Lung Cancer Detection using Deep Learning

Thesis · April 2018

DOI: 10.13140/RG.2.2.33602.27841

Jelo Salomon
Technological University Dublin - City Campus
1 PUBLICATION **0** CITATIONS

Lung Cancer Detection using Deep Convolutional Networks
Final Year Project Report

DT2282
BSc in Computer Science

Jelo Salomon
Bianca Schoen Phelan

School of Computing
Dublin Institute of Technology

13/04/2018

Contents

List of Figures

Listings

List of Tables

Abstract

Pulmonary cancer also known as lung carcinoma is the leading cause for cancer-related death in the world. Early stage detection cancer detection using computed tomography (CT) could save hundreds of thousands of lives every year. However analysing hundreds of thousands of these scans are an enormous burden for radiologists and too often they suffer from observer fatigue which can reduce their performance. Therefore, a need to read, detect and provide an evaluation of CT scans efficiently exists.

In this paper, the author proposes a method of detecting lung cancer in a CT scan using a 2D-UNet model on a web application. The author cropped 2D cancer masks on its reference image using the center of the lung cancer given in the dataset and trained a model with different techniques and hyperparameters. Finally the result is evaluated using a dice coefficient and confusion matrix metrics. The author reaches a 65.7% accuracy on the dice coefficient and an average 0.88% true positive rate and 0.71% false positive rate on a test set of positive and negative samples.

Declaration

I declare that this thesis, which I submit to Dublin Institute of Technology for examination in consideration of the award of a bachelors degree BSc.Computer Science International is my own personal effort. Where any of the content presented is the result of my own hard work and research into the topic area which includes existing research that has been already conducted.

This is duly acknowledged in the text such that it is possible to ascertain how much of the work is my own. I have not already obtained a degree in Dublin Institute of Technology or elsewhere on the basis of this work. Furthermore, I took reasonable care to ensure that the work is original, and, to the best of my knowledge, does not breach copyright law, and has not been taken from other sources except where such work has been cited and acknowledged within the text.

SIGNED: Jelo Salomon
Jelo Salomon,

Acknowledgements

This academic year has been an incredible year for learning. I'd like to give a special shout out these people who have provided incredible support:

Firstly towards my parents for their patience and support throughout the entire year, without them I would not be able to finish this project.

Carla for always being there to support me since the beginning.

My friends and college mates who generously gave their input, advice and times helped me debug a certain problem when I was blindly frustrated.

My supervisor Bianca for meeting with me every week to discuss the project and help steer me in the right direction.

Dr.Seamus Linanne for taking time out of his busy schedule and meet with me to gain feedback, correspond and share medical expertise.

Chapter 1

Introduction

1.1 Introduction

This section aims to give a brief overview into the project, the goals of the project, the challenges faced and the structure of the report.

1.2 Project Overview

1.2.1 Lung Cancer Detection Application

A web application has been developed to demonstrate a proof of concept. The application requires a user to upload a CT Scan. The application then processes the file and displays the images to the user. The user then chooses which scan he or she wants to predict then the application pre-processes the CT Scan and infers the image to the predictive model. The output of the model is then displayed to the user. The user has the choice to view the images via a carousel or a gallery mode.

1.2.2 Research Element

This report contains many aspects of research that support deep learning's ability to find lung cancer within CT Scans. The data used to train the model was gathered from a deep learning competition called LUNA 2016 which are CT Scans that have been annotated by radiologists. The radiologists determined whether a nodule is lung cancer or not and this location has been specified in 3D coordinates.

Deep Learning research has also been conducted to ensure that the correct architecture would be used to to train the model. The architecture used is called U-Net. It acts as a deep learning segmenter of images. Using our evaluation methods we could see that the deep learning architecture is able to segment regions of interest relative to it's accuracy but found that the model creates many false positives which means that the model required more regularization and training time, although it was trained for 40 hours.

1.3 Project Goals

The completed project should accomplish the following Objectives:

- **Self Education on Deep Learning** A large part of this project contains a lot of self education, initially the author knew very little about deep learning, as part of this project the author should have a good grasp on deep learning concepts.
- **User Research and Evaluation** The project should have a user-centred design aspect. This means that the system should be designed to help certain users. In this project the user would be medical professionals who work in diagnosing lung cancer.
- **System: Upload CT Scans** The system should be capable of getting CT Scans from Users that will be utilized by the Deep Learning Model.
- **System: Detection of Lung Cancer** The system should be able to detect the lung cancer within the CT scan images that users have uploaded.
- **System: Display Results** The system should be able to give information that our user can appropriately understand and gain insight from it.

1.4 Project Challenges

This section discusses the challenges that were overcome to complete this project. Each of the challenges are briefly introduced and detailed in later sections of the report. This section can be useful for those who aim to do a deep learning project with large image datasets.

1.4.1 Large 3D Dataset Management and Preprocessing

Working with the dataset is the most difficult challenge that had to be overcome for this project. The dataset is very large (around 70 Gigabytes) which makes managing and analysing the dataset and training the model very computationally heavy. Figure 1.1 shows one instance of a patient's CT Scan which is approximately 60 Megabytes. A potential solution to this is to migrate the computation workload to a cloud service such as Google Cloud, AWS or Floydhub which could result in a more efficient work flow for the project. A single CT scan is also 3 Dimensional which can be complex to work with especially during feature selection and data preparation.

1.4.2 Neural Network Architecture Design Choices

Neural networks are architectures that are designed in of itself and there are many types of architectures out there that exist to solve different problems. A key challenge of the author is to use an architecture that is able to find malignant tumour patterns in the data. To overcome this, the author has to undertake necessary research to implement the correct model design prior to training. There are many neural network architectures such as multi-layer perceptrons, convolutional networks and sequence models.

1.4.3 Understanding Deep Learning Concepts

Deep learning is an exciting but new concept for the author. To overcome this challenge the author has performed research and study to deliver the project goals. A useful resource that has been used is Stanford's professor Andrew Ng's deep learning course on Coursera. This course enables the learners to gain a foundation to deep learning.

1.4.4 Training the Model

Deep learning models require hours of training time, the best performing model that has been trained in this project required 40 hours of training time on a Tesla K80 GPU available on Floydhub. Compared to standard machine learning algorithms, performing experiments and getting results take much longer. This problem can stop the project's progress. Careful planning before running experiments can minimize this problem. Using keras.Callback features such as CSVLogger, ModelCheckpoint, LearningRateScheduler and etc can be used to debug the models quicker when models don't improve which directly increase productivity.

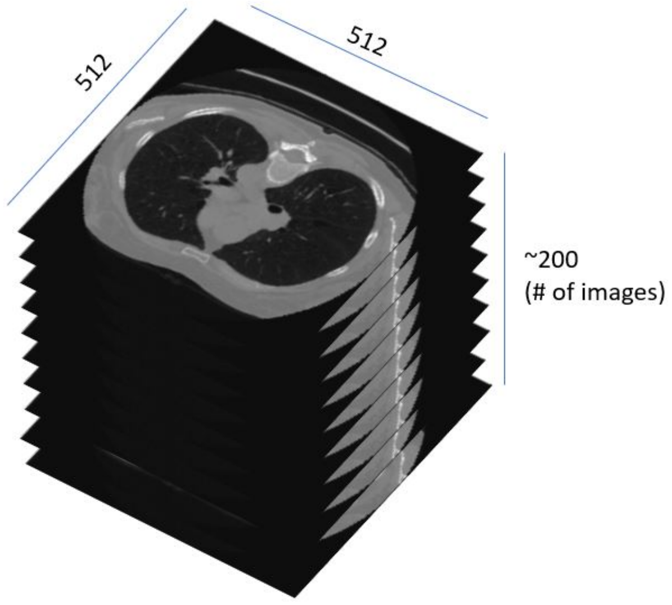


Figure 1.1: One Instance of a CT Scan Image in Kaggle Dataset

1.4.5 Deep Learning Integration

Integrating deep learning models into applications using Python is different compared to standard machine learning algorithms. Standard machine learning algorithms can be serialized into a file and loaded into an application. Deep learning models use Tensorflow as a framework. Integrating the model requires additional code to ensure that the model runs sequentially on the same thread as the application. This is further explained in the implementation chapter.

