat{\mathbf{Y}}_i, \mathbf{Y}_i) = \frac{2|\hat{\mathbf{Y}}_i \cap \mathbf{Y}_i|}{|\hat{\mathbf{Y}}_i| + |\mathbf{Y}_i|} = \frac{2|\hat{\mathbf{Y}}_i \cdot \mathbf{Y}_i|}{|\hat{\mathbf{Y}}_i| + |\mathbf{Y}_i|}, \tag{2.5}$$

where $|\mathbf{Y}|$ represents the sum of all the elements in matrix $\mathbf{Y}$, and $\hat{\mathbf{Y}}_i \cap \mathbf{Y}_i$ is the intersection of matrices $\mathbf{Y}_i$ and $\hat{\mathbf{Y}}_i$. Matrix intersection can be represented by the entrywise product of the matrices, $\hat{\mathbf{Y}}_i \cdot \mathbf{Y}_i$, also known as the Hadamard or Schur product.

Closely related to the dice coefficient is the intersection over union (IOU) metric, given by

$$IOU(\hat{\mathbf{Y}}_i, \mathbf{Y}_i) = \frac{|\hat{\mathbf{Y}}_i \cap \mathbf{Y}_i|}{|\hat{\mathbf{Y}}_i \cup \mathbf{Y}_i|} = \frac{|\hat{\mathbf{Y}}_i \cdot \mathbf{Y}_i|}{|\hat{\mathbf{Y}}_i| + |\mathbf{Y}_i| - |\hat{\mathbf{Y}}_i \cdot \mathbf{Y}_i|} \,. \tag{2.6}$$

The IOU metric (also known as Overlap) is perhaps the most commonly used metric in lung nodule segmentation literature (Messay et al., 2015).

The 3 metrics we've seen so far compared a single output matrix $\hat{\mathbf{Y}}$ with a single ground truth matrix $\mathbf{Y}$. In practice however, if we want an accurate measurement of the performance of our score function f, we average the metric function's results over N different $\hat{\mathbf{Y}}$ and $\mathbf{Y}$ matrices, where N is the total number of examples in our dataset.

Our ultimate goal is to develop a score function f that produces accurate $\hat{\mathbf{Y}}$ matrices, according to a previously defined metric. In order to find the best possible f, we will use supervised learning. Supervised learning consists in utilizing a set of data (called training data) to infer the best possibel score function f. The training data is of the form $\{(\mathbf{X}_1, \mathbf{Y}_1), (\mathbf{X}_2, \mathbf{Y}_2), ..., (\mathbf{X}_N, \mathbf{Y}_N)\}$. Each pair $(\mathbf{X}_i, \mathbf{Y}_i)$ is called a training example, where $\mathbf{X}_i$ is the input to f (in our case, a CT scan region), and $\mathbf{Y}_i$ is the matching ground truth matrix (in our case, the correct segmentation mask). During the training process, the following steps are repeated multiple times:

1. An input image $\mathbf{X}_i$ is given as input to f, which produces an output matrix $\hat{\mathbf{Y}}_i$.
2. The produced $\hat{\mathbf{Y}}_i$ is compared to the ground truth matrix, $\mathbf{Y}_i$. If these matrices are different, a penalty is made. Usually, the greater the difference between $\hat{\mathbf{Y}}_i$ and $\mathbf{Y}_i$, the greater the penalty.

This penalty is given by a function L, called loss function. Thus, our goal is to find a score function f such that, for a chosen loss function L, the penalties are the lowest.

Loss Functions

The purpose of a loss function is to penalize wrong outputs from our score function f. Typically, the output of a loss function is called an error, because it calculates some sort of distance between $\hat{\mathbf{Y}}$ and $\mathbf{Y}$. If $\hat{\mathbf{Y}}$ is very similar to $\mathbf{Y}$, the loss function will output a value close to 0 (small error). If the loss function's result is a large value, the error is big and $\hat{\mathbf{Y}}$ is most likely very different from $\mathbf{Y}$. A popular loss function is the squared error loss (or

quadratic loss), given by

$$L = \frac{1}{2N} \sum_{i=1}^{N} \sum_{p=1}^{P} (\hat{\mathbf{Y}}_{i,p} - \mathbf{Y}_{i,p})^2, \tag{2.7}$$

where N is the total number of images in the training set, and P is the total number of elements in $\mathbf{Y}_i$ (and $\hat{\mathbf{Y}}_\mathbf{i}$, since both these matrices have the same dimensions). The letter p represents a matrix element in a specific position (x, y, z), and the inner sum $\sum_{p=1}^{P}(...)$ goes through every element in the 3D matrices. The i in $\hat{\mathbf{Y}}_i$ indicates that the output score matrix was obtained by passing training example $\mathbf{X}_i$ through our score function f. In other words, $f(\mathbf{X}_i) = \hat{\mathbf{Y}}_i$. This loss function takes into account all N training examples, hence the outer sum $\sum_{i=1}^{N}(...)$. As an example, $\hat{\mathbf{Y}}_{2,4}$ can be read as "single value in position 4 inside the output 3D matrix $\hat{\mathbf{Y}}_2$".

In segmentation problems, a commonly used loss function is the Dice Coefficient loss function. For a given training example $\mathbf{X}_i$, the loss is given by

$$L_i = 1 - Dice(\hat{\mathbf{Y}}_i, \mathbf{Y}_i), \tag{2.8}$$

where $Dice(\hat{\mathbf{Y}}_i, \mathbf{Y}_i)$ is the Dice Coefficient.

Another perhaps less commonly used loss function, is the IOU loss function (Rahman & Wang, 2016),

$$L_i = 1 - IOU(\hat{\mathbf{Y}}_i, \mathbf{Y}_i), \tag{2.9}$$

where IOU is the intersection over union (also known as Jaccard Index), given by Equation 2.6.

L_2 Regularization

Machine learning models with a high amount of weights (e.g. deep neural networks) can overfit. In a nutshell, overfitting occurs when a machine learning model has good performance on the training set, but fails to generalize to data outside the training set (Hawkins, 2004). L_2 regularization is a technique used to prevent overfitting and reduce the search space for our score function's weights, i.e., the values of matrix $\mathbf{W}$ (Krogh & Hertz, 1992).

If we consider the loss of a single training example $\mathbf{X}_i$ to be represented by L_i, the final loss function with regularization is given by

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + \lambda \sum_{k=1}^{K} \mathbf{W}_k^2 \,, \tag{2.10}$$

where $\sum_{k=1}^{K} \mathbf{W}_k^2$ is the L_2 regularization term, λ is the regularization parameter, and K is the number of entries in the weight matrix $\mathbf{W}$.

Joining equations 2.6, 2.9 and 2.10, our loss function will be

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + \lambda \sum_{k=1}^{K} \mathbf{W}_k^2 \tag{2.11}$$

$$= \frac{1}{N}\sum_{i=1}^{N} (1 - IOU(\hat{\mathbf{Y}}_i, \mathbf{Y}_i)) + \lambda \sum_{k=1}^{K} \mathbf{W}_k^2 \tag{2.12}$$

$$= \frac{1}{N}\sum_{i=1}^{N} \left(1 - \frac{|\hat{\mathbf{Y}}_i \cdot \mathbf{Y}_i|}{|\hat{\mathbf{Y}}_i| + |\mathbf{Y}_i| - |\hat{\mathbf{Y}}_i \cdot \mathbf{Y}_i|}\right) + \lambda \sum_{k=1}^{K} \mathbf{W}_k^2 \,, \tag{2.13}$$

where $\hat{\mathbf{Y}}_i$ is the output of our score function f, $\mathbf{Y}_i$ is the real segmentation mask, $\mathbf{W}_k$ is element k of the score function's weight matrix, and N is the number of training examples.

Optimization

The goal of training is to find the weight matrix $\mathbf{W}$ that will make our score function f produce the most accurate segmentation scores possible. This can be accomplished by solving the optimization problem

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} L(\mathbf{X}, \mathbf{Y}, \mathbf{W}), \tag{2.14}$$

where L is the loss function given by Equation 2.13.

One way of solving the optimization problem described in Equation 2.14 is by using a method called gradient descent (Bottou, 2010). Gradient descent consists in updating the weights $\mathbf{W}$ in the opposite direction of the gradient $\nabla L(\mathbf{W})$. A simple weight update rule based on gradient descent is, for example,

$$\mathbf{W} := \mathbf{W} - \alpha \nabla L(\mathbf{W}) \,, \tag{2.15}$$

where $:=$ is the assignment operator, and α is the learning rate, a key hyperparameter in machine learning. A hyperparameter is a value that is set before the training process begins.

For our work, we'll use the Adam (Kingma & Ba, 2014) update rule, which is a bit more complex than the one in 2.15. The Adam update rule combines different optimization techniques like first and second moments of the gradients, and is available in multiple deep learning frameworks.

After training, our score function f should have weights $\mathbf{W}$ that produce the lowest possible loss function output values. In other words, f should be able to receive any input 3D CT Scan region $\mathbf{X}$, and produce a score matrix $\hat{\mathbf{Y}}$ such that the error produced by our loss function is minimal.

In our case, since the final segmentation masks have binary values, the score function's output ($\hat{\mathbf{Y}}$) values must be converted to binary values. This is done by applying a threshold t on the output of f, $\hat{\mathbf{Y}}$.

2.2.2 Neural Networks

Neural networks (artificial neural networks, to be precise) are machine learning models made up of units called neurons (or artificial neurons). These neurons perform a linear multiplication of their inputs followed by a non-linear, element-wise operation, performed by a function called activation function.

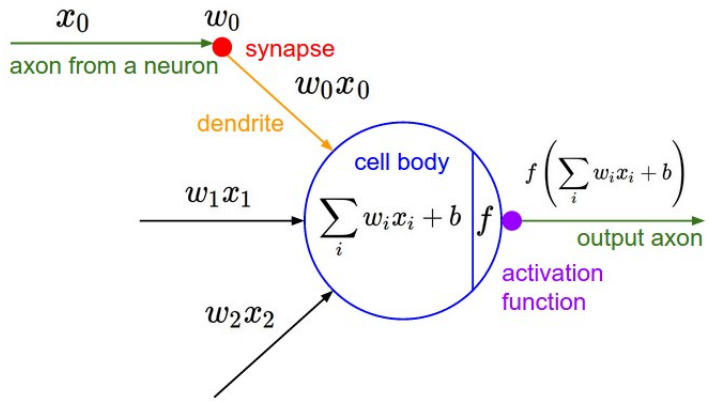


Figure 2.6: An artificial neuron. The neuron's inputs x are multiplied by the parameters, or weights, w. The cell body sums up the multiplication results, takes the sum result and passes it through an activation function. Image taken from the PhD thesis of Andrej Karpathy (Karpathy, 2016).

Neurons are organized across layers in a neural network, according to Figure 2.7. Neurons in one layer have connections to all neurons in the previous layer but are not connected to neurons within the same layer.

As we can see in Figure 2.6, a neuron computes a composite function. This composite function is a linear function followed by a non-linear function,

$$h(x) = f(\sum_i w_i g_i(x)), \tag{2.16}$$

where w_i is a single weight, $g_i(x)$ is the output of the previous layer, and f is an activation function (i.e. the non-linear part in $h(x)$).

Activation Functions

An activation function (also known as non-linearity) is responsible for the non-linear part of a neural network. They are used to guarantee neural networks are able to learn non-linear patterns in the data. If a neural network didn't have non-linearities, it would just be a composition of linear functions, i.e. a linear function. Two examples of non-linearities are the Rectified Linear Unit (ReLU),

$$f(x) = max(0,x), \tag{2.17}$$

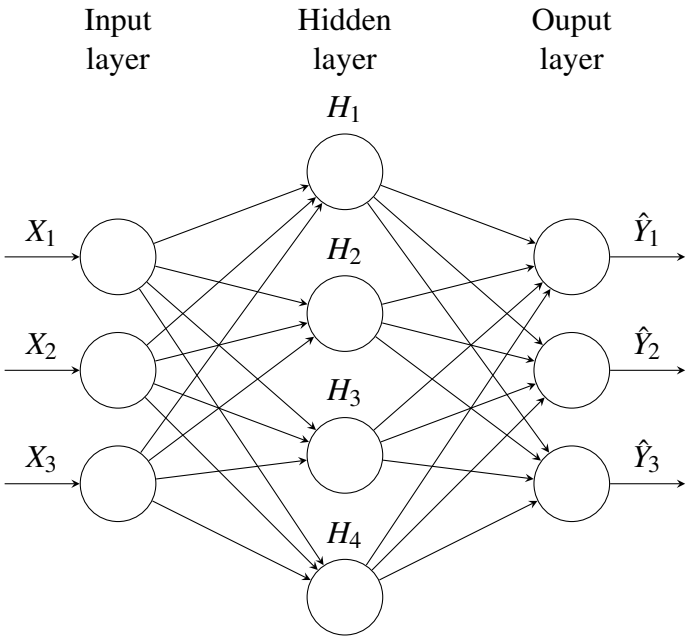
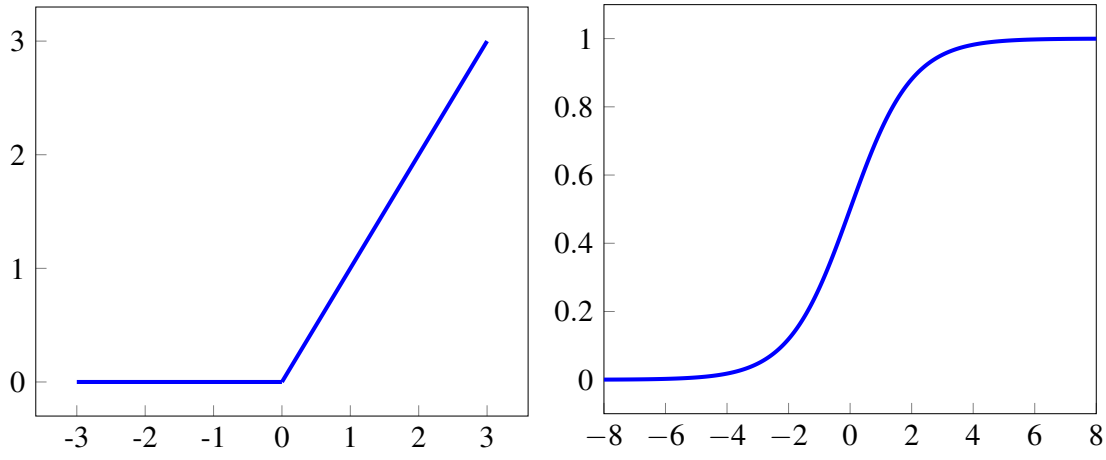


Figure 2.7: A neural network with two layers. The input layer doesn't count as a neural network layer, only the single hidden layer and output layer count. There are three inputs. The inputs, in our case, are pixels from CT scans, which will be far more than just three values. The hidden layer has four artificial neurons. The artificial neurons are the same as in Figure 2.6. Finally, we have an output layer, which produces our final scores.

commonly used in recent years, and the sigmoid function,

$$f(x) = \frac{1}{1+e^{-x}}\,. \tag{2.18}$$

Both activation functions can be visualized in Figure 2.8.



(a) ReLU (b) Sigmoid

Figure 2.8: Two examples of activation functions: the ReLU function, and the Sigmoid function.

Optimization Techniques for Neural Networks

To properly adjust the weights in a neural network (and in other score functions), we need to compute the gradients of the loss function with respect to the weights. However, unlike most score functions, a neural network is a composite function, with an arbitrary number of composite inner functions (i.e. layers). In order to find the gradients of a composite function, we can use the chain rule,

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial w_i}, \tag{2.19}$$

where f is our score function (i.e. the whole neural network), w_i is a single neural net weight, and L is the loss function.

Neural networks use a method similar to the chain rule, called backpropagation. In a nutshell, backpropagation consists in propagating gradients throughout the neural net, starting from the output at the top (where the neural net produces its scores) all the way to the bottom (where the neural network receives its external inputs, e.g. a lung nodule 3D image). The process of applying backpropagation to find the gradients is also called "backward pass" (LeCun et al., 2015).

Once we have gradients from all the layers (thanks to backpropagation), we can update our weights using, for example, the weight update rule in Equation 2.15. During the training process, multiple weight updates are performed. The weight updates should steer the neural network's weights towards producing the best possible outputs (LeCun et al., 2015).

Training a Neural Network

Overall, there are two main processes that are repeated many times throughout the training of a neural network:

1. Feeding an input image to the neural network and computing its corresponding scores. This is called a "forward pass".
2. Finding the gradients of the loss function with respect to all the weights in the neural network, via backpropagation. Also known as a "backward pass".

An epoch is a concept that frequently shows up in deep learning literature. During training, we say "one epoch has passed" when the neural network has processed all the training examples once. In other words, "one epoch has passed" when one forward and backward pass of all the training data has been performed. Training usually requires the neural networks to process all the training examples multiple times, so epochs are used to define how much the neural nets are trained.

To recap and give a broad overview of the whole process:

- The machine learning model we'll use is called a neural network, which is a composition of linear and non-linear mathematical functions. The final model should receive a 3D image (CT Scan region that contains a lung nodule) and produce an accurate lung nodule segmentation score matrix. This will be treated as a supervised learning problem, so we need a dataset consisting of two things: lung nodules, and their corresponding segmentation masks.
- Before training begins, the parameters (or weights) of the neural network are initially set to pseudo-random values. At this point, the neural network is able to receive any 3D image and produce a segmentation mask. However, the produced segmentation masks are likely useless, given that the neural network hasn't been trained yet.
- Forward pass: Images are given as input to the neural network, and the final scores are calculated. The scores are the neural network's output, and in our case, can be converted into segmentation masks. During training the scores are passed through a loss function, responsible for computing the difference between the output scores and real segmentation masks.
- Backward pass: Gradients are computed for all the weights in the neural network. The training process solves an optimization problem, where the goal is to minimize the loss function. The minimization methods used in deep learning rely on first finding the gradients of the loss function with respect to the model's weights. These gradients are calculated using a technique called backpropagation. Once we have the gradients, we can update our model's weights.

Once the training process has finished, our neural network model should be able to look at pixel intensities from an input 3D lung scan region, and produce the best possible lung nodule segmentation score matrix, which is then thresholded into a real segmentation mask.

2.2.3 Convolutional Neural Networks

A convolutional neural network (CNN) is a type of neural network in which the connectivity pattern between its neurons is inspired by the visual cortex of animals. Traditional CNNs are built using two main building blocks: convolutional layers and pooling layers (LeCun et al., 2015). They often appear alternating in a CNN. However, two or more convolutional layers may appear together in the architecture.

Convolution layers perform n convolution operations on their input $\mathbf{X}$, where n is the number of convolution filters. A tradition convolution operation outputs a feature map (M in Figure 2.9), which is the results of convolving a filter all over the convolution layer's

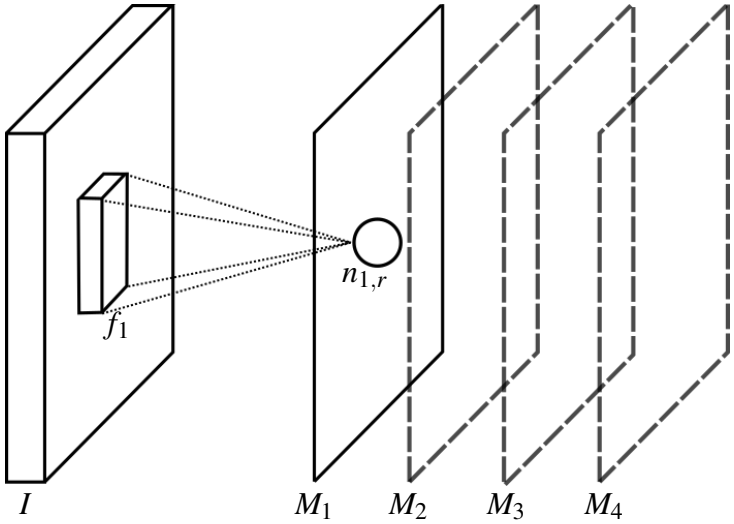


Figure 2.9: Convolution operation. The inner box f_1 is a convolutional filter, being applied to the input image I at a certain position. The result of applying filter f_1 to region r is the neuron $n_{1,r}$. If we convolve filter f_1 on the whole input image I, we get M_1, also known as a feature map (or just map, hence the M). The other feature maps (M_2, M_3, M_4) are the result of convolving other filters (f_2, f_3, f_4) which are not shown in this Figure. A convolution layer is defined by a set of filters ($f_1, f_2, f_3, ..., f_i$), also known as a filter bank. The output of a convolution layer is thus a volume of feature maps, $[M_1, M_2, M_3, ..., M_i]$, also known as an output volume.

input $\mathbf{X}$. For a given input $\mathbf{X}$ region, with values $[x_1, x_2, ..., x_9]$, if we apply a convolution filter f_i on that region, we will get a scalar value $n_{i,r}$,

$$n_{i,r} = \sum \left(\overbrace{\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix}}^{\text{filter } f_i} \cdot \overbrace{\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix}}^{\text{region } r} \right) \tag{2.20}$$

$$= (w_1 \times x_1) + (w_2 \times x_2) + (w_3 \times x_3) + ... + (w_9 \times x_9), \tag{2.21}$$

where $\cdot$ denotes the entrywise multiplication operation, i is the filter number (because convolutional layers can have multiple filters), and r is the region in $\mathbf{X}$ where the filter is applied. In this case, region r has values $[x_1, x_2, ..., x_9]$. Applying a convolution filter to an image $\mathbf{X}$ consists in repeating the operation in Equation 2.21 multiple times for multiple regions r in $\mathbf{X}$. The result is a feature map, which is a multidimensional array of $n_{i,r}$ values, e.g. $[n_{i,1}, n_{i,2}, ..., n_{i,finalr}]$. A visual representation of a convolution operation can be seen in Figure 2.9.

The other basic building block of a CNN is the max-pooling layer, responsible for downsampling feature maps along a CNN. A visual explanation of the max-pooling operation is given in Figure 2.10. The pooling layers are responsible for providing local

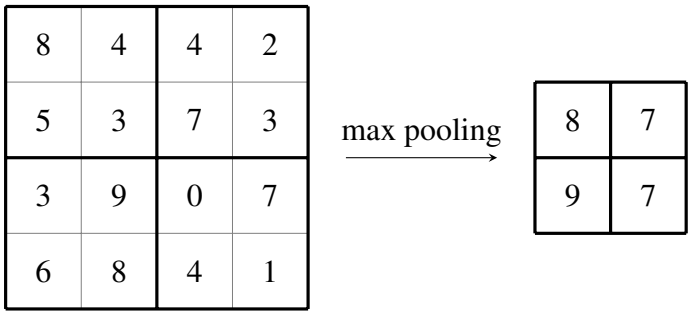


Figure 2.10: Example of a pooling operation. This particular type of pooling is called max-pooling. It looks at different regions of the input matrix (matrix on the left), and calculates their max. In this particular case, the max pooling operation has a window size of 2×2, and a stride of 2. There are other types of pooling, but pooling is usually associated with downsampling.

translation invariance (Jaderberg, Simonyan, Zisserman, & Kavukcuoglu, 2015). This is achieved by combining the outputs of the convolutional layers in a way that is position-independent, but only within a certain spatial location.

The last layers of a CNN are usually fully connected layers. These layers are traditional artificial neural networks (as seen in Figure 2.7), and they are responsible for outputting the final score vector. If we want to reshape the output vector of a CNN (i.e. the CNN's final output scores), we can leave the rest of the CNN intact and just change the number of neurons in the fully connected layers. A CNN architecture with fully connected layers can be seen in Figure 2.11.

A recent trend in deep learning is to remove pooling layers and/or fully connected layers. This allows convolutional neural networks to be built with only convolutional layers (Springenberg, Dosovitskiy, Brox, & Riedmiller, 2014).

CNNs have seen widespread use for computer vision tasks. They owe their success largely to their scalability: the convolution operation allows for the same filter (small set of weights) to be applied multiple times across a single image (see Figure 2.9 and Equation 2.21). This allows the model to be used much more effectively, compared to typical fully connected neural networks, which would require 1 weight per pixel or image area.

Recent CNN architectures can have 10 to 20 convolution layers, hundreds of millions of weights, and billions of connections between units. Training such large networks could have taken weeks only two years ago, but progress in hardware (specifically GPUs), software and parallelization techniques have reduced training times significantly. Today, some deep learning models are trained in less than one day (LeCun et al., 2015).

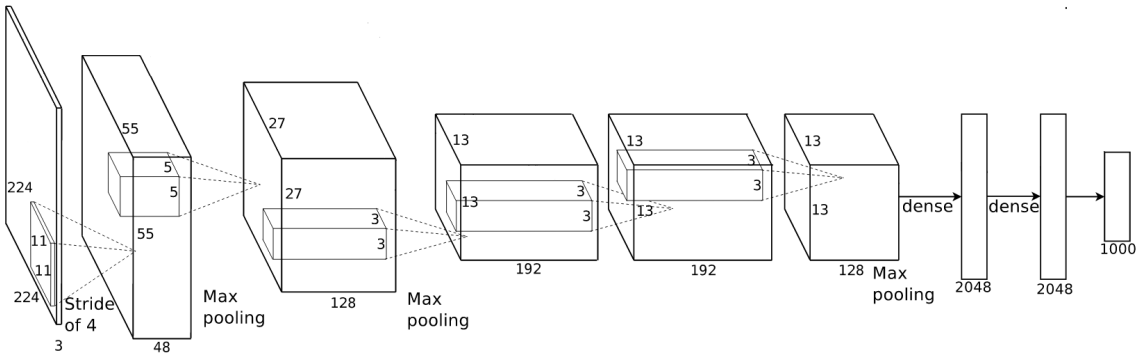


Figure 2.11: Example of a convolutional neural network architecture. Typical CNNs are made up of convolutional layers, followed by pooling layers. At the end, the dense layers (also known as fully connected layers) may be inserted. Dense layers are simple neural networks, as seen in Figure 2.7. Original image taken from Krizhevsky et al. (2012).

Invariance to Image Transformations

Filters in a CNN consist of learnable weights (as in Equation 2.21). They learn to detect particular patterns at multiple locations in the input images. These patterns often occur in different orientations and scales in real life images. As a result, CNNs will often learn multiple copies of the same filter in different orientations and scales (see Figure 2.12). However, despite CNN's ability to learn features at different orientations, CNNs are not invariant to some transformations, like image rotations (Worrall, Garbin, Turmukhambetov, & Brostow, 2016).

Not being rotation invariant means CNNs may classify the same image differently if the image is rotated. For some tasks, this makes perfect sense (e.g. people don't walk on walls), but for the task at hand (lung nodule segmentation in computed tomography scans) and other medical applications, rotation invariance is essential. In our case, a rotation invariant model is desirable because we don't care about the orientation of a lung nodule (i.e. whether the nodule appears at 30 or 60 degrees). We just care about the presence of the nodule, and whether a pixel belongs to a nodule or not.

The simplest way to achieve approximate invariance to rotation transformations (among other types of transformations) is to train a neural network with data augmentation. If, during training, the training examples are randomly perturbed with transformations, the neural network can learn to become (approximately) invariant to that particular transformation. Data augmentation via image rotation can be useful not only if the data exhibits rotational symmetry, but also in domains where professionally labeled datasets are scarce (e.g. medical imaging).

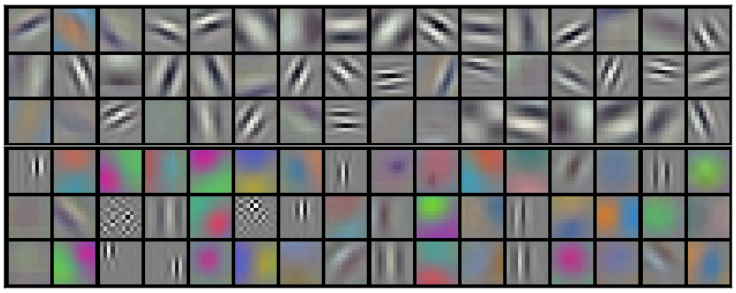


Figure 2.12: An example of the features learned by the filters in the first convolutional layer of a CNN. The first layer filters usually learn low-level features, like edges at different orientations and scales. The layers higher up in the architecture learn higher-level features (e.g. eyes, noses, etc, if the neural net was trained to detect human faces). Image taken from Krizhevsky et al. (2012).

Residual Neural Networks

He et al. (2015) achieved state-of-the-art performance in the ImageNet competition, obtaining a top 5 error of 3.57% - lower than 5.1%, which is what humans get (Russakovsky et al., 2015). These results were obtained using a deep learning architecture they named residual neural network.

He et al. (2015) showed that increasing the number of layers in a convolutional neural network doesn't necessarily increase performance. In fact, an increase in layers can hurt performance, not only on the test data, but on the training data as well.