1.5 Report Structure

The research chapter details research about lung cancer, impact statistics, existing solutions, challenges that medical professionals face, technologies used and the dataset for the project. The main aim of this chapter is to introduce how lung cancer is an incredibly complex problem and that the current ways its being diagnosed.

The deep learning chapter deals with the Deep Learning intuitions needed to develop and train the deep learning model. This chapter introduces deep learning concepts including the research papers associated with the concepts to help debug neural network issues and identify problems during training.

The approach and methodology chapter deals with standard practice used to deliver the project. The two mentioned is Agile Software Development and CRISP-DM. The idea of this chapter is to ensure that the project is managed using standard industry practices.

The design chapter deals with the design methodologies used, user centred design artefacts such as persona, scenarios, design wireframes and technical design artefacts such as use case diagrams and system architecture designs. The goal of this chapter is to demonstrate what kind of user would be using this application and how it is specifically designed for that type of user.

The implementation chapter details the process of creating the project, following the approach and methodology, adhering to the designs created and performing deep learning experiments drawing from relevant research aforementioned.

The project plan chapter outlines how the project has evolved since the interim throughout the entire development process and how the tasks were estimated.

The conclusion chapter contains results gained, a proof of concept evaluation, future and final thoughts about the project overall.

1.6 Conclusion

The project integrates different topics in Computer Science to try and solve a real world problem in the medical domain. The project consists of data mining and software development to deliver a proof of concept. The application is a lung cancer detection system to help doctors make better and informed decisions when diagnosing lung cancer.

In the next chapter, the author outlines the relevant research for the project.

Chapter 2

Research

2.1 Research Overview

In this chapter the author discuss the research that has been undertaken. In this chapter, relevant research from different sources has been collated to demonstrate knowledge in the medical domain, specifically lung cancer detection. This chapter also contains research regarding technology and data that has been used.

2.2 General Statistics

According to Jamal et al[28]currently Lung Cancer is one of the most common cancer and according to the American Cancer Society in 2017 there is estimated to be 222,500 new cases, 155,870 deaths. The incidence rate is 62.7% per 100,000 and death rate tends to be around 44.7 in the US population. It is clear that even though the death rate has gone down over the years from the highest 90.6% to 44.7% in 1987 it is still a prevalent disease to this day.

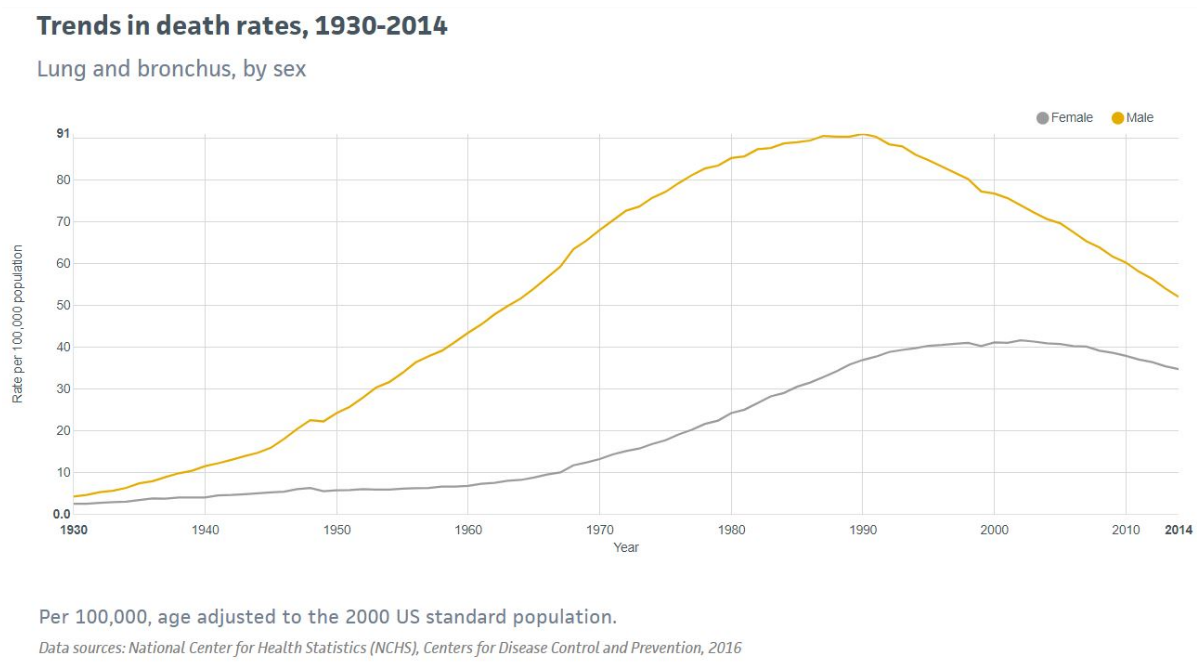


Figure 2.1: Death Rate Trend of Lung Cancer in the US

2.3 Doctor's Challenges during Diagnosis

The author has met with Dr. Seamus Linnane who is a respiratory physician at Beaumont Hospital. Dr. Linanne shared his valuable insight into the current way of diagnosing lung cancer. Dr. Linanne is part of a multidisciplinary team involving radiologists, oncologist, surgeons and other respiratory physicians and consultants at Beaumont Hospital to diagnose lung cancer.

This multidisciplinary team uses a variety of data, CT Scans, X-Rays, Pet Scans and Biopsies to assess whether a patient has lung cancer. These tests help the team to fully diagnose a patient and the approach would be to use all of them to gather data.

According to Dr. Linanne, In 2016, 134 patients were diagnosed with lung cancer at Beaumont Rapid Access Clinic where 217 patients were diagnosed in total across all hospital services. This means that approximately 1/3 of all referrals at the clinic have lung cancer in Beaumont Hospital.

Medical professionals use TNM classification to help characterise lung cancer from basic to advanced forms of malignant tumours[33]. IA being the most earliest stage of cancer which is more likely that it was accidentally discovered, the more difficult it will be to perform a biopsy and ultimately the better the prognosis (likelihood to survive). On the other hand, IV is an advanced stage of cancer which means that it is easier to diagnose, including biopsy, highly likely to cause symptoms and the worse the prognosis.

TNM 7 Classification	No. of Patients
IA	214
IB	72
IIA	53
IIB	64
IIIA	139
IIIB	71
IV	368
Total	**981**

Table 2.1: TNM 7 Classification Count in Beaumont Hospital 2016

Table 2.1 shows the TNM7 Lung Cancer classification diagnosis in 2016. The majority of people who get diagnosed already developed the most advanced form of lung cancer (IV). The second majority are in the earliest classification (IA), according to Dr.Linanne this group had accidentally been diagnosed with lung cancer while taking a CT Scan for another problem.

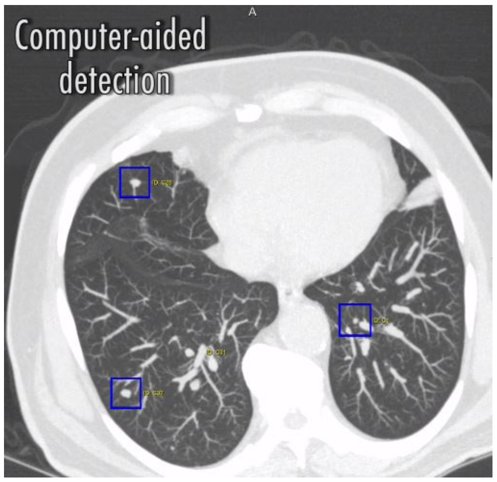
A main challenge with medical professionals when dealing with IA classification is that the tumours are so small less than 4 mm that it is very difficult to diagnose them via CT Scan images. To know for sure a biopsy needs to be conducted however this process can be very invasive and infeasible to do as IA tumours have around 50/50 chance to be benign/malignant. The general medical advice is to rescan in 6-12 weeks to see any evidence for growth which can grow more than double in size.[32]

Another challenge Doctor's face while analysing CT Scans is observer fatigue,According to Krupinski et al[30], after a day of clinical reading, radiologists have reduced ability to focus, increased symptoms of fatigue and oculomotor strain and reduced ability to detect fractures and further continues to say that radiologists need to be aware of the effects of fatigue on diagnostic accuracy and take steps to mitigate these effects.