In theory, a model with many layers (e.g. 100 layers) should be able mimic the performance of a model with less layers (e.g. 10 layers), without losing performance. This could be done by adding identity layers (i.e. layers that don't modify the input, or "dummy" layers) to the larger model, forcing the larger model to behave like the smaller one. Residual neural network models can learn to do something similar if necessary, by introducing "shortcut connections" between layers.

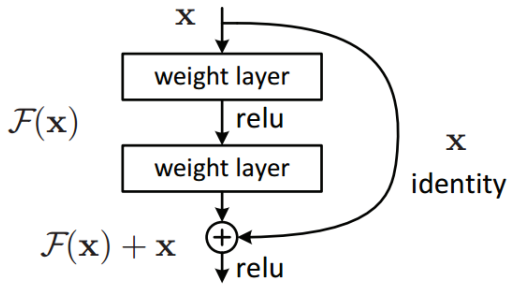


Figure 2.13: Image taken from He et al. (2015). $\mathcal{F}(x)$ represents the output of the two weight layers. These weight layers are layers with learnable weights, like typical convolution layers. Residual neural networks add a shortcut connection between the input of these layers, x, and their output.

Shortcut connections, as seen in Figure 2.13, are the main idea behind residual neural networks. These shortcut connections allow the network to have many more layers than traditional CNNs (e.g. a hundred more layers). With shortcut connections, layers can learn to be discarded, when necessary. This has a regularizing effect, allowing the final networks to have many more layers. Some researchers believe shortcut connections enable ResNets to behave like an ensemble of smaller neural networks (Veit, Wilber, & Belongie, 2016). During the backward pass, shortcut connections also help with propagating the gradient signal, reducing the risk of vanishing gradients.

Residual neural networks were an important contribution to deep learning, and some of the best results in image segmentation were achieved thanks to them (H. Zhao et al., 2016).

2.3 Segmentation via Deep Learning

There have been multiple attempts at using deep learning models to perform image segmentation. Early work made extensive of patchwise models, e.g. patchwise convolutional neural networks (Farabet, Couprie, Najman, & LeCun, 2013; Pinheiro & Collobert, 2014). Patchwise models look at a small region (often square) in the image, and output the label for that region's central pixel. This is done for all the regions in the image, and the end result is a label for every pixel in the image. The work of Farabet et al. (2013) mixes patchwise convolutional neural networks with other techniques like superpixels and segmentation trees.

2.3.1 Fully Convolutional Networks

A fully convolutional network (Shelhamer et al., 2016), or FCN, is built upon a pre-trained convolutional neural network. The pre-trained CNN was trained on the ImageNet Large Scale Visual Recognition Challenge classification dataset (Russakovsky et al., 2015). As such, the pre-trained CNN was initially used for image classification instead of segmentation. Classification is the task of assigning a label to an image. A label can be anything from "cat", "airplane", "car", and so on. Two modifications are made to turn the classification CNN into a segmentation CNN:

1. The last fully connected layers are changed to 1x1 convolutional layers. This allows the CNN to output a spatial map (i.e. an output matrix that resembles an image), instead of a traditional classification label vector.
2. An upsampling layer is added at the end, ensuring the final spatial map has the same dimensions as the input image.

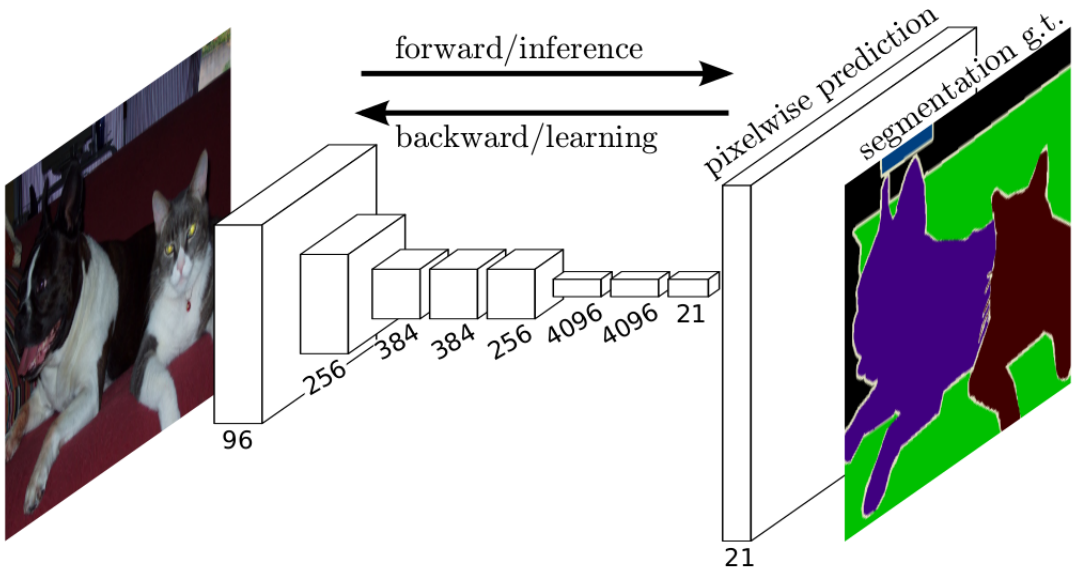


Figure 2.14: Image taken from Shelhamer et al. (2016). This image depicts a fully convolutional network, which is a neural network architecture designed for semantic segmentation. It is designed like a typical CNN, except that instead of dense layers at the end, the authors introduce an upsampling operation. This final upsampling operation guarantees that the output (segmentation mask) is the size of the input image.

In addition to the two mentioned modifications, FCNs make use of skip connections (Shelhamer et al., 2016). Skip connections fuse shallower layers (i.e. layers closer to the network's input) and deeper layers, creating a feature volume consisting of feature maps from different places in the network. A 1x1 convolution is then applied to this volume, creating the final output. Skip connections allow the final prediction to have higher precision, by taking into account information from different layers along the network.

2.3.2 U-Net

The U-Net architecture (Ronneberger et al., 2015) builds upon the FCN architecture. The architecture is made up of two parts: a contracting path, and an expanding path. The contracting part behaves like a typical CNN, with feature maps getting smaller as max-pooling operations are applied. The expanding path is similar to the contracting path, but instead of using max-pooling operations to reduce feature map size, upsampling operations are applied to increase feature map size.

The main difference between U-Nets and FCNs is the upsampling part, or the expanding path, as the authors call it. U-Nets apply upsampling operations multiple times, across different layers. This results in several different upsampled volumes, which are then fused with previous feature maps of corresponding size. In the FCN architecture, the upsampled feature maps are joined into one single feature map volume. Also, the FCN

uses a pre-trained network, while the U-Net is trained from scratch.

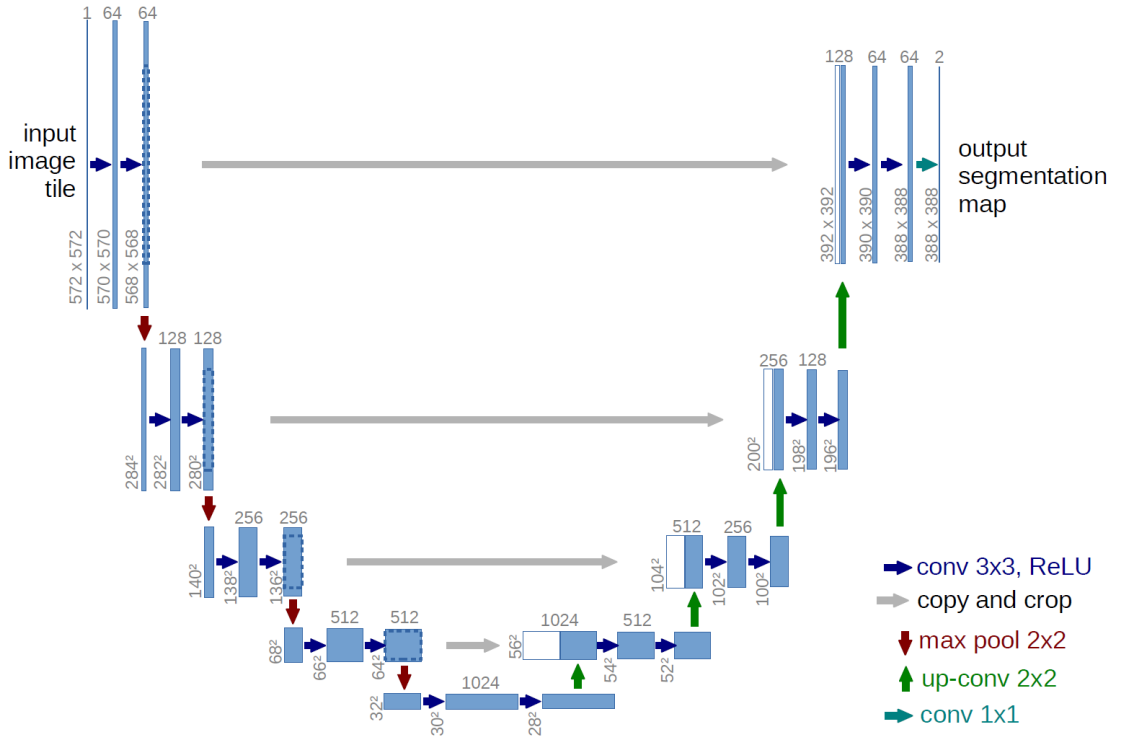


Figure 2.15: Image taken from Ronneberger et al. (2015). The U-Net architecture. Very similar to the FCN architecture (see Figure 2.14). The main difference is how the upsampling is done across multiple layers.

In addition to the architecture, the authors make use of extensive data augmentation techniques, and achieved the number one position in the ISBI cell tracking challenge (Arganda-Carreras et al., 2015).

The U-Net architecture is used widely in image segmentation challenges (e.g. in the "Dstl Satellite Imagery Feature Detection" challenge (Kaggle, 2017), top entries were based mostly on U-Nets). The wide use of U-Nets is mainly due to their simplicity. There are only 4 operations involved:

1. Convolutions — standard convolution operations, like we've seen in Section 2.2.3.
2. Deconvolutions — up-sampling operation (up-conv in Figure 2.15). In a nutshell, deconvolution consists in convolving a filter across an input image $\mathbf{X}$, in such a way that $\mathbf{X}$ is upsampled. Deconvolution filters have learnable weights, just like convolution filters (Dumoulin & Visin, 2016). Deconvolution is sometimes called transposed convolution.
3. Concatenation — feature maps from some layers are concatenated with feature maps from other layers, as we can see in Figure 2.15. This is similar to the skip connections found in fully convolutional networks (Shelhamer et al., 2016).

4. Pooling — regular max-pooling, as described in Section 2.2.3.

All these operations are illustrated in Figure 2.15.

2.3.3 Pyramid Spatial Pooling Network

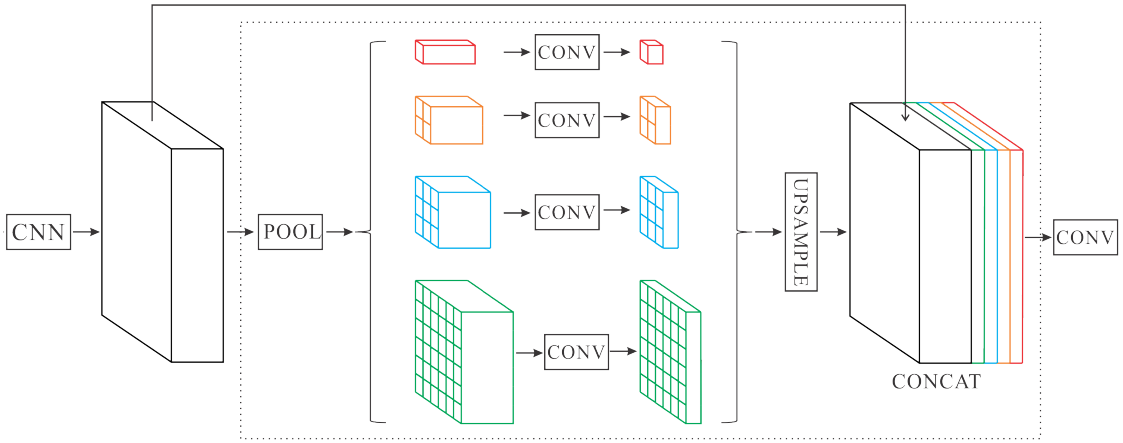


Figure 2.16: Image taken from H. Zhao et al. (2016). The PSP Net architecture. A pre-trained CNN is used (CNN in this Figure). It produces an output volume, which is then fed into the PSP module (zone with the dashed rectangle). This module takes the output volume, and performs multiple pooling operations on it. The result of these pooling operations is passed through a 1×1 convolution, upsampled, and concatenated with the original output volume.

The pyramid spatial pooling network (H. Zhao et al., 2016), or PSP Net, is also built upon the fully convolutional network. Just like FCNs, the PSP Net is based on a pre-trained network (a ResNet, represented by the words "CNN" in Figure 2.16), and utilizes a learnable up-sampling operation (deconvolution).

Unlike fully convolutional networks, PSP nets make use of a special kind of convolutional filters, called dilated convolutional filters (Yu & Koltun, 2015). The dilated filters get to keep the pre-trained filters' weights (that's the point of using pre-trained nets), and are applied to a greater region of the image. The region a neuron looks at in an image is also called "field of view", or FOV. By applying a filter to a greater image area, the neurons' FOV increases, and the neurons take into account more contextual information. In other words, dilated filters give greater contextual information than traditional filters, while having the same amount of weights. Also, now that the filters cover a greater region of the input image, the amount of calculations associated with convolution layers is reduced. To illustrate why this happens, imagine a filter so big, that it covered the whole input image. This filter would only need to be applied once - no need to convolve the filter across the input image.

Using pre-trained ResNets with dilated filters is not a novel idea. The work of Chen, Papandreou, Kokkinos, Murphy, and Yuille (2016) used a similar method. The main contribution of PSP Nets is the pyramid spatial pooling module (or PSP module, hence the name of the architecture). The PSP module offers global contextual information that most deep learning architectures fail to provide. Global contextual information is obtained by taking the pre-trained ResNet's output volume and performing average pooling on it. The average pooling is performed multiple times, at different scales. The pooling results are upsampled and concatenated with the original pre-trained ResNet's output volume. Then, a 1x1 convolution is applied to the final concatenated volume, just like it's done with Fully Connected Networks and U-Nets. See Figure 2.16 for a visual explanation. The PSP module is a clever and simple way to improve the neural network's performance.