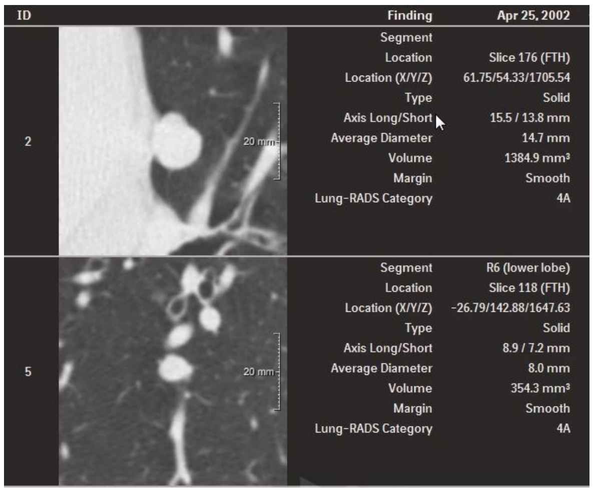
2.4 Existing Solutions for Lung Cancer Diagnostics

According to Mayo Clinic, In order to diagnose lung cancer, The recommended ways are:

- **Imaging tests** where an CT Scan on your lungs reveal abnormal mass or nodules.
- **Sputum Cytology** which analysing Phlegm (Mucus Cells).
- **Biopsy** which is the most invasive method where abnormal cells are removed from your body to be analysed. This method is done in a number of ways such as bronchoscopy in which an incision is made at your neck and surgical tools are inserted behind your breastbone to take tissue samples.[9]



(a) CADe: Computer-Aided Detection System



ID	Finding	Apr 25, 2002
2	Segment	
	Location	Slice 176 (FTH)
	Location (X/Y/Z)	61.75/54.33/1705.54
	Type	Solid
	Axis Long/Short	15.5 / 13.8 mm
	Average Diameter	14.7 mm
	Volume	1384.9 mm³
	Margin	Smooth
	Lung-RADS Category	4A
5	Segment	R6 (lower lobe)
	Location	Slice 118 (FTH)
	Location (X/Y/Z)	-26.79/142.88/1647.63
	Type	Solid
	Axis Long/Short	8.9 / 7.2 mm
	Average Diameter	8.0 mm
	Volume	354.3 mm³
	Margin	Smooth
	Lung-RADS Category	4A

(b) CADx: Computer-Aided Diagnosis Systems

Figure 2.2: Output of Computer Aided Lung Cancer Detection Systems

For this project, research on current tests on medical imaging for lung cancer detection has been undertaken. There exists computer aided detection systems that assist radiologists in detecting anomalies. According to Fermino et al[23], There are two main computational systems developed to assist radiologists, they are: CADe(Computer-Aided detection system)Figure 2.2a and CADx(computer-aided diagnosis system).Figure 2.2b. CADe detects abnormal mass in medical images while CADx aims to measure the characteristics of the once it has been detected.

2.5 Technology Decisions

2.5.1 Overview

In this section, the author details the technologies that he has used for this project. Although there are many tools that exist out there in the market, the author has found that these tools outlined perform well for the problem that needs to be solved.

2.5.2 Python

Python is a high level interpreted language used for general purpose programming. It is widely used for scientific computing and can be used for a wide variety of general tasks from data mining to software development. Python is the main language used for this project.

2.5.3 Anaconda

Anaconda is a popular data science platform where you can create data science projects and machine learning.[17]. Libraries such as NumPy, Pandas, Matplotlib, Tensorflow and etc come with Anaconda and IDE's such as Jupyter Notebook, Spyder and etc.

2.5.3.1 Numpy

Numpy is a library in Python that allows for efficient numerical computing in Python. This library is highly optimized to do mathematical tasks. In the project workflow Numpy is heavily used in data pre-processing and preparation One of the main features about Numpy is it's highly efficient n-dimensional array (ndarray). Compared to a list in Python a Numpy array can be n-dimensions and has more features associated with the ndarray. Numpy can also perform more efficient mathematical operations compared to the math library in Python.

2.5.3.2 Pandas

Pandas is also a library in Python, like numpy is also used for data pre-processing and preparation. One of the main features about pandas is the DataFrame and Series data structure. These data structures are optimized and contain fancy indexing that allow a variety of features such as reshaping, slicing, merging, joining and etc to be available.
Pandas and Numpy are extremely powerful when used together for manipulating data.

2.5.3.3 Matplotlib

Matplotlib is a Python plotting library that allows programmers to create a wide variety of graphs and visualizations with ease of use. The great feature about Matplotlib is that it integrates very well with Jupyter Notebook and creating visualizations is simplified. Matplotlib also works very well with pandas and numpy.

2.5.3.4 OpenCV

OpenCV (Open Source Computer Vision) is a well established computer vision library which is written in C/C++ and has been abstracted to interface with C++, Python and Java. This is a powerful tool when working with images and has a myriad of tools regarding image data manipulation, feature extraction and etc.

2.5.3.5 Tensorflow

Tensorflow is an open source deep learning library by Google. It was originally developed by Google's engineers who were working on Google Brain and has been used for research on machine learning and deep learning. Tensorflow at it's core is about computations of multidimensional arrays called tensors but what makes Tensorflow great is its ability to be flexible to deploy computations on different devices such as CPU's and GPU's [16].

2.5.3.6 Keras

Keras is also a Deep Learning Framework that abstracts much of the code in the other Frameworks like Tensorflow and Theano. Compared to the other frameworks Keras is more minimalist[20].

2.5.3.7 Jupyter Notebook IDE

The Anaconda distribution comes with a variety of software that includes Jupyter Notebooks for scientific computing. Jupyter Notebooks[14] is an open source software IDE that allows developers to create and share documents that contain live code and more.

2.5.4 FloydHub

Floydhub is a Deep Learning Platform in the Cloud[5]. Floydhub allows developers to run heavy tasks on the cloud such as training the deep learning model or heavy preprocessing tasks. The CPU's and GPU's available on Floydhub are fully configured to work on different Frameworks such as Tensorflow. One of the main benefits to Floydhub is that it is quite powerful and easy to use with easy to follow documentation.

2.5.5 Web Development

The goal for this project is to deliver a proof of concept application so that doctors can upload CT scans and gain predictions from a model. To do this, a web application has been developed with an integrated deep learning model. The web app was built using HTML, CSS, Javascript with a back end developed in Python using a micro framework called Flask.

2.5.5.1 Flask

Flask is a micro web framework that uses Python. It is extremely flexible, minimalist and easy to set up. Flask is also BSD Licensed which allows Flask to be further modified[18]. The great feature about Flask is that is very flexible and minimalist to use.

2.6 Dataset Research

2.6.1 Description

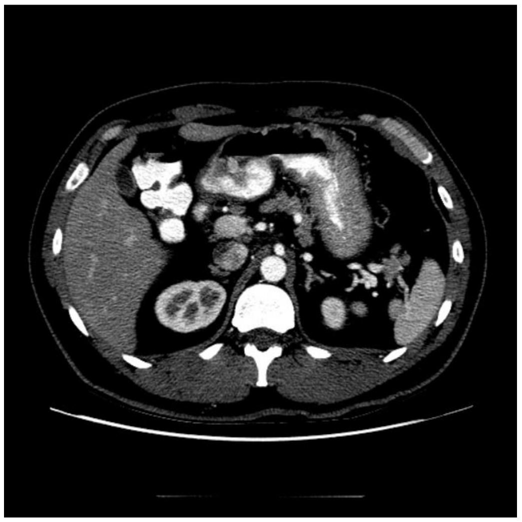
Building deep learning models require a lot of data. For this project datasets has been researched and identified before any real work has begun.
Since there is a heavy emphasis on building models for this project, a key part of the project relies on a Dataset. Prior to coding, I had to ensure I had a great dataset to work with to build a model. From doing my research there are 2 large datasets that I could work with.

2.6.2 Kaggle 3D Unlabeled Dataset: Data Science Bowl 2017

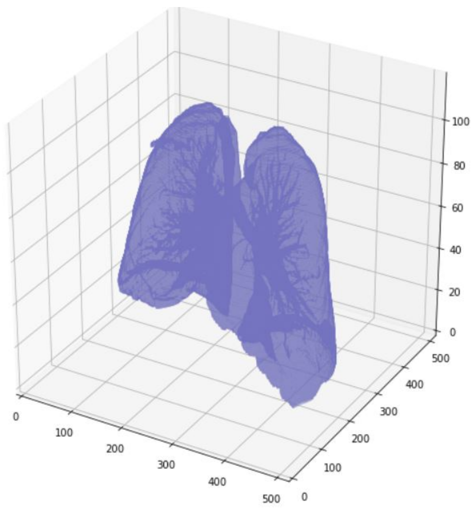
This dataset was part of the Kaggle competition Data Science Bowl 2017 [4]. The topic of the competition was about lung cancer detection. The dataset was provided by the National Lung Cancer Screening Trial, The Cancer Imaging Archive, Diagnostic Image Analysis Group (Radboud University), Lahey Hospital and Medical Center and Copenhagen University Hospital. The dataset contains full CT scan images of a patients lungs. The dimension for this is (512,512, 200) which is (Height, Width, No. of Images). See Figure 2.4a. For this project the dataset has been used to segment different parts of the CT scans as part of feature engineering and visualizations. See Figure 2.4b and Figure 2.4c. This dataset was what the author originally wanted to work with as the data was labeled as desired and useful for the project. However the largest challenge that hindered the author from continuing using this data is the size of the entire dataset. The entire dataset is about 100GB zipped which could not fit on the authors laptop. Preprocessing the entire dataset would also be too computationally heavy.

2.6.3 LUNA16 Labeled 3D - LUng Nodule Analysis Dataset 2016

The LUNA16 dataset is also 3D CT scans of lung cancer annotated by radiologists. The dataset contains 3D images and a CSV file containing annotations.This dataset was part of the LUNA16 Grand Challenge in 2016[38]. The dataset is still publicly available for research. Like the Kaggle dataset the format of the 3D Image is a 3 dimensional array (512,512, 200) (height,width,no. of images). See Figure 2.3a. The main advantage of this dataset is that the dataset is broken down into 10 subsets which makes it much easier to work with.



(a) Sample Image Slice



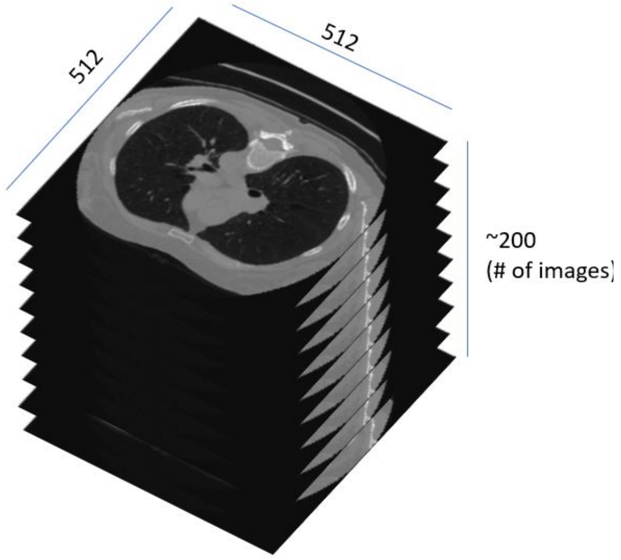
(b) Sample Instance Lungs Segmented

Figure 2.3: 3D LUNA16 Dataset 2016/[8]

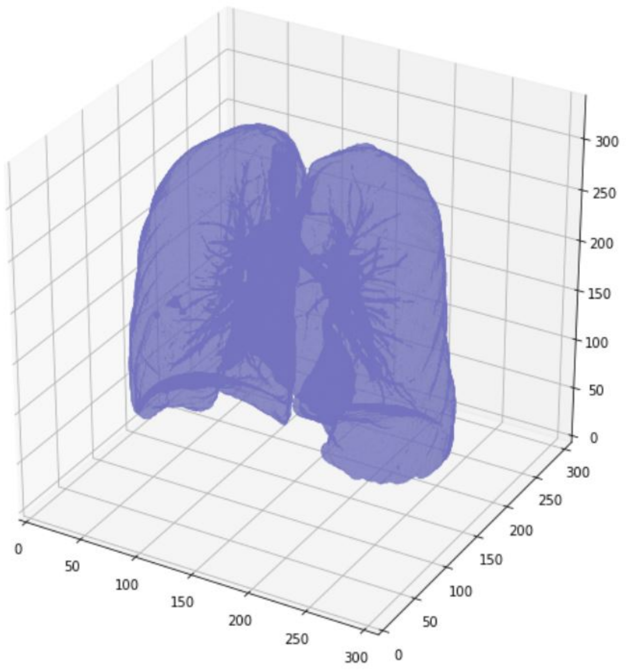
2.7 Conclusion

Lung cancer is an extremely complex problem to solve however with early detection a patient has a high increase of survivability. The diagnostics data suggests that the highest frequency of people who get diagnosed have already an advanced form of cancer and the second most frequent cohort are have been accidentally diagnosed and have the cancer that is hardest to determine and is easiest to cure. The research leads the author to believe that deep learning could be a powerful tool in diagnosing very small and very hard to determine nodules to aid medical decision making process.

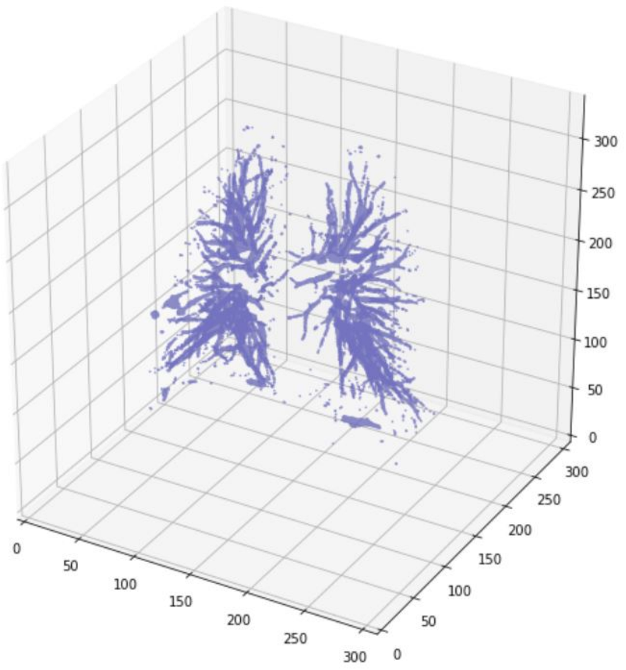
In the next chapter, the author details deep learning concepts that has been studied. The main goal of the next chapter is to demonstrate different intuitions in deep learning development.



(a) 3D Kaggle Dataset Format



(b) Segmented Lungs



(c) Segmented Nodules Region of Interest

Figure 2.4: 3D Kaggle Dataset 2017

Chapter 3

Deep Learning

3.1 Overview

According to Andrew Ng[34],The theory of Deep Learning has been around for decades but it has only been in the past few years that it has risen and taken off. In our society we have gone from having a relatively small amount of data to huge amounts to the point that we don't know what to do with them.

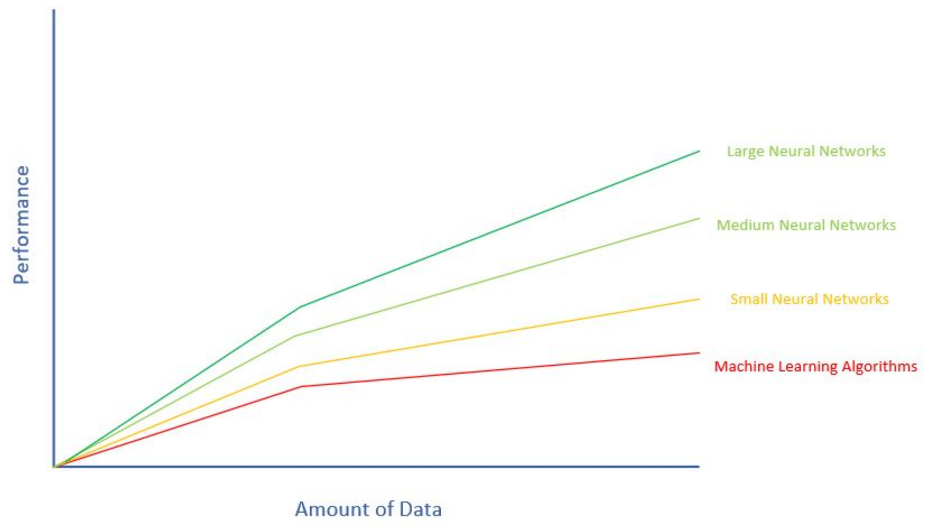


Figure 3.1: Performance of Neural Networks compared to Traditional Learning Algorithms by Andrew Ng[34]

According to Huang et al[26], Deep Learning has gained large popularity within different fields of interest ranging from a theoretical research in academia to practical application in industry. Newly emerging algorithms have expanded applications of Learning Machines to be able to compute large amounts of data with the help of hardware implementations and optimized parallel computational techniques available today. With this Deep Learning Architectures are able to be generalized, customized and configured to advance efficient and accurate models.