There are several datasets used to benchmark image segmentation models. A popular one is the PASCAL VOC 2012 Object Segmentation challenge. The challenge ended in 2012, however, to this day there are still submissions being made, and researchers around the world are constantly competing to get top results. As of May 2017, the PSP Net is currently placed 1st in the leaderboard (University of Oxford, 2012).

Chapter 3

Building our Lung Nodule Segmentation System

In this chapter, we'll go into detail about the data, experiments, and the techniques utilized to build our model. Finally, we discuss our model's performance, along with some of the observed results. The next section talks about the dataset we utilized to train our final model, along with some tools that helped us build our lung nodule segmentation system.

3.1 Data and Tools

Data is at the heart of this work, and as such, it deserves a dedicated section. The used dataset, and sole reason why this work was made possible, was the LIDC-IDRI dataset (Armato et al., 2011). The LIDC-IDRI dataset is made available by The Cancer Imaging Archive (Clark et al., 2013), under the Creative Commons 3.0 Unported License.

The LIDC-IDRI dataset is not easy to navigate: it comes with a lot of information associated with each scan, so much so that the original authors call it a "database", instead of a dataset. However, there are free valuable tools online to help navigate the LIDC-IDRI dataset. In particular, the pylidc python library (Hancock & Magnan, 2016; Matthew C. Hancock, 2016) was used to extract and clean nodules and their segmentation masks.

There are many types of medical scans: MRI, X-ray, DXA, Ultrasound Scans, etc. The scans in the LIDC-IDRI dataset are CT (Computed Tomography) scans, and so, these will be the scans our model is built upon.

3.1.1 Computed Tomography

Computed Tomography (CT) (Brenner & Hall, 2007) is the preferred imaging modality for lung nodule detection and analysis (Kurihara et al., 2014). As such, our lung nodule

segmentation system will be designed to work exclusively on CT scans. The system will likely to fail on images originating from MRI scans, chest X-Rays, or other types of scans.

Tomography is an image extraction technique that relies on penetrating waves to extract 2D images from 3D objects. The 2D images are created by sections, and resemble slices from the original 3D object. In Computed Tomography, the penetrating wave is an X-ray, and the resulting 2D slices are then combined and processed by a computer. CT scans allow the user to see inside the scanned objects without recurring to surgical procedures. CT scans are sometimes called Computed Axial Tomography (CAT) scans, or Computer Aided Tomography scans.

Some say the X-ray radiation needed to perform CT scans can increase the risk of cancer (Brenner & Hall, 2007). That's why lung cancer screening programs like the NLST (National Cancer Institute, 2014) opt for CT scans that require a smaller radiation dose (low-dose CT scans), compared to standard-dose CT scans. Low-dose CT scans reduce the amount of radiation the patients are exposed to, while achieving comparable results to standard-dose CT scans (Kubo et al., 2016). The dataset our work is based on contains both standard-dose and lower-dose CT scans.

3.1.2 LIDC-IDRI Dataset

Our final lung nodule segmentation system will be trained on the LIDC-IDRI dataset (Armato III et al., 2015). The LIDC-IDRI dataset is a public, free dataset of thoracic CT scans. Each CT scans has been annotated by expert radiologists. The CT scan annotations provide multiple information lung nodules present in the scan.

There are 1018 CT scans in the LIDC-IDRI dataset. There are 8 patients with 2 CT scans, but other than these 8 patients, every scan belongs to a different patient. So, in total, the LIDC-IDRI database has CT scans from 1010 patients.

The CT scans in the LIDC-IDRI dataset were taken with different types of CT scanners. Not only were the scanners different, their settings (e.g. tube voltage used, exposure, etc.) may also differ from scan to scan. The dataset is made up of both standard dose and lower-dose CT scans. Also, the CT scans may display various types of unexpected artifacts. All these factors lead to great variability between CT scans. In our case, this will make our final deep learning model work for a wider range of CT scans. The final model will also be robust to unexpected anomalies present in CT scans.

Each scan may have multiple annotations in the XML format. These annotations are made by radiologists. The radiologists were asked to find three possible types of lesions in a CT scan:

1. Nodules with diameter under $3mm$ (nodules $< 3mm$).

2. Nodules with diameter greater or equal to 3*mm* (nodules $\geq 3mm$).
3. Non-Nodular lesions (e.g. apical scars).

The radiologists used the following three step process for defining lesion types:

1. Lesion identification — Is the observed structure a possible lung lesion?
2. Lesion size specification — Is the lesion $< 3mm$ or $\geq 3mm$?
3. Definition of the lesion type — Is the lesion a nodule? If the lesion is $< 3mm$, is it benign?

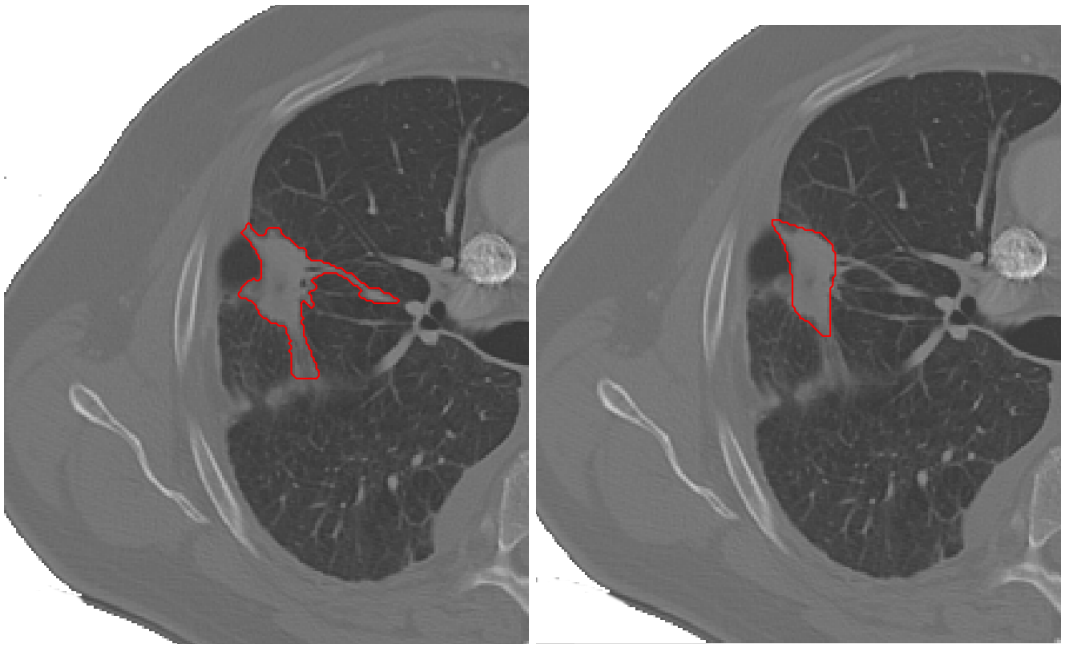
The main type of lesion, and the main focus of the LIDC-IDRI are nodules $\geq 3mm$. For these nodules, the radiologists were asked to provide the following additional information:

1. Subtlety — How difficult is the nodule to detect.
2. Internal Structure — What's inside the nodule (soft tissue, fluid, fat or air).
3. Calcification — Is the lung nodule calcified or not, and what is the pattern of calcification.
4. Spiculation — Nodule's degree of spiculation. Can be either "none" or "marked". Spiculation is "marked" when there are linear strands extending from the nodule margin, making the nodule's margin look uneven or "spiky".
5. Lobulation — Nodule's degree of lobulation. Can be either "none" or "marked". Lobulation is "marked" when a nodule looks like several round nodules joined into one.
6. Shape Sphericity — How spherical is the nodule.
7. Solidity (or texture) — How solid is the nodule (solid, ground glass or mixed).
8. Margin — How well-defined is the nodule's margin.
9. Likelihood of Malignancy — Nodule's likelihood of malignancy, assuming the patient is a 60 year old male smoker. This feature is a valuable feature if we wanted to create a machine learning model that looks for cancer in pulmonary CT scans.
10. Segmentation Mask — Hand drawn outline of the nodule. Some segmentation masks were created with the help of computer programs.

Most of these features are ranked using values ranging from 1 to 5. Again, these features are only present for nodules with a diameter greater than 3*mm*.

A total of 12 radiologists participated in annotating the LIDC-IDRI CT scans. However, only 4 radiologists looked at any single scan. This means that for every CT scan, there

are 4 sets of annotation belonging to 4 different radiologists. Some of the resulting annotations show substantial difference among radiologists, not only in the segmentation masks, but also on the labeling of nodules (e.g. one radiologist could say a nodule was malignant, while others say it's not). See Figure 3.1 for an example of how much the nodule segmentation masks differ from radiologist to radiologist. Also, the definition of "lung nodule" was deliberately not specified by the LIDC-IDRI Research Group, meaning radiologists could have different definitions of what constitutes a "lung nodule".



(a) Annotation 1 (b) Annotation 2

Figure 3.1: This particular nodule was annotated by all radiologists. In this Figure we see two very different segmentation contours by two different radiologists. Scan taken from the LIDC-IDRI dataset (Armato et al., 2011).

One very important detail for our work is how annotation grouping is done. Annotation grouping is the process of grouping annotations that belong to a specific lesion in a given scan. The LIDC-IDRI dataset doesn't provide information on which lesion a given annotation belongs to. Let's assume a CT scan, S_1, shows lesions $l_{1,2,...,i}$. The radiologists look at S_1, identify the lesions, and create annotations $a_{1,2,...,j}$. Notice that the amount of lesions (i) may be different from the amount of annotations (j), because certain lesions may go unnoticed by some radiologists, and some lesions may have multiple annotations by different radiologists. In theory, the maximum amount of annotations a lesion may have is 4, because a maximum of 4 radiologists looked at any given scan. The issue here is that there is no way for the dataset user to know which annotations belong to a given lesion. For example, annotations a_2, a_{11} and a_{26} in the LIDC-IDRI dataset may all belong to lesion l_1, but LIDC-IDRI doesn't provide this information. The process of establishing

which annotations belong to which lesions is referred to as "grouping" or "clustering" annotations. This process is highly subjective, and literature based on the LIDC-IDRI dataset may report different amounts of lesions for the same set of scans, due to different grouping procedures.

The grouping of annotations in the LIDC-IDRI main paper (Armato et al., 2011) was done by visual inspection of the annotations. The internal inventory of lesions reported in the LIDC-IDRI main paper was:

- 7371 nodules (either $< 3mm$ or $\geq 3mm$), by at least 1 radiologist.
- 2669 nodules $\geq 3mm$ by at least 1 radiologist.
- 928 nodules $\geq 3mm$ by all 4 radiologists.

Grouping of annotations can not only be a highly subjective process, but it can also be a tedious one. And that's where the pylidc library (Matthew C. Hancock, 2016) comes into play. The pylidc library is written in python, and was built to facilitate user interaction with the LIDC-IDRI dataset. It does a lot of things, but most importantly for our work, it has a built-in algorithm for grouping nodules together. The pylidc library is able to handle CT scans in their original filetype - the DICOM (Digital Imaging and Communications in Medicine) filetype.

3.1.3 Data Cleaning and Pre-Processing

Thanks to the pylidc library, we now have a reliable and consistent way to group nodules with a diameter of $3mm$ or more (our lesions of interest). The next step is the creation of the training, validation and test sets. All of these sets are made up of volumetric regions that contain lung nodules. In other words, only certain volumes of interest inside a given CT scan are used, instead of the whole CT scan. These volumes of interest (VOIs) have $40 \times 40 \times 40$ pixels, where each pixel represents a $1mm \times 1mm \times 1mm$ cube. Every VOI has a matching lung nodule segmentation mask. The segmentation masks have the same dimensions as the VOIs, and are made up of binary data, with "1" indicating that a pixel belongs to a lung nodule, and "0" if the pixel doesn't belong to a lung nodule.

The creation of the segmentation masks is not a trivial task. We need to take into account all the existing segmentation masks for a given nodule, and create a single, final segmentation mask. Following the approach of Kubota et al. (2011), Messay et al. (2015) and others, our final segmentation masks were composed of pixels with a 50% consensus truth. 50% consensus truth means that a pixel belongs to a nodule (i.e. has label "1") if that pixel is in at least 50% of the segmentation masks. So if a nodule has 4 segmentation masks, a pixel has to be in at least 2 of the masks to be labeled as "1", or "nodule". If a

Table 3.1: Our reported number of nodules in the LIDC-IDRI dataset, training, validation and test sets. The train, val and test sets are all subsets of the LIDC-IDRI dataset. Nodules with only 1 annotation were excluded from the train, val and test sets. All nodules were grouped with the pylidc library. The first line in the Table reads: "LIDC-IDRI has 2651 nodules annotated by 1 or more radiologists".

Nodule Set	Number of Nodules	Annotated by X Radiologists
LIDC-IDRI	2651	1+
LIDC-IDRI	1882	2+
LIDC-IDRI	910	4 (All)
Training Set	1605	2+
Validation Set	178	2+
Test Set	79	3+

nodule has 3 annotations, a pixel has to be in 1.5 (rounded to 2) masks. If a nodule has 2 annotations, pixels belonging in either mask are considered. Finally, nodules annotated by only 1 radiologist were not considered.

From Table 3.1, we can see that our test set is made up of 79 lung nodules. The test set is based on CT scans from the original LIDC dataset (McNitt-Gray et al., 2007), which was a precursor to the LIDC-IDRI dataset. The original CT scans belonging to the LIDC dataset are now part of the LIDC-IDRI dataset, so all we're doing is defining the test set a sub-set of the LIDC-IDRI dataset. There have been multiple papers that utilize this test set, using only nodules that were annotated by 3 or more radiologists (Messay et al., 2015). The specific scans used for the test set are available in the following URL (via the link "LIDC Image Dataset"):

- http://dx.doi.org/10.7937/K9/TCIA.2014.V7CVH1JO

This particular link was made available by Messay et al. (2015), in an effort to help and encourage future work on this area.

The training and validation sets are defined by all the nodules that are not in the LIDC test set. There are 1783 such nodules. The validation set is 10% of 1783, or 178 nodules, as we can see in Table 3.1. The training set is thus $1783 - 178 = 1605$ nodules.

Also from Table 3.1, we can see that leaving out nodules only annotated by 1 radiologists means discarding a huge portion of the LIDC-IDRI dataset. This may not be such a good idea, given that deep learning models require large datasets to work properly (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).

Considering the original LIDC-IDRI paper reported 2669 nodules annotated by 1+ radiologists, and 928 nodules annotated by all four, looking at Table 3.1 we can conclude

that the pylidc library is doing a good job grouping nodules together. The numbers aren't exactly the same, but they're fairly close.