Compared to other techniques such as standard Machine Learning algorithms,Deep Learning is a preferred option as performance increases as the data scales[34]. Machine Learning algorithms such as Support Vector Machines, Random Forest and etc, performance plateaus when data increases in these algorithms. See Figure 3.1. Deep Learning Neural Networks works well for my lung cancer detection project as our dataset is large and can iteratively grow larger as more people infer new data to it which will result in a boost in performance.

In this chapter, the author explores the intuitions made into creation deep learning models.

3.2 Neural Network

A neural network is a model that has been inspired by the brain, the brain consists of nearly 10 billion neurons with 60 trillion connections between each other. A neuron consists of a cell body called the soma where the nucleus is found, many dendrites where input signals are received and transmitted and a synapse which is basically connections between neuron to neuron.[13]

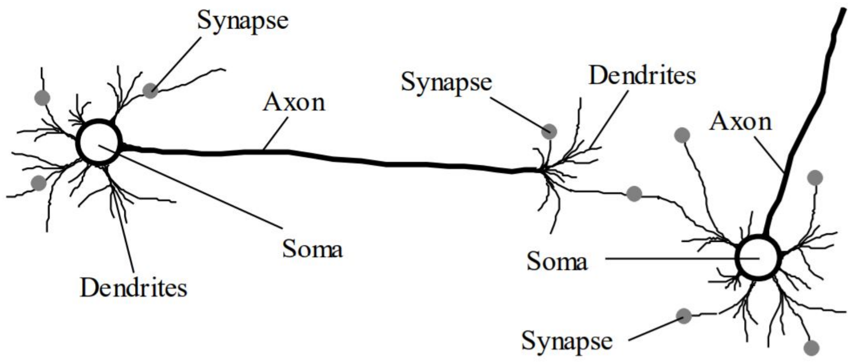


Figure 3.2: A Single Brain Neuron

3.2.1 Artificial Neuron

Like a brain neuron, neural networks consists of an artificial neuron. An artificial neuron consists of a weight to determine the strength of a connection, a linear function that needs to be computed and an activation function that computes the weighted sum of the linear function that is then compared to a threshold value.

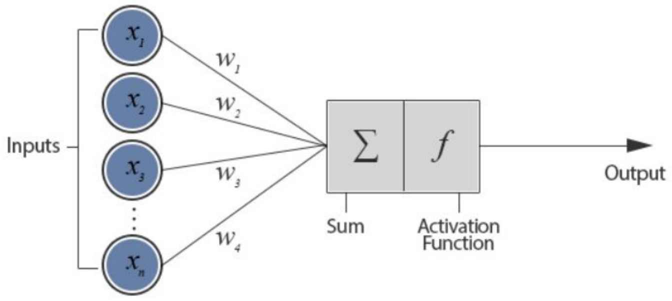


Figure 3.3: An Artificial Neuron also known as a Perceptron [27]

It is also worthy to note that artificial neurons like Figure 3.3 can be modified to contain different linear function algorithms such as Gradient Descent, Logistic Regression, etc and also different activation functions such as ReLU (Rectified Linear Unit) or TanH functions which depends on the problem to solve.

3.2.2 Artificial Neural Network

An Artificial Neural Network is an interconnected architecture where there exists an input layer where input data is placed, a hidden layer(s) where artificial neurons are stacked on on top of each other and an output layer where the prediction or classification is made.

3.2.3 Forward Propagation

Forward propagation is technique in which data moves through from the corresponding input layer, hidden layers and output layer sequentially.

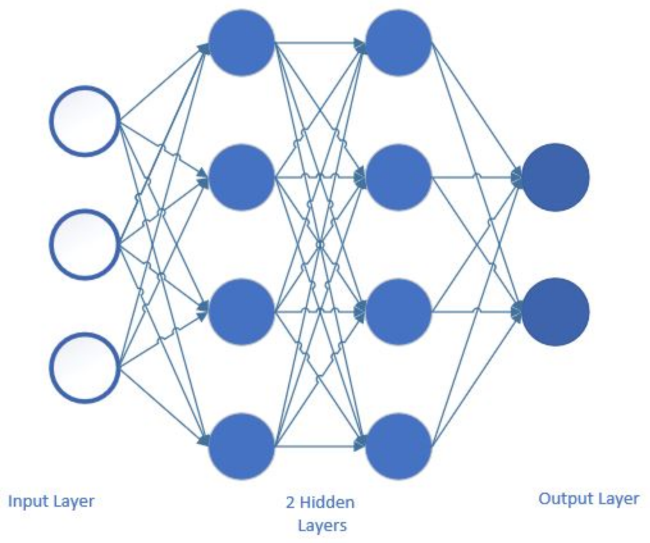


Figure 3.4: An Artificial Neural Network also known as a Multi Layer Perceptron

A hidden layer usually consists of an weight which is used in an optimization algorithm such as Gradient Descent, an activation function such as Sigmoid, TanH, LeakyReLU etc and a Loss Function to calculate the loss or error of the function which we use to back propagate to adjust the weights.

3.2.4 Back Propagation

Back propagation is the opposite of Forward propagation as it feeds the network backwards. Back propagation is used to adjust the weight of the neural network after the errors have been computed by forward propagation algorithms. In later sections of this chapter I detail how Gradient Descent is used in back propagation to adjust the weights of the neural network.

3.2.5 Activation Functions

Activation functions are an important part of a neural network. They allow neural networks to create non linear functions to solve problems. The 3 most used activation functions are Sigmoid, TanH and ReLU.
Activation functions allow or stop neurons from firing into the next layer by comparing it into the activation function thresholds.
Activation functions are used both in the forward and backward propagation where in the forward propagation an activation function is used to calculate the loss where the output of a function is compared to the a real number and in backward propagation they are used to update the parameters of the neural network.
Figure 3.5 shows commonly used activation functions in a neural network.

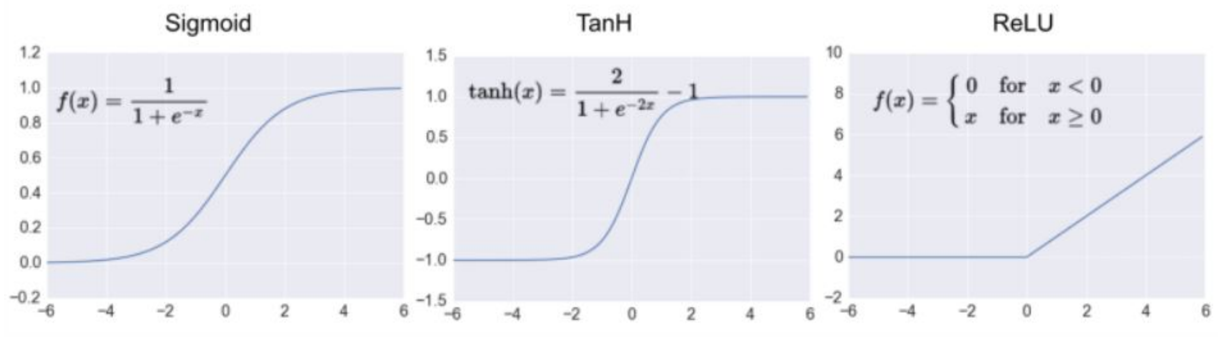
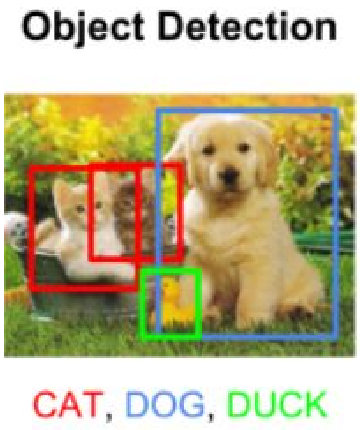