Pixel Values

Pixels in a CT scan slice have values from +3071 to -1024 on the Hounsfield scale. The Hounsfield scale is a quantitative scale used for describing radiodensity, i.e. the relative inability of X-rays to pass through a particular material. In the Hounsfield scale, water has a radiodensity of 0, and air has a value of -1000.

Neural networks tend to work better with values in a lower range (Lapedes & Farber, 1987). So, in order to reduce the magnitude of the input values, we rescaled the values into a range of $[0,1]$. But before they were rescaled, any Hounsfield values above 400 were removed, as these values tend indicate bones in the Hounsfield scale. Also, lung nodules measurements with values above 400 are most likely due to unwanted artifacts (Perandini et al., 2016; Swensen et al., 2000). Thresholding values at 400 was also common practice in the LUNA16 challenge (Setio et al., 2017).

After the values were in the range of $[0,1]$, a mean was performed of all the pixel values in the LIDC-IDRI dataset. The mean pixel value was found to be $\mu = 0.25$. The mean is then subtracted for all the images in the dataset, zero-centering the dataset. This tends to help with the training process, and is common practice in neural network literature (Krizhevsky et al., 2012).

Voxel Volume

The next step was to make the voxel volume uniform across all CT scans, setting it to $1mm \times 1mm \times 1mm$. A voxel is a 3D version of a pixel. Since we're dealing with 3D scans, the term "pixel" is not scientifically accurate, voxel should be used instead. But for simplicity, we sometimes use both terms interchangeably.

There were 2 DICOM properties used to make voxel volume uniform:

- The "Pixel Spacing" property. This property represents the distance between pixel centers, in a 2D horizontal scan slice.
- The "Slice Thickness" property, which represents the distance between consecutive 2D horizontal scan slices. Slice thickness has values between $0.6mm$ and $5.0mm$.

Once we have pixel spacing and slice thickness, we can resample (upsample or downsample) the 3D scans into having $1mm \times 1mm \times 1mm$ voxel volume. Resampling was done using cubic spline interpolation (McKinley & Levine, 1998).

Summary

To recap, the data cleaning and pre-processing phase are made up of the following ordered steps:

1. Define the training, validation and test sets. The test set consists of nodules in the original LIDC dataset (McNitt-Gray et al., 2007). The training set is made up of all the remaining nodules in the LIDC-IDRI dataset (Armato et al., 2011). The validation set is made up of 178 randomly selected nodules from the training set. The segmentation masks for these sets are created using a 50% consensus truth.
2. Rescale the original Hounsfield values to values between $[0, 1]$.
3. Normalize the voxel size via resampling, ensuring every voxel is $1mm \times 1mm \times 1mm$.
4. Subtract the dataset mean from each picture.

The NumPy library (van der Walt, Colbert, & Varoquaux, 2011) was used to pre-process the data. The original DICOM scans, once loaded into python via the pylidc library, have a property called "pixel_array". This property contains the CT scan data as a 3D NumPy array, in Hounsfield values. Once we have access to the data in the form of a NumPy 3D array, we can use the NumPy library and all its functions to manipulate the data.

3.2 Model Details and Training Techniques

In this section we'll describe our model's architecture, along with some optimization details and performance enhancing techniques.

3.2.1 Final Architecture

The chosen deep learning model was a modified 3D U-Net (Çiçek et al., 2016). The model's architecture was chosen because it's a model used specifically for segmentation, and contains 3D convolution filters, instead of the regular 2D filters.

In theory, we could use 2D convolution filters. However, there are some issues associated with 2D filters, such as:

- If our input images are 3D, we must convert them to 2D. There are multiple ways to do this, and finding the best one could be a difficult task. Also, most pre-trained neural networks take 2D images with 3 channels (R,G,B), so we'd have to find some way to reshape our original 3D images, not only making them 2D, but also giving them 3 color channels (or something similar).

- Incorrect segmentation results may occur in slices near the end of a nodule. These slices are mostly empty. 2D CNNs may be incapable of learning how to segment these almost empty slices, given 2D convolution filters don't take into account information from adjacent slices.

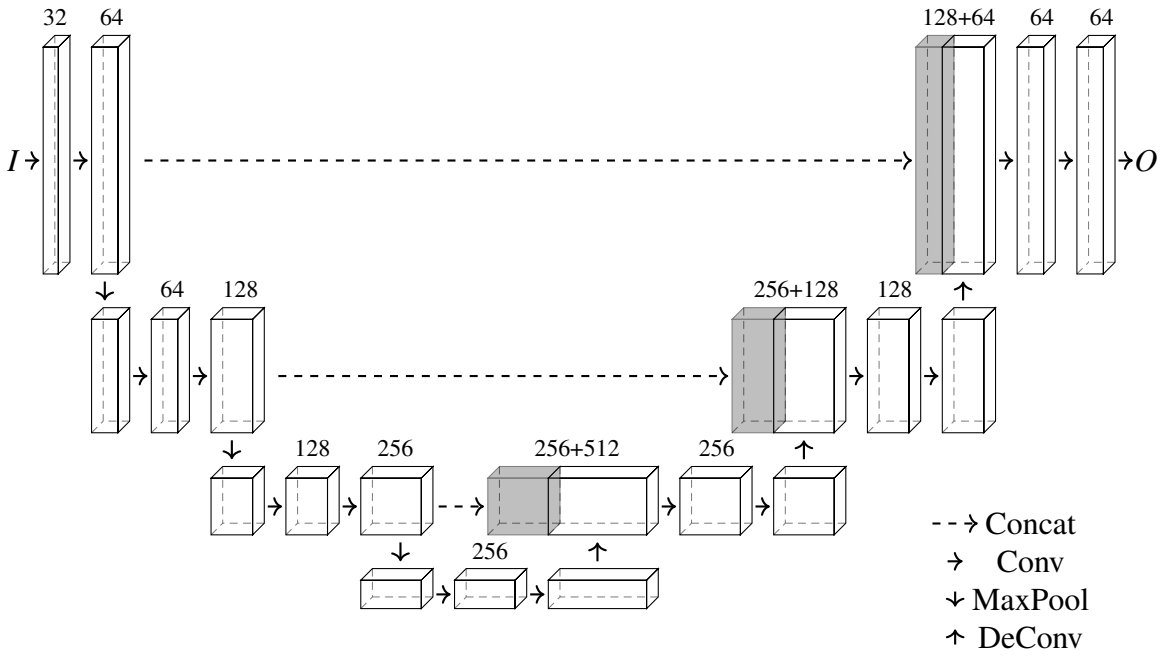


Figure 3.2: Drawing of our model's architecture, the 3D U-Net Architecture, originally by Çiçek et al. (2016). A grey box indicates a feature volume that has been concatenated. The numbers on top of the boxes indicate the number of feature maps belonging to each box. I represents the input image, which in our case, is a $40 \times 40 \times 40$ lung CT region. O is the model's output, a $40 \times 40 \times 40$ matrix of scores, with values in the range of $[0, 1]$. Scores near 1 should indicate pixels that belong to a lung nodule, while scores near 0 should indicate non-nodule pixels.

Figure 3.2 depicts our architecture, the 3D U-Net (Çiçek et al., 2016). All the convolutions use $3 \times 3 \times 3$ filters with a ReLU activation function, except for the last convolution (before O), which uses a $1 \times 1 \times 1$ filter, followed by a sigmoid activation function (and thus, producing values in the range $[0, 1]$). From Figure 3.2, we can also see that the max-pooling operation (downward arrow) downsamples the volumes (i.e. makes the boxes smaller in height, but not smaller in width). These operations create what the authors call the "contracting path". The deconvolution operation, on the other hand, upsamples the volumes to their original shape, making up the "expanding path".

There are some slight differences between our 3D U-Net and the original. The differences are:

Table 3.2: Information on the first 4 layers in our model.

Layer (type)	Output Shape	Parameters	Connected to
input_1 (InputLayer)	(None, 40, 40, 40, 1)	0	0
conv3d_1 (Conv3D)	(None, 40, 40, 40, 32)	896	input_1
conv3d_2 (Conv3D)	(None, 40, 40, 40, 64)	55360	conv3d_1
maxpool3d_1 (MaxPooling3D)	(None, 20, 20, 20, 64)	0	conv3d_2

- No batch normalization (Ioffe & Szegedy, 2015). Batch normalization is a technique used to ensure the inputs of every convolution layer have zero mean and unit variance. The authors claim this makes the training process faster, more robust, and also has a regularizing effect.
- Different loss function. The authors used a modified softmax loss, while we used the intersection over union loss.
- The original 3D u-net has $19,069,955$ parameters (or trainable weights), while ours has $19,068,993$ (approx. minus 1000 parameters). The difference of parameters is possibly due to the lack of batch normalization in our model.

The first 4 layers in our model and their corresponding number of parameters can be seen in Table 3.2. Table A.1, in the appendix section, has information on all our model's layers.

3.2.2 Optimization

The intersection over union (IOU) loss function and the Adam (Kingma & Ba, 2014) update rule were used. The IOU loss function was used instead of the traditional Dice loss function, because the most commonly used metric in lung nodule literature is the IOU metric (Messay et al., 2015), and so we decided to optimize directly for that metric.

There was no IOU loss function in the Keras (Chollet et al., 2015) framework, so we had to create our own. The code for the IOU loss function is:

```
def iou_loss(y_true, y_pred):
    return 1. - iou(y_true, y_pred)

def iou(y_true, y_pred):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)

    intersection_vec = y_true_f * y_pred_f
    union_vec = y_true_f + y_pred_f - intersection_vec

    return K.sum(intersection_vec) / K.sum(union_vec)
```

In the above code, K is the Keras backend, a library that provides optimized, low-level multidimensional array manipulation functions.

The Adam update rule was used, with a learning rate that was changed multiple times during training. This was done mainly manually, without a fixed learning rate schedule. The manual changing of the learning rate was performed when the training and validation loss stagnated after a few epochs. More details on the learning rate will be given in Section 3.3.

3.2.3 Improving our Model's Performance

To improve our model's performance and reduce overfitting, we used data augmentation and L_2 regularization. L_2 regularization was applied to every weight in every layer.

In terms of data augmentation techniques, two techniques were used: image rotation, and image scaling. Both scaling and rotating were applied to the 3D images, meaning the augmentations were possible in different combinations of axis, e.g. scaling along X and Z axis, only along Y axis, rotating along X and Y axis, etc. Having three axis means the data augmentation possibilities with 3D images are greater compared to regular 2D images. The axis selection for the augmentations were random. Whether to apply or not apply these augmentations was also a random choice, meaning any image could have scaling augmentation, rotation, both or none.

The data augmentation techniques were applied to the images in a minibatch before being fed into our model, i.e. they were done at training time.

3.3 Training Experiments

The Keras (Chollet et al., 2015) framework was used for creating and training our deep learning model, using the tensorflow backend (Abadi et al., 2015).

Training involved a lot of trial and error. The first step in training was to make sure the model overfitted on a small subset of the training data. This first test was done because large neural networks are prone to overfitting (Srivastava et al., 2014), so if our neural network fails to overfit, it probabily isn't big enough. A big neural network is one with a

large amount of parameters (learnable weights). After overfitting on a small subset of data, we overfitted the model on the whole training set.

In order to overfit the model on the whole training set, we started with a learning rate of 10^{-5}, and let the model train this way for the first 20 epochs. After the first 20 epochs, the model's training error wasn't going down, and the training and validation errors were very similar. In other words, this learning rate wasn't enough to make the model overfit. So after the first 20 epochs, we lowered the learning rate to 10^{-6}, and the model trained for another 60 epochs. Again, this learning rate wasn't low enough. It wasn't until the learning rate got down to 10^{-7} that the model started to overfit. The training loss got down to 0.1 (with an IoU of 90%), while the validation loss stayed at 0.6 (IoU of 40%).

Once the model overfitted on the training set, it was time to make it operate in acceptable ways. This meant reducing the overfitting caused in the previous step, by introducing two techniques: data augmentation, and L_2 regularization, as explained in Section 3.2.3.

3.3.1 Regularization Experiments

The regularization techniques described in this section were performed with a learning rate of 10^{-5}. The batch size was 4 for all the experiments we made. The batch size was fixed at 4 due to GPU memory constraints (batch sizes over 4 exhausted the GPU's memory).

The first regularization experiment was done with a regularization hyperparameter λ of 10^{-2}. Next, we tried λ values of 10^{-3}, 10^{-4} and 10^{-5}. Any of these λ values, along with data augmentation, were sufficient to reduce the overfitting caused in the previous step. The training set's IOU was now between 0.73 and 0.75, with the validation set's IOU between 0.69 and 0.71. To give some perspective on these values, the best ever IOU reported during training was 0.72 (with 50% consensus truth), in the work of Messay et al. (2015), although they used a different sub set of LIDC-IDRI nodules. Table 3.3 has more information on the effect of the regularization parameter λ on the training results.

The final model's λ value was 10^{-5}, as smaller values seemed to hurt the validation set's performance (maybe because the model started to overfit).

3.3.2 Learning Rate

The learning rate experiments in this section were made with a fixed regularization parameter λ of 10^{-5}. Also, the model had trained for 80 epochs when we began these experiments.

Until now, the learning rate was fixed at 10^{-5}. However, the model's performance started to stagnate, and we decided to tune the learning rate to see if we could improve the model's performance.

Table 3.3: Effect of L_2 regularization on the training and validation IOU scores. As we lower λ, the training IOU score improves, while the validation IOU decreases (as is expected - if λ is close to 0, then there is no regularization and the net goes back to overfitting). The higher the IOU, the better. The learning rate was fixed at 10^{-5}. The highlighted validation set value (0.715) was the best result we got in this experiment.

λ (Reg. Parameter)	Training IOU	Validation IOU
10^{-2}	0.712	0.68
10^{-3}	0.72	0.693
10^{-4}	0.735	0.7
10^{-5}	0.736	**0.715**
10^{-6}	0.74	0.706
10^{-7}	0.74	0.7

The basic strategy for finding the best learning rate was:

1. For a given learning rate, train the model until the validation performance stagnates.
2. Once the model's performance stagnates, lower the learning rate. Repeat step 1.

In Table 3.4 we show the model's performance for different learning rates. Overall, reducing the learning rate had a positive effect on the IOU scores, but this effect seemed to wear off for values under 10^{-6}.

Table 3.4: Effect of the learning rate on the training and validation IOU scores. The higher the IOU, the better. The regularization parameter λ was fixed at 10^{-5}. Also, note that all of these learning rate values were tested on a model already trained for 80 epochs, with stagnated IOU scores (both on the training and validation set). The highlighted validation set value (0.721) was the best result we got in this experiment.

Learning Rate	Training IOU	Validation IOU
1×10^{-4}	0.731	0.7
5×10^{-5}	0.73	0.71
1×10^{-5}	0.736	0.715
5×10^{-6}	0.74	0.713
1×10^{-6}	0.742	**0.721**
5×10^{-7}	0.742	0.72

3.3.3 Decision Threshold

The decision threshold was defined after we found our final λ and learning rate: 10^{-5} and 10^{-6}, respectively. The outputs of our deep learning model are real values between 0 and 1. However, segmentation masks are made up of boolean values, with 0's and 1's indicating whether or not a pixel belongs to a lung nodule. To convert the score value to booleans, we must specify a decision threshold t. Every score above the threshold t is considered as 1, or belonging to a nodule, and every score below t is considered a 0.

Before applying threshold t, the performance (Intersection Over Union) scores were computed with our model's raw output values (in the range $[0,1]$). Thresholding the raw output values using a t value of 0.5 seemed improve performance, but only by about 0.005%. From the validation set, we found that the best t value was approximately 0.8, which gave a minor 0.03% increase in the validation IOU (from 0.72% to 0.724%). While this increase is minor, Figure 3.3 shows a pattern: increasing t seems to also increase performance.

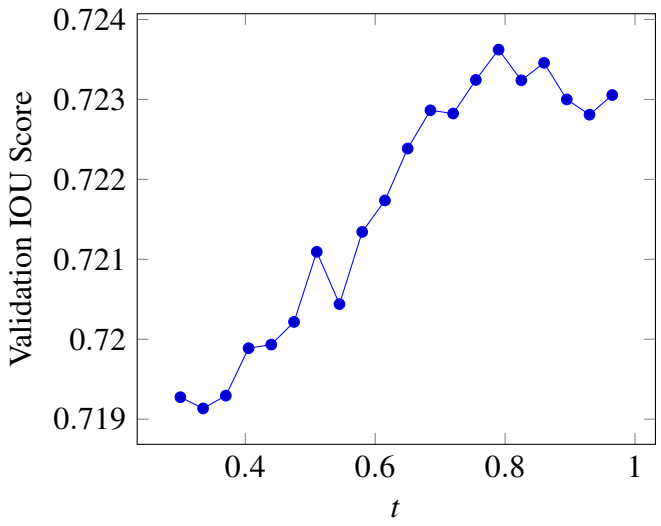


Figure 3.3: Validation IOU score, for different threshold values.

3.3.4 Summary

To recap, our training strategy was based on the following three steps:

1. Overfit the model. This is a test done to see if the deep learning architecture we chose is capable of dealing with the task at hand. If the a model isn't capable of overfitting on a small set of data, there's probably something wrong.
2. Reduce overfitting, by applying techniques with a regularizing effect.

3. Increase the model's final performance, by lowering the learning rate manually.

The hyperparameters we tuned were:

1. The batch size. The final batch size was 4, due to GPU memory constraints.
2. The regularization parameter, λ. The final λ was 10^{-5}.
3. The learning rate. The final learning rate was 10^{-6}.
4. The decision threshold, t. This is not a machine learning model's hyperparameter, but it's an important value to adjust. The final t was 0.8.

3.3.5 Failed Experiments

There are two additional techniques we tried, which unfortunately didn't bring any benefit to our final system:

1. Batch normalization (Ioffe & Szegedy, 2015), also known as BatchNorm - in a nutshell, this technique applies a set of transformations to the convolution layers' inputs, ensuring they have zero mean and unit variance. The authors claim this makes the training process faster, more robust, while also having a regularizing effect.
2. DropOut (Srivastava et al., 2014) - to put it simply, DropOut is a technique that helps prevent overfitting in neural networks. It works by randomly removing (or dropping out) neurons in a neural network during training.

In the original 3D U-Net paper (Çiçek et al., 2016), the authors used BatchNorm after each convolutional layer. We tried to implement BatchNorm in two separate ways: before and after each convolution layer. Both seemed to have no effect on the model's performance, so we decided to not include BatchNorm layers in our final model. One of the claims of the original BatchNorm paper is a decrease in training time. However, our training time was about 1 day for the model without BatchNorm, so training time wasn't an issue for us.

DropOut, unlike BatchNorm, had a very negative effect on our model's performance. Instead of getting around 72% IOU score on the validation set (which was our best performance), with DropOut, our model's performance peaked at 55%. Since DropOut reduces the number of neurons used in each forward pass, we speculate DropOut failed because our model needs to have a great number of neurons to perform well. After all, our model produces a $40 \times 40 \times 40$ output matrix of scores. That's a total of $64,000$ scores. Accurately predicting such a large amount of numbers is bound to require a model with large capacity.

Table 3.5: Performance of several different lung nodule segmentation systems, including our own. All the systems' IOU scores were calculated using scans from the LIDC dataset, with nodules annotated by three or more radiologists. These results all use a 50% consensus truth for defining nodule segmentation masks. Note that despite using the LIDC dataset, the test set's nodules are possibly different from system to system (e.g. our system's test set had 79 nodules, while Messay et al. (2015) had 77). This Table was based on Table 7 from Messay et al. (2015)

System	**IOU**
B. Zhao, Yankelevitz, Reeves, and Henschke (1999)	43%
Okada and Akdemir (2005)	45%
Kuhnigk et al. (2006)	56%
Wang et al. (2009)	58%
Kubota et al. (2011)	59%
Messay, Hardie, and Rogers (2010)	63%
Messay et al. (2015)	69%
Our System	**70%**

3.4 Results and Discussion

The final model's predictions were scored with the intersection over union (IOU) metric, as is tradition in lung nodule segmentation literature (Messay et al., 2015). Our final model achieved an IOU score of 70% on the LIDC dataset's nodules. A comparison with other lung nodule segmentation approaches is given in Table 3.5.

To our knowledge, our 70% IOU score is the best score on the LIDC dataset, using nodules annotated by 3 or more radiologists and with a consensus truth of 50%. However, it is hard to say if this result would be the same for other test sets, as the LIDC dataset presents a great amount of irregular and subtle nodules, thus presenting a greater challenge than other nodules in the LIDC-IDRI dataset (Kubota et al., 2011; Wang et al., 2009).

Tables 3.6, 3.7 and 3.8 show our model's performance for nodules of different subtlety, texture and margin. We created these tables because literature on lung nodule segmentation sometimes focuses on a specific type of lung nodule (e.g. solid, non-solid, etc.), and also to make sure our model was general enough to work for different types of nodules. The performance results were all obtained in the test set, which is made up of 79 nodules found in the LIDC dataset.

In Table 3.6, we show the performance of our model for different subtlety values. Subtlety is a measure of how easy a nodule is to detect. The values in Table 3.6 were obtained by averaging the radiologists' subtlety values for a given nodule. So if a nodule

Table 3.6: Model's performance on nodules of different subtlety. A nodule's subtlety indicates how easy it is to detect. For a given nodule, the subtlety value (1 to 5) was calculated by averaging all of the subtlety values assigned by different radiologists. There were no extremely subtle nodules in the test set.

Subtlety	Number of Nodules	IOU
5 (Obvious)	26	0.738
4 (Moderately Obvious)	33	0.717
3 (Fairly Subtle)	13	0.654
2 (Moderately Subtle)	7	0.564
1 (Extremely Subtle)	0	?

Table 3.7: Model's performance on nodules of different texture (or solidity). Texture indicates how solid the nodule is. For a given nodule, the texture value (1 to 5) was calculated by averaging all of the texture values assigned by different radiologists.

Texture	Number of Nodules	IOU
5 (Solid)	47	0.731
4 (Solid/Mixed)	25	0.663
3 (Part Solid/Mixed)	2	0.707
2 (Non-Solid/Mixed)	4	0.588
1 (Non-Solid/GGO)	1	0.633

was assigned subtlety values of 3, 3, and 5, by 3 different radiologists, the final subtlety value is the average,

$$\frac{3+3+5}{3} = 3.667, \tag{3.1}$$

which is then rounded to 4. The same principle was applied in Tables 3.7 and 3.8.

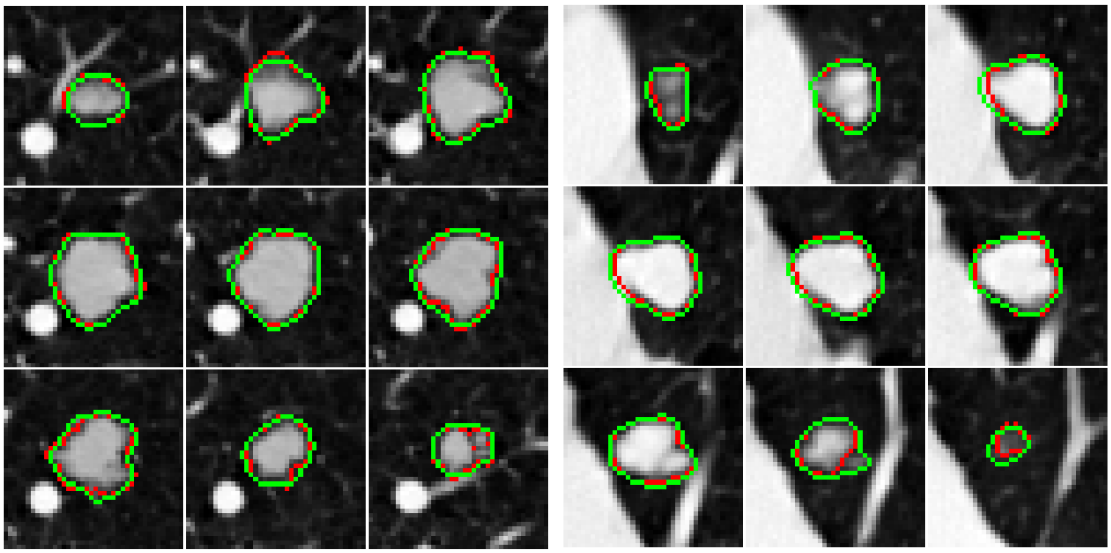
From Table 3.6, there is a clear relation between subtlety and the model's performance: the more subtle the nodule is, the worse our model performs. This relation is also observed in Table 3.8, which shows our model's performance for nodules with different kinds of margin (well defined / poorly defined margins). Maybe there is a direct relation between poorly defined margins and nodule subtlety. This would explain the performance similarities.

Table 3.7 shows our model's performance for nodules with different texture (also known as solidity). Texture is a measure of how solid a nodule is. It's hard to say if there is any relation between texture and performance, given that most of the test set's nodules were either solid or solid/mixed. However, we can see that our model performs best with solid nodules, and the performance seems to decrease with nodule solidity.

Table 3.8: Model's performance on nodules of different margin. The margin value indicates how well defined a nodule's margin is. For a given nodule, the margin value (1 to 5) was calculated by averaging all of the margin values assigned by different radiologists. There were no nodules with poorly defined margins in the test set.

Margin	Number of Nodules	IOU
5 (Sharp)	18	0.737
4 (Near Sharp)	37	0.714
3 (Medium Margin)	17	0.655
2 (Near Poorly Defined)	7	0.641
1 (Poorly Defined)	0	?

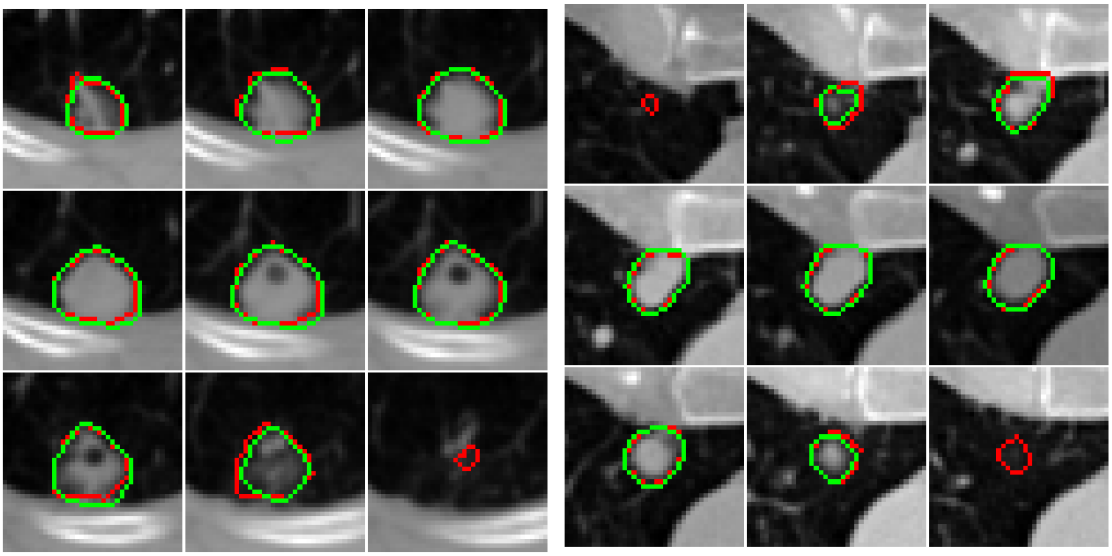
Figures 3.4, 3.5 and 3.6 compare our model's segmentation masks (in green) with the real segmentation masks (in red). Seeing more a lot of green and little red is a good thing - this means our system's segmentation mask overlaps with the real one. In some nodule slices, we see only a red contour. This means our model didn't produce a segmentation contour for that given slice.



(a) $IOU = 0.886$ (b) $IOU = 0.885$

Figure 3.4: Two nodules with the best IOU score: 0.886 and 0.885, respectively. Both are attached to lung structures: figure (a) is attached to vascular structures, and figure (b) appears to be attached to the lung wall. Red is the original segmentation contour, green is our model's segmentation contour. The slices we see here were selected manually, for better visualization of the nodule.