Figure 3.5: Common Activation Functions[11]

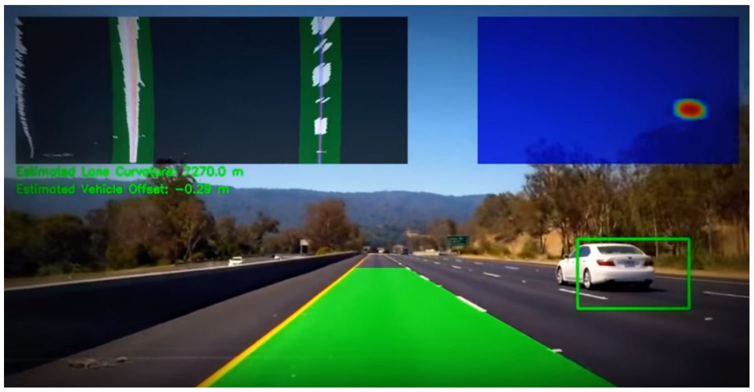
3.3 Convolutional Neural Networks

3.3.1 Overview

Convolutional Neural Networks is all about using Deep Learning with Computer Vision. A good way to gain intuition about this is to think about a Neural Network Architecture and how it is applied to visual tasks i.e. Images and Video. Convolutional Neural Networks have allowed us to create Facial Recognition, Object RecognitionFigure 3.6a, Self Driving Cars Figure 3.6b and more.



(a) Object Detection[12]


Estimated Lane Curvature: 270.0 m
Estimated Vehicle Offset: -0.29 m

(b) Self Driving Car: Lane Finding Detection[31]

Figure 3.6: Convolutional Neural Networks Examples

3.3.2 Architecture

Like a Neural Network, a typical Convolutional Neural Network consists of a multiple hidden layers called a Convolutional Layer where the linear function computes the strided convolutions over an image to extract features. It also consists of a pooling layer that computes another function such as Max Pool or Average Pool to reduce the size of the image in the neuron to speed up the computation. It does it by extracting the features of the neuron image and ignoring the rest, this makes the network more robust. There is also fully connected layer which is like a hidden layer in a neural network where the sum of the outputs of each layer are flattened and where each value is an input to the next layer followed by an activation function and an output[3]. See Figure 3.7.

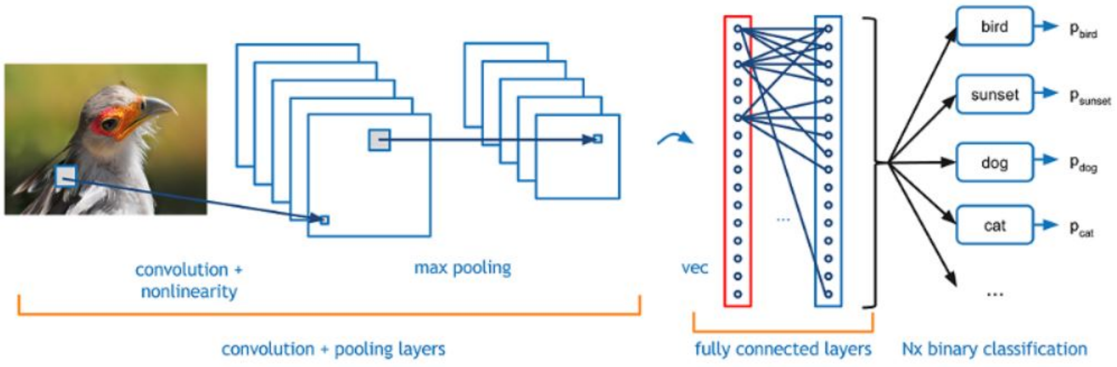


Figure 3.7: A Convolutional Neural Network Architecture Example [35]

3.3.3 Convolutional Layer

In a Convolutional Neural Network the linear function that is used is called a convolutional layer. Each node in the hidden layer extracts different features by using image processing feature detectors. For example, in the first layer, the first node may extract the horizontal edges of an image, the second node may extract vertical edges and etc. These features are extracted using a kernel. Figure 3.11 shows an example of how strided convolutions work on an image. The bottom is the original image and the top is the output of the convolutions. It is also worth noting that the output of the convolutions reduce the dimension of the original image.

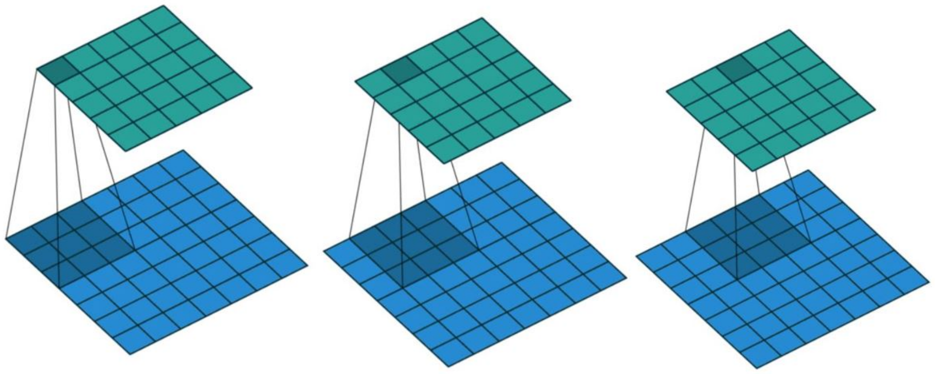


Figure 3.8: Strided Convolution Example[41]

3.3.4 Pooling Layer

The pooling layer happens tends to be computed after the convolutional layer. The reason why pooling is done is to further reduce the dimensions of the convolutional layer and just extract out the features to make the model more robust. There are two types of pooling done: max pooling and average pooling. Max pooling extracts out the highest pixel value out of a feature while average pooling calculates the average pixel value that has to be extracted. Figure 3.9 shows how the max and average pooling operation works.

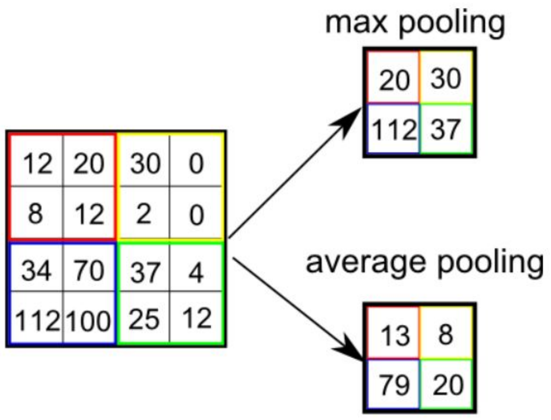
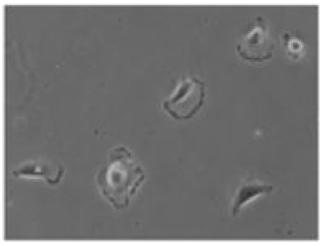


Figure 3.9: Max and Average Pooling Operation [2]

3.4 U-Net Model

3.4.1 Overview

A U-Net model is a different variation of a Convolutional Neural Network. It has been mainly used for biomedical image segmentation but I have also seen it being used in a variety of ways to segment regions of interest. See Figure 3.10. In this section the author details variant nature of U-Net models.


(a) U-Net Input Example

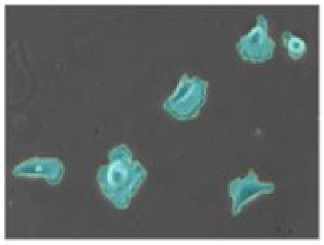

(b) U-Net Output Example

Figure 3.10: U-Net Segmentation

3.4.2 Architecture

Ronneberger et al 2015,[36] mentions that a U-Net model consists of a contracting path (left side). This is the same as standard Convolutional Networks outlined in the last section(3x3 Conv with ReLU and 2x2 Max Pooling). As earlier discussed, the great idea about this is that the network is able to create features necessary to detect patterns that we are looking to predict(In this case its lung cancer). The problem with this on its own is that convolutional layers downsample the size of the image. Hence data is lost as it feed forwards.

The unique design of the U-Net model lies in its expanding path (right side) which consists of up-convolutions (size 2x2) and merge layers. The up-convolutions take a downsampled sized image and expands the borders to increase the size of the image. This is done before the merge happens, the merge layers then takes output from an earlier convolution layer and concatenate it with the previous upsampled image. This concatenated data or 'Merge' is then fed through another convolution layer. The network is able to send data from an earlier network to a later one through skip connections. This is how the network is able to segment the images after it has found the features. See Figure 3.11,Figure 3.21 and Figure 3.22.

The last layer is a 1x1 convolutional layer to map feature vectors to a desired number of classes.

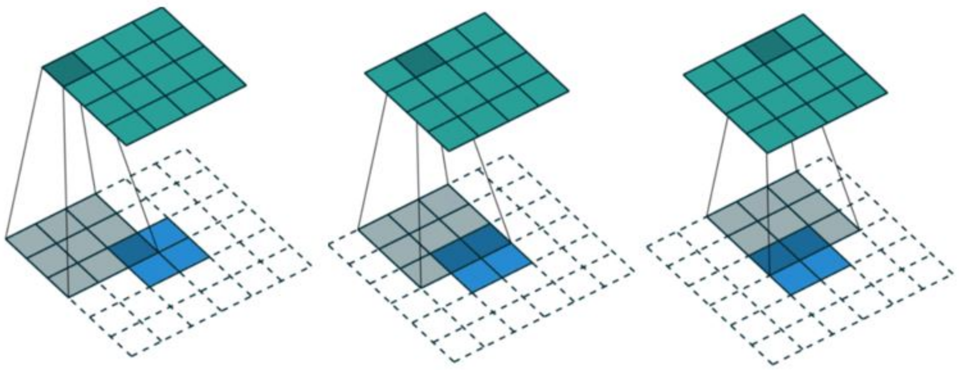


Figure 3.11: Upsampling aka Deconvolution[41]

3.5 Improving Deep Neural Networks

3.5.1 Overview

This section discusses the decisions made into improving the U-Net model. Deep Learning is a highly iterative process and training is extremely slow and can get stuck on plateaus. i.e. the model is not getting better. I've had this experience many times while training the U-Net for hours and getting bad results. To avoid this, different techniques have been applied on the model that helped tremendously in training and reaching better convergence. It's also worth mentioning that there are many ways to improve a neural network and that it is constantly evolving as new techniques are created. It is completely infeasible to explain each one. The author has decided to only introduce techniques that are effective and currently state of the art.

3.5.2 Regularization using Dropout

Neurons in a neural network have a high tendency to adapt co-dependency among each other during training time. This means that certain activations become more heavily weighted and results in a network becoming overfit to the training set. To stop the network from overfitting Dropout is used. Figure 3.12

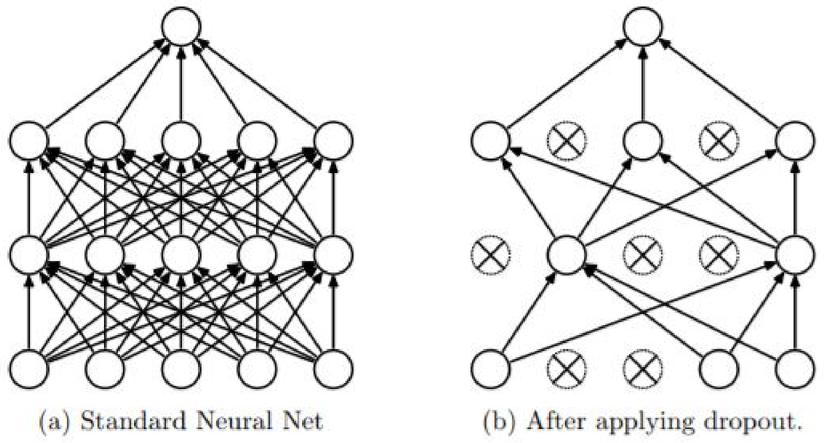
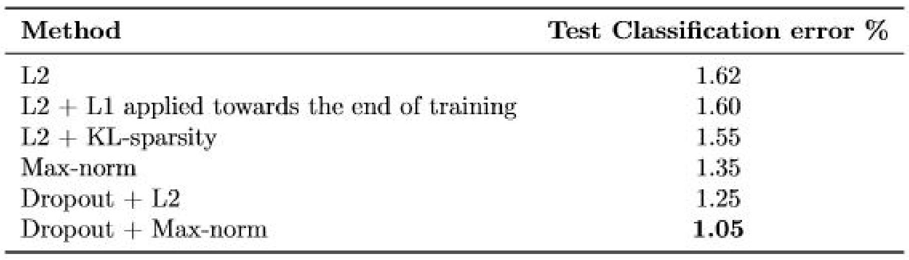


Figure 3.12: Dropout on a Neural Network [40]

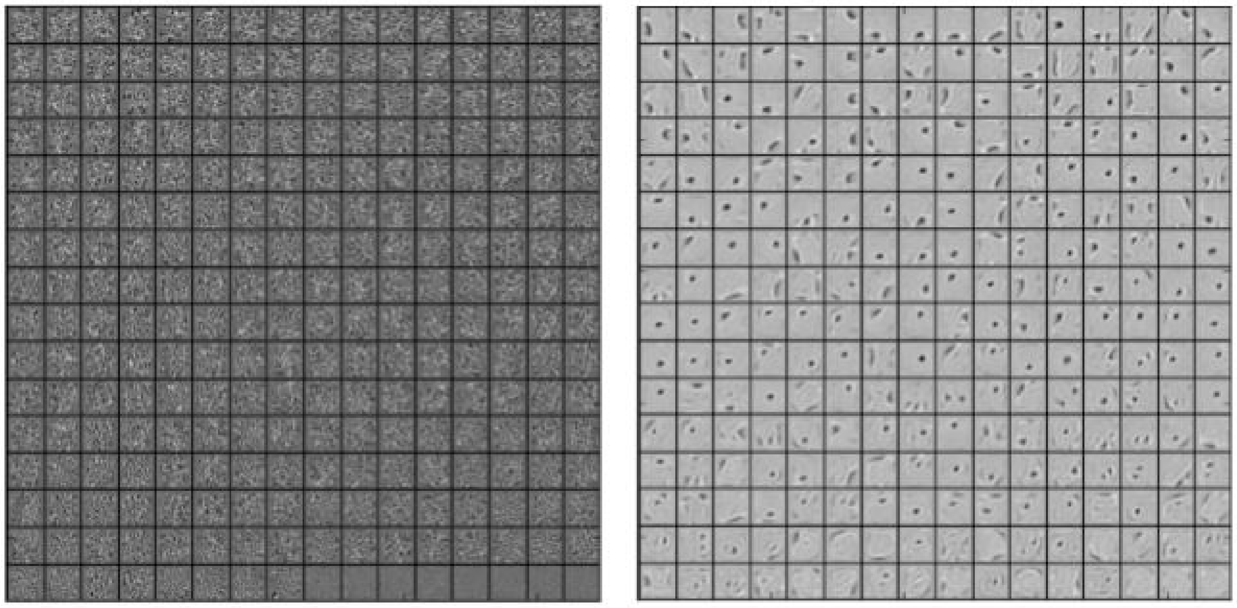
According to Hinton et al[40], Dropout is a regularization technique that reduced neuron's ability to be interdependent among one another during training time. This is achieved by shutting off random neurons in the network during time. This forces the network to learn better features that are more robust relative to random subsets of other neurons.



Method	Test Classification error %
L2	1.62
L2 + L1 applied towards the end of training	1.60
L2 + KL-sparsity	1.55
Max-norm	1.35
Dropout + L2	1.25
Dropout + Max-norm	**1.05**

Figure 3.13: Comparing Regularization Technniques [40]

Hinton et al also mentions that compared to the standard regularization techniques such as L2 with learning decay. Test classification error is greatly reduced which means that Dropout is currently state of the art (See Figure 3.13). The effects of learned features on MNIST benchmark can be seen at Figure 3.14



(a) Without dropout (b) Dropout with $p = 0.5$.