In Figure 3.4, we have the 2 nodules with the best IOU scores. We can see that our model is able to produce good segmentation masks, even if the nodules are attached to

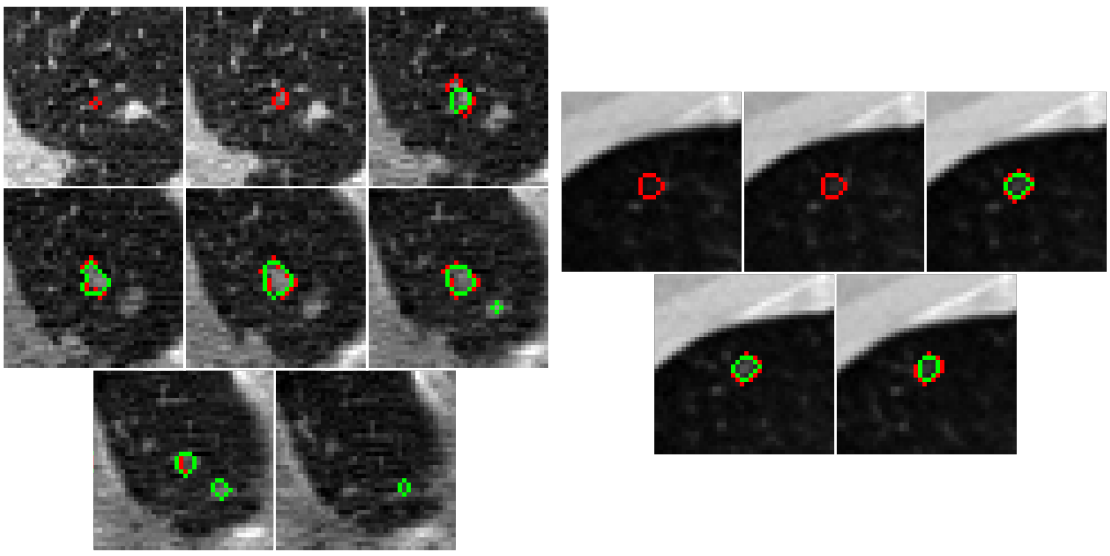


(a) $IOU = 0.837$ (b) $IOU = 0.833$

Figure 3.5: Two nodules clearly attached to lung structures. Red is the original segmentation contour, green is our model's segmentation contour.

different types of lung structure (vascularity, lung wall, etc). Figure 3.5 shows 2 additional nodules attached to the lung wall.

Figure 3.6 shows the 2 worst performing nodules in the test set. These are both small, slightly subtle nodules. When dealing with small nodules, a small slip up in the segmentation output can have a serious impact on the IOU score. For example, in Figure 3.6 (b), we can see that our model failed to segment the nodule in 2 slices. Since the nodule was small to begin with (only 5 slices, 1mm thick), the IOU score was calculated by dividing a small intersection term by a large union term, resulting in a small IOU score. Furthermore, in Figure 3.6 (a) we can see that our model created 2 contours in 1 single mask. It's unlikely that our model learned to detect multiple nodules in a $40 \times 40 \times 40$ image, given that the training set had images with multiple nodules close together. In this particular instance, we think it's more likely that the model thought it was looking at a nodule made up of two weakly connected portions. Such results were rare, and could possibly be fixed with a simple set of hand written rules (e.g. if there is more than 1 contour, only keep the contour nearest to the volume center). However, this would add unnecessary complexity to our system, which already achieves state-of-the-art performance.



(a) $IOU = 0.259$ (b) $IOU = 0.438$

Figure 3.6: Two nodules with the worst IOU scores: 0.259 and 0.438, respectively. Red is the original segmentation contour, green is our model's segmentation contour.

Chapter 4

Conclusion and Future Work

We have built a lung nodule segmentation system using exclusively deep learning. Our deep learning model uses a modified 3D U-Net architecture (Çiçek et al., 2016), and was trained with the intersection over union loss function (Rahman & Wang, 2016). We have showed deep learning models can be trained with the intersection over union loss, and still achieve state-of-the-art results. To be precise, our system achieved a 70% intersection over union score on a test set consisting of nodules from the original LIDC dataset. The nodules on our test set were annotated by at least 3 radiologists. The segmentation masks were created using a 50% consensus truth. This was all done in accordance with lung nodule segmentation literature (Messay et al., 2015).

Lung nodule segmentation system such as the ones described in this work have strong potential to assist doctors in evaluating more patients, in a more efficient and effective way. The results look promising, but there is still a lot of work to do.

In particular, one aspect that could be worked on is the input to lung nodule segmentation systems. Our system's input is a CT Scan VOI (Volume Of Interest) with a lung nodule. In real life, these CT Scan VOIs are not readily available. They are either created by radiologists, or by a Computer-Aided Detection system. Most lung nodule segmentation systems in literature take as input a seed point. However, in theory this is also not necessary. Ideally, a system would look at a complete CT scan, detect all the nodules, and automatically segment them, without any interaction from the user.

He, Gkioxari, Dollár, and Girshick (2017) introduced a deep learning architecture called Mask R-CNN. This architecture is capable of performing both detection and instance segmentation. Creating a lung nodule CAD system with such an architecture would eliminate the need for human input.

Our architecture used 3D convolution filters because we thought a 2D filter neural net would be harder to implement, and probably have worse performance. However, no experiments were done to verify these claims. It's true, using pre-trained neural nets

(especially pre-trained ResNets) has been the go to approach in producing some of the best performing image segmentation results. Finding a way to utilize pre-trained neural nets with 2D filters on 3D medical images could yield satisfactory results.

In terms of data, there are still some improvements to be made. Despite the LIDC-IDRI being a glorious dataset, it is not a reliable segmentation dataset: the nodules still need to be grouped together, and how the final segmentation masks are created varies from author to author. We believe a dedicated dataset for lung nodule segmentation would help future work on this area.

Finally, segmenting a lung nodule is only one part in a multi-step process that leads to diagnosing lung cancer. Further research is necessary to determine the viability of using deep learning segmentation models in a clinical setting.

Appendix A

Model Summary

Table A.1 has all the layers in our final model, along with the number of parameters and output shapes.

If a convolution layer has output shape $(None, 40, 40, 40, 32)$, this means the layer has 32 filters, each producing a $40 \times 40 \times 40$ feature map. The same logic applies to other convolution and deconvolution layers. These layers all use $3 \times 3 \times 3$ filters, except for the final convolution layer (conv3d_15), which uses a $1 \times 1 \times 1$ filter. The final convolution layer is also not followed by a ReLU activation function, unlike all the other convolution and deconvolution layers. Instead, it uses a sigmoid activation function,

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

producing output scores in the range of $[0, 1]$. All the other activation functions are ReLUs,

$$f(x) = max(0, x), \tag{A.2}$$

not represented in Table A.1, since they were considered to be an integral part of the convolution and deconvolution layers.

The total number of trainable parameters in our model is $19,068,993$.

Table A.1: Summary of our final model.

Layer (type)	Output Shape	Parameters	Connected to
input_1 (InputLayer)	(None, 40, 40, 40, 1)	0	0
conv3d_1 (Conv3D)	(None, 40, 40, 40, 32)	896	input_1
conv3d_2 (Conv3D)	(None, 40, 40, 40, 64)	55,360	conv3d_1
maxpool3d_1 (MaxPooling3D)	(None, 20, 20, 20, 64)	0	conv3d_2
conv3d_3 (Conv3D)	(None, 20, 20, 20, 64)	110,656	maxpool3d_1
conv3d_4 (Conv3D)	(None, 20, 20, 20, 12)	221,312	conv3d_3
maxpool3d_2 (MaxPooling3D)	(None, 10, 10, 10, 12)	0	conv3d_4
conv3d_5 (Conv3D)	(None, 10, 10, 10, 12)	442,496	maxpool3d_2
conv3d_6 (Conv3D)	(None, 10, 10, 10, 25)	884,992	conv3d_5
maxpool3d_3 (MaxPooling3D)	(None, 5, 5, 5, 256)	0	conv3d_6
conv3d_7 (Conv3D)	(None, 5, 5, 5, 256)	1,769,728	maxpool3d_3
conv3d_8 (Conv3D)	(None, 5, 5, 5, 512)	3,539,456	conv3d_7
deconv3d_1 (Deconvolution)	(None, 10, 10, 10, 51)	2,097,664	conv3d_8
concat_1 (Concatenate)	(None, 10, 10, 10, 76)	0	deconv3d_1, conv3d_6
conv3d_9 (Conv3D)	(None, 10, 10, 10, 25)	5,308,672	concat_1
conv3d_10 (Conv3D)	(None, 10, 10, 10, 25)	1,769,728	conv3d_9
deconv3d_2 (Deconvolution)	(None, 20, 20, 20, 25)	524,544	conv3d_10
concat_2 (Concatenate)	(None, 20, 20, 20, 38)	0	deconv3d_2, conv3d_4
conv3d_11 (Conv3D)	(None, 20, 20, 20, 12)	1,327,232	concat_2
conv3d_12 (Conv3D)	(None, 20, 20, 20, 12)	442,496	conv3d_11
deconv3d_3 (Deconvolution)	(None, 40, 40, 40, 12)	131,200	conv3d_12
concat_3 (Concatenate)	(None, 40, 40, 40, 19)	0	deconv3d_3, conv3d_2
conv3d_13 (Conv3D)	(None, 40, 40, 40, 64)	331,840	concat_3
conv3d_14 (Conv3D)	(None, 40, 40, 40, 64)	110,656	conv3d_13
conv3d_15 (Conv3D)	(None, 40, 40, 40, 1)	65	conv3d_14
activation_1 (Activation)	(None, 40, 40, 40, 1)	0	conv3d_15

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `http://tensorflow.org/` (Software available from tensorflow.org)

Adams, R., & Bischof, L. (1994, jun). Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *16*(6), 641–647. Retrieved from `https://doi.org/10.1109/34.295913` doi: 10.1109/34.295913

American Cancer Society. (2016). *Cancer Facts & Figures 2016.* `https://www.cancer.org/research/cancer-facts-statistics/all-cancer-facts-figures/cancer-facts-figures-2016.html`. (Accessed: 2017-05-10)

Arganda-Carreras, I., Turaga, S. C., Berger, D. R., Cireşan, D., Giusti, A., Gambardella, L. M., ... Seung, H. S. (2015, nov). Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in Neuroanatomy*, *9*. Retrieved from `https://doi.org/10.3389/fnana.2015.00142` doi: 10.3389/fnana.2015.00142

Armato, S. G., McLennan, G., Bidaut, L., McNitt-Gray, M. F., Meyer, C. R., Reeves, A. P., ... Clarke, L. P. (2011, jan). 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), 915–931. Retrieved from `https://doi.org/10.1118/1.3528204` doi: 10.1118/1.3528204

Armato, S. G., Roberts, R. Y., Kocherginsky, M., Aberle, D. R., Kazerooni, E. A., MacMahon, H., ... Clarke, L. P. (2009, jan). Assessment of radiologist performance in the detection of lung nodules. *Academic Radiology*, *16*(1), 28–38. Retrieved from `https://doi.org/10.1016/j.acra.2008.05.022` doi: 10.1016/j.acra.2008.05.022

Armato III, S. G., McLennan, G., Bidaut, L., McNitt-Gray, M. F., Meyer, C. R., Reeves,

A. P., ... Clarke, L. P. (2015). *Data from lidc-idri.* The Cancer Imaging Archive. doi: 10.7937/k9/tcia.2015.lo9ql9sx

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of compstat 2010* (pp. 177–186). Physica-Verlag HD. Retrieved from https://doi.org/10.1007/978-3-7908-2604-3_16 doi: 10.1007/978-3-7908-2604-3_16

Brenner, D. J., & Hall, E. J. (2007, nov). Computed tomography — an increasing source of radiation exposure. *New England Journal of Medicine*, *357*(22), 2277–2284. Retrieved from https://doi.org/10.1056/nejmra072149 doi: 10.1056/nejmra072149

Campos, D. M., Simões, A., Ramos, I., & Campilho, A. (2014). Feature-based supervised lung nodule segmentation. In *IFMBE proceedings* (pp. 23–26). Springer International Publishing. Retrieved from https://doi.org/10.1007/978-3-319-03005-0_7 doi: 10.1007/978-3-319-03005-0_7

Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2016). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, *abs/1606.00915*. Retrieved from http://arxiv.org/abs/1606.00915

Chollet, F., et al. (2015). *Keras.* https://github.com/fchollet/keras. GitHub.

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., & Ronneberger, O. (2016). 3d u-net: Learning dense volumetric segmentation from sparse annotation. *CoRR*, *abs/1606.06650*. Retrieved from http://arxiv.org/abs/1606.06650

Clark, K., Vendt, B., Smith, K., Freymann, J., Kirby, J., Koppel, P., ... Prior, F. (2013, jul). The cancer imaging archive (TCIA): Maintaining and operating a public information repository. *Journal of Digital Imaging*, *26*(6), 1045–1057. Retrieved from https://doi.org/10.1007/s10278-013-9622-7 doi: 10.1007/s10278-013-9622-7

Dehmeshki, J., Amin, H., Valdivieso, M., & Ye, X. (2008, apr). Segmentation of pulmonary nodules in thoracic CT scans: A region growing approach. *IEEE Transactions on Medical Imaging*, *27*(4), 467–480. Retrieved from https://doi.org/10.1109/tmi.2007.907555 doi: 10.1109/tmi.2007.907555

de Hoop, B., Gietema, H., van de Vorst, S., Murphy, K., van Klaveren, R. J., & Prokop, M. (2010). Pulmonary ground-glass nodules: Increase in mass as an early indicator of growth. *Radiology*, *255*(1), 199-206. Retrieved from https://doi.org/10.1148/radiol.09090571 (PMID: 20123896) doi: 10.1148/radiol.09090571

Diciotti, S., Lombardo, S., Falchini, M., Picozzi, G., & Mascalchi, M. (2011, dec). Automated segmentation refinement of small lung nodules in CT scans by local shape analysis. *IEEE Transactions on Biomedical Engineering*, *58*(12), 3418–3428. Retrieved from https://doi.org/10.1109/tbme.2011.2167621 doi: 10.1109/tbme.2011.2167621

Dumoulin, V., & Visin, F. (2016, March). A guide to convolution arithmetic for deep learning. *ArXiv e-prints*.

Eisenhauer, E., Therasse, P., Bogaerts, J., Schwartz, L., Sargent, D., Ford, R., ... Verweij, J. (2009, jan). New response evaluation criteria in solid tumours: Revised RECIST guideline (version 1.1). *European Journal of Cancer*, *45*(2), 228–247. Retrieved from https://doi.org/10.1016/j.ejca.2008.10.026 doi: 10.1016/j.ejca.2008.10.026

Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2015, January). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, *111*(1), 98–136.

Farabet, C., Couprie, C., Najman, L., & LeCun, Y. (2013, aug). Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1915–1929. Retrieved from https://doi.org/10.1109/tpami.2012.231 doi: 10.1109/tpami.2012.231

Farag, A. A., Munim, H. E. A. E., Graham, J. H., & Farag, A. A. (2013, dec). A novel approach for lung nodules segmentation in chest CT using level sets. *IEEE Transactions on Image Processing*, *22*(12), 5202–5213. Retrieved from https://doi.org/10.1109/tip.2013.2282899 doi: 10.1109/tip.2013.2282899

Hancock, M. C., & Magnan, J. F. (2016, dec). Lung nodule malignancy classification using only radiologist-quantified image features as inputs to statistical learning algorithms: probing the lung image database consortium dataset with two statistical learning methods. *Journal of Medical Imaging*, *3*(4), 044504. Retrieved from https://doi.org/10.1117/1.jmi.3.4.044504 doi: 10.1117/1.jmi.3.4.044504

Hawkins, D. M. (2004, jan). The problem of overfitting. *Journal of Chemical Information*

and Computer Sciences, *44*(1), 1–12. Retrieved from https://doi.org/10.1021/ci0342472 doi: 10.1021/ci0342472

He, K., Gkioxari, G., Dollár, P., & Girshick, R. B. (2017). Mask R-CNN. *CoRR*, *abs/1703.06870*. Retrieved from http://arxiv.org/abs/1703.06870

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, *abs/1512.03385*. Retrieved from http://arxiv.org/abs/1512.03385

Henschke, C. I., Yankelevitz, D. F., Mirtcheva, R., McGuinness, G., McCauley, D., & Miettinen, O. S. (2002, may). CT screening for lung cancer. *American Journal of Roentgenology*, *178*(5), 1053–1057. Retrieved from https://doi.org/10.2214/ajr.178.5.1781053 doi: 10.2214/ajr.178.5.1781053

Heuvelmans, M. A., Oudkerk, M., de Bock, G. H., de Koning, H. J., Xie, X., van Ooijen, P. M. A., ... Vliegenthart, R. (2013, mar). Optimisation of volume-doubling time cutoff for fast-growing lung nodules in CT lung cancer screening reduces false-positive referrals. *European Radiology*, *23*(7), 1836–1845. Retrieved from https://doi.org/10.1007/s00330-013-2799-9 doi: 10.1007/s00330-013-2799-9

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., ... others (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97.

Ho, P.-G. (Ed.). (2011). *Image segmentation*. InTech. Retrieved from https://doi.org/10.5772/628 doi: 10.5772/628

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, *abs/1502.03167*. Retrieved from http://arxiv.org/abs/1502.03167

Jaderberg, M., Simonyan, K., Zisserman, A., & Kavukcuoglu, K. (2015). Spatial transformer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems 28* (pp. 2017–2025). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/5854-spatial-transformer-networks.pdf

Kaggle. (2017). *Dstl Satellite Imagery Feature Detection.* https://www.kaggle.com/c/dstl-satellite-imagery-feature-detection. (Accessed:

2017-06-10)

Karpathy, A. (2016). *Connecting images and natural language* (Unpublished doctoral dissertation). Stanford University.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, *abs/1412.6980*. Retrieved from http://arxiv.org/abs/1412.6980

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Krogh, A., & Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in neural information processing systems* (pp. 950–957).

Kubo, T., Ohno, Y., Takenaka, D., Nishino, M., Gautam, S., Sugimura, K., ... Hatabu, H. (2016). Standard-dose vs. low-dose CT protocols in the evaluation of localized lung lesions: Capability for lesion characterization—iLEAD study. *European Journal of Radiology Open*, *3*, 67–73. Retrieved from https://doi.org/10.1016/j.ejro.2016.03.002 doi: 10.1016/j.ejro.2016.03.002

Kubota, T., Jerebko, A. K., Dewan, M., Salganicoff, M., & Krishnan, A. (2011, feb). Segmentation of pulmonary nodules of various densities with morphological approaches and convexity models. *Medical Image Analysis*, *15*(1), 133–154. Retrieved from https://doi.org/10.1016/j.media.2010.08.005 doi: 10.1016/j.media.2010.08.005

Kuhnigk, J.-M., Dicken, V., Bornemann, L., Bakai, A., Wormanns, D., Krass, S., & Peitgen, H.-O. (2006, apr). Morphological segmentation and partial volume analysis for volumetry of solid pulmonary lesions in thoracic CT scans. *IEEE Transactions on Medical Imaging*, *25*(4), 417–434. Retrieved from https://doi.org/10.1109/tmi.2006.871547 doi: 10.1109/tmi.2006.871547

Kurihara, Y., Matsuoka, S., Yamashiro, T., Fujikawa, A., Matsushita, S., Yagihashi, K., & Nakajima, Y. (2014, mar). MRI of pulmonary nodules. *American Journal of Roentgenology*, *202*(3), W210–W216. Retrieved from https://doi.org/10.2214/ajr.13.11618 doi: 10.2214/ajr.13.11618

Lapedes, A., & Farber, R. (1987). How neural nets work. In *Proceedings of the 1987 international conference on neural information processing systems* (pp. 442–456).

Cambridge, MA, USA: MIT Press. Retrieved from http://dl.acm.org/citation.cfm?id=2969644.2969691

Lassen, B. C., Jacobs, C., Kuhnigk, J.-M., van Ginneken, B., & van Rikxoort, E. M. (2015, jan). Robust semi-automatic segmentation of pulmonary subsolid nodules in chest computed tomography scans. *Physics in Medicine and Biology*, *60*(3), 1307–1323. Retrieved from https://doi.org/10.1088/0031-9155/60/3/1307 doi: 10.1088/0031-9155/60/3/1307

LeCun, Y., Bengio, Y., & Hinton, G. (2015, May 28). Deep learning. *Nature*, *521*(7553), 436-444. Retrieved from http://dx.doi.org/10.1038/nature14539 (Insight)

Li, B., Chen, Q., Peng, G., Guo, Y., Chen, K., Tian, L., ... Wang, L. (2016, may). Segmentation of pulmonary nodules using adaptive local region energy with probability density function-based similarity distance and multi-features clustering. *BioMedical Engineering OnLine*, *15*(1). Retrieved from https://doi.org/10.1186/s12938-016-0164-3 doi: 10.1186/s12938-016-0164-3

MacMahon, H., Naidich, D. P., Goo, J. M., Lee, K. S., Leung, A. N. C., Mayo, J. R., ... Bankier, A. A. (2017). Guidelines for management of incidental pulmonary nodules detected on ct images: From the fleischner society 2017. *Radiology*, *284*(1), 228-243. Retrieved from https://doi.org/10.1148/radiol.2017161659 (PMID: 28240562) doi: 10.1148/radiol.2017161659

Matthew C. Hancock. (2016). *Pylidc - An object relational mapping for the LIDC dataset using sqlalchemy.* https://github.com/pylidc/pylidc/. (Accessed: 2017-06-10)

McKinley, S., & Levine, M. (1998). Cubic spline interpolation. *College of the Redwoods*, *45*(1), 1049–1060.

McNitt-Gray, M. F., Armato, S. G., Meyer, C. R., Reeves, A. P., McLennan, G., Pais, R. C., ... Clarke, L. P. (2007, dec). The lung image database consortium (LIDC) data collection process for nodule detection and annotation. *Academic Radiology*, *14*(12), 1464–1474. Retrieved from https://doi.org/10.1016/j.acra.2007.07.021 doi: 10.1016/j.acra.2007.07.021

Messay, T., Hardie, R. C., & Rogers, S. K. (2010, jun). A new computationally efficient CAD system for pulmonary nodule detection in CT imagery. *Medical Image*

Analysis, *14*(3), 390–406. Retrieved from https://doi.org/10.1016/j.media.2010.02.004 doi: 10.1016/j.media.2010.02.004

Messay, T., Hardie, R. C., & Tuinstra, T. R. (2015, may). Segmentation of pulmonary nodules in computed tomography using a regression neural network approach and its application to the lung image database consortium and image database resource initiative dataset. *Medical Image Analysis*, *22*(1), 48–62. Retrieved from https://doi.org/10.1016/j.media.2015.02.002 doi: 10.1016/j.media.2015.02.002

National Cancer Institute. (2014). *National Lung Screening Trial.* https://www.cancer.gov/types/lung/research/nlst. (Accessed: 2017-05-11)

National Cancer Institute. (2016). *Can Lung Cancer Be Found Early?* https://www.cancer.org/cancer/lung-cancer/prevention-and-early-detection/early-detection.html. (Accessed: 2017-05-11)

Okada, K., & Akdemir, U. (2005). Blob segmentation using joint space-intensity likelihood ratio test: application to 3d tumor segmentation. In *Computer vision and pattern recognition, 2005. cvpr 2005. ieee computer society conference on* (Vol. 2, pp. 437–444).

payaovat. (2017). *Cats.* https://pixabay.com/en/cat-eyes-face-animals-home-2438095/. Pixabay.

Perandini, S., Soardi, G. A., Motton, M., Augelli, R., Zantedeschi, L., & Montemezzi, S. (2016, feb). CT imaging features in the characterization of non-growing solid pulmonary nodules in non-smokers. *Polish Journal of Radiology*, *81*, 46–50. Retrieved from https://doi.org/10.12659/pjr.895307 doi: 10.12659/pjr.895307

Pinheiro, P., & Collobert, R. (2014). Recurrent convolutional neural networks for scene labeling. In T. Jebara & E. P. Xing (Eds.), *Proceedings of the 31st international conference on machine learning (icml-14)* (p. 82-90). JMLR Workshop and Conference Proceedings. Retrieved from http://jmlr.org/proceedings/papers/v32/pinheiro14.pdf

Rahman, M. A., & Wang, Y. (2016). Optimizing intersection-over-union in deep neural networks for image segmentation. In *Advances in visual computing* (pp. 234–

244). Springer International Publishing. Retrieved from https://doi.org/10.1007/978-3-319-50835-1_22 doi: 10.1007/978-3-319-50835-1_22

Revel, M.-P., Bissery, A., Bienvenu, M., Aycard, L., Lefort, C., & Frija, G. (2004, may). Are two-dimensional CT measurements of small noncalcified pulmonary nodules reliable? *Radiology*, *231*(2), 453–458. Retrieved from https://doi.org/10.1148/radiol.2312030167 doi: 10.1148/radiol.2312030167

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *CoRR*, *abs/1505.04597*. Retrieved from http://arxiv.org/abs/1505.04597

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252. Retrieved from http://dx.doi.org/10.1007/s11263-015-0816-y doi: 10.1007/s11263-015-0816-y

Schmidhuber, J. (2014). Deep learning in neural networks: An overview. *CoRR*, *abs/1404.7828*. Retrieved from http://arxiv.org/abs/1404.7828

Setio, A. A. A., Traverso, A., de Bel, T., Berens, M. S., van den Bogaard, C., Cerello, P., ... Jacobs, C. (2017, dec). Validation, comparison, and combination of algorithms for automatic detection of pulmonary nodules in computed tomography images: The LUNA16 challenge. *Medical Image Analysis*, *42*, 1–13. Retrieved from https://doi.org/10.1016/j.media.2017.06.015 doi: 10.1016/j.media.2017.06.015

Shelhamer, E., Long, J., & Darrell, T. (2016). Fully convolutional networks for semantic segmentation. *CoRR*, *abs/1605.06211*. Retrieved from http://arxiv.org/abs/1605.06211

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., ... Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, *529*, 484–503. Retrieved from http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. A. (2014). Striving for simplicity: The all convolutional net. *CoRR*, *abs/1412.6806*. Retrieved from http://arxiv.org/abs/1412.6806

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014).

Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, *15*(1), 1929–1958.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *CoRR*, *abs/1409.3215*. Retrieved from http://arxiv.org/abs/1409.3215

Swensen, S. J., Viggiano, R. W., Midthun, D. E., Müller, N. L., Sherrick, A., Yamashita, K., ... Weaver, A. L. (2000, jan). Lung nodule enhancement at CT: Multicenter study. *Radiology*, *214*(1), 73–80. Retrieved from https://doi.org/10.1148/radiology.214.1.r00ja1473 doi: 10.1148/radiology.214.1.r00ja1473

University of Oxford. (2012). *PASCAL VOC Challenge performance evaluation and download server*. http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11&compid=6. (Accessed: 2017-05-10)

van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011, mar). The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, *13*(2), 22–30. Retrieved from https://doi.org/10.1109/mcse.2011.37 doi: 10.1109/mcse.2011.37

van Ginneken, B., Armato, S. G., de Hoop, B., van Amelsvoort-van de Vorst, S., Duindam, T., Niemeijer, M., ... Prokop, M. (2010, dec). Comparing and combining algorithms for computer-aided detection of pulmonary nodules in computed tomography scans: The ANODE09 study. *Medical Image Analysis*, *14*(6), 707–722. Retrieved from https://doi.org/10.1016/j.media.2010.05.005 doi: 10.1016/j.media.2010.05.005

Veit, A., Wilber, M. J., & Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems 29* (pp. 550–558). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/6556-residual-networks-behave-like-ensembles-of-relatively-shallow-networks.pdf

Wang, Q., Song, E., Jin, R., Han, P., Wang, X., Zhou, Y., & Zeng, J. (2009, jun). Segmentation of lung nodules in computed tomography images using dynamic programming and multidirection fusion techniques1. *Academic Radiology*, *16*(6), 678–688. Retrieved from https://doi.org/10.1016/j.acra.2008.12.019 doi: 10.1016/j.acra.2008.12.019

Way, T. W., Hadjiiski, L. M., Sahiner, B., Chan, H.-P., Cascade, P. N., Kazerooni, E. A., ... Zhou, C. (2006, jun). Computer-aided diagnosis of pulmonary nodules on CT scans: Segmentation and classification using 3d active contours. *Medical Physics*, *33*(7Part1), 2323–2337. Retrieved from https://doi.org/10.1118/1.2207129 doi: 10.1118/1.2207129

World Health Organization. (2017). *Cancer Fact sheet.* http://www.who.int/mediacentre/factsheets/fs297/en/. (Accessed: 2017-05-10)

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., & Brostow, G. J. (2016). Harmonic networks: Deep translation and rotation equivariance. *CoRR*, *abs/1612.04642*. Retrieved from http://arxiv.org/abs/1612.04642

Yu, F., & Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *CoRR*, *abs/1511.07122*. Retrieved from http://arxiv.org/abs/1511.07122

Zhao, B., Yankelevitz, D., Reeves, A., & Henschke, C. (1999, jun). Two-dimensional multi-criterion segmentation of pulmonary nodules on helical CT images. *Medical Physics*, *26*(6), 889–895. Retrieved from https://doi.org/10.1118/1.598605 doi: 10.1118/1.598605

Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2016). Pyramid scene parsing network. *CoRR*, *abs/1612.01105*. Retrieved from http://arxiv.org/abs/1612.01105