Figure 3.14: Dropout Effects on MNIST [40]

3.5.3 Weight Initializations

One of the main problems encountered in deep learning is vanishing / exploding gradients. This is when the activations become smaller or larger at each layer. To summarise briefly, if the weights in a network are too small, then the signals shrink as it passes through each layer until its too small to be useful conversely if the weights in the network start too large then the signals becomes too large to be useful.

This also has terrible consequence for the parameter updates on the weights as the network back propagates either giving very small updates or very large into the network. This has terrible consequence for large neural networks as it means the larger the network is the more likely it will be to be slow to train and inaccurate.

The two main sophisticated ways to initialize the weights is Xavier [42] and He[25]. Both of these initializations put neurons into a Gaussian or uniform distribution and draws weights from the distribution at a certain mean and variance.

After reading He et al's paper [25], the author decided to use this technique as apparently it performs better than Xavier.(See Figure 3.15)

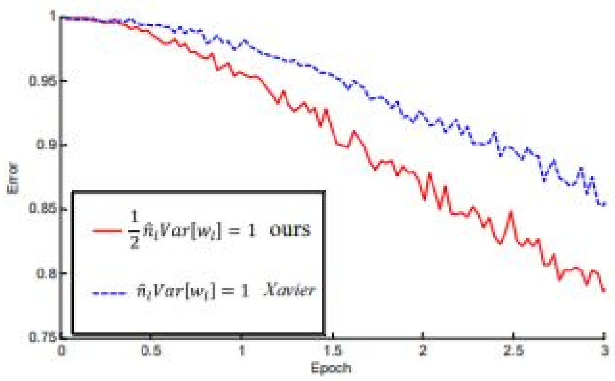


Figure 3.15: He vs Xavier Performance [25]

3.5.4 Gradient Descent

Gradient descent is the learning algorithm used in Deep Learning. It is mainly used in the back propagation step in the training of the neural network. In a neural network the cost function measures the inconsistency between a predicted value and the actual value. The idea behind gradient descent is that we increase the quality of the weights so that the weights minimize a cost function in the neural network.

Gradient descent takes the loss function and calculates a derivative. This is also known as calculating the gradient of a loss function. The idea behind calculating the gradient is that it computes the best direction which the weights must be updated to guarantee a best outcome for minimizing the cost function.

The learning rate parameter then is used to determine the size of the steps we take given the direction. The closer the network gets to convergence i.e. lowest point in the cost function, the more accurate the neural network becomes.

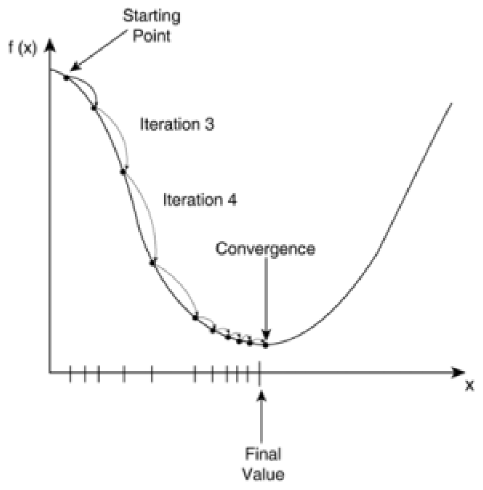


Figure 3.16: Gradient Descent Example

3.5.5 Variants of Gradient Descent

Over time there have been many variations of gradient descent that have improved the algorithm's capability. The main ones are batch gradient descent, stochastic gradient descent and mini-batch gradient descent. Sebastian Ruder wrote a great paper on gradient descent optimization algorithms[37] and according to Ruder the trade-off between the algorithms is the accuracy of the parameter update and the time it takes to perform the update.

3.5.5.1 Batch Gradient Descent

Batch gradient descent computes the gradient of a cost function relating to the parameters for the entire dataset. This basically means that batch gradient descent has to look at the entire dataset before an updating the parameters at a set number of epochs.

An epoch is a pre-defined number of times a neural network will a forward and backward pass during training. It also means predefines how many times the parameters gets updated. 1 epoch = 1 forward pass = 1 backward pass = 1 parameter update on the weights. Listing 3.1 shows the batch gradient descent algorithm.

Listing 3.1: Batch Gradient Descent Algorithm AKA Vanilla

```
for i in range(nb_epochs):
        params_grad = evaluate_gradient(loss_function, data, params)
        params = params - learning_rate * params_grad
```

Since batch gradient descent looks at an entire dataset before updating the parameters, it could mean that training could be really slow and inefficient as it recomputes gradients before updating a parameter. This could also mean that the algorithm could get stuck on a local minima and not improve per epoch. [37]

3.5.5.2 Stochastic Gradient Descent

Stochastic gradient descent on the other hand performs a parameter update on each training instance(x) and its corresponding label(y). Listing 3.2 shows the SGD algorithm.

Listing 3.2: Stochastic Gradient Descent

```
for i in range(nb_epochs):
    np.random.shuffle(data)
    for example in data:
        params_grad = evaluate_gradient(loss_function , example, params)
        params = params - learning_Rate * params_grad
```

SGD removes the redundancy problem that batch gradient descent performs which means it could be faster during training. It also means that stochastic gradient descent could jump out of a local minima during training and go towards a better local minima.
The downside to this is that, it could complicate convergence on a local minima as the direction fluctuates per training example.

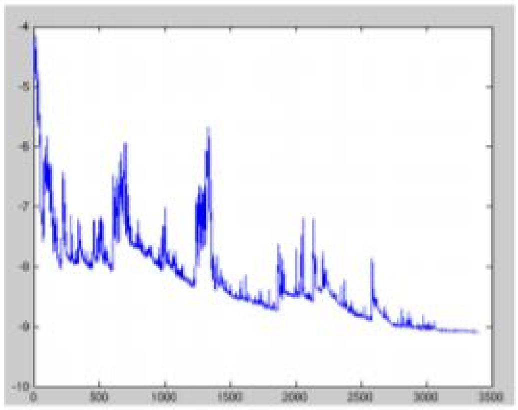


Figure 3.17: Stochastic Gradient Descent Training Fluctuations

3.5.5.3 Mini-Batch Gradient Descent

Mini-batch gradient descent takes both elements of batch gradient descend and stochastic gradient descent. Mini batch gradient descent splits training data into n mini-batches and performs an update for each one. Listing 3.3 shows the mini batch gradient descent algorithm.

Listing 3.3: Mini-Batch Gradient Descent Algorithm

```
for i in range(nb_epochs):
    np.random.shuffle(data)
    for batch in get_batches(data,batch_size=50)
        params_grad = evaluate_gradient(loss_function , batch,params)
        params = params - learning_rate * params_grad
```

Using mini batches means that there would be no redundancy in computing gradients and at the same time less variance in the gradients themselves which leads to faster training and less fluctuations of the cost function during training time.

Mini-batch gradient descent is usually the method of choice for training neural networks.

3.5.5.4 Conclusion

These learning algorithms help in training the neural networks however on their own they cannot guarantee good convergence. In the next section I introduce Optimization algorithms that help gradient descent learning algorithms perform better.

3.5.6 Gradient Descent Optimization Algorithms

3.5.6.1 Overview

Although there are different gradient descent algorithms out there with their own advantages and disadvantages, there are still some issues regarding training the network that has to be solved in a different way. In this section I introduce some ways we can optimize our gradient descent algorithm.

According to Ruder, choosing a learning rate can be difficult as statically predefining the learning rate can either mean slow convergence when its too small or fluctuate around the minimum when the learning rate is too large. To solve this problem there needs to be a dynamic way to choose the learning rate.

Choosing the same learning rate also means that an update applies to each parameter, this can be a problem if we don't want to update the weights that find less frequent features in our data. This means that its not a good idea to update weights that find rare occurring features compared to commonly occurring ones.

The last challenge is minimizing non-convex error functions aka saddle points. Bengio et al[22], mentions that minimizing over saddle points give great difficulty for a neural network to find the global minimum which means learning can be extremely slow and give an impression that a network is stuck on a local minimum. The reason for this is because a saddle point's gradient is extremely small which makes it difficult for the gradient descent algorithm to minimize the cost function. Figure 3.18 shows gradient descent traversing through a saddle point plateau.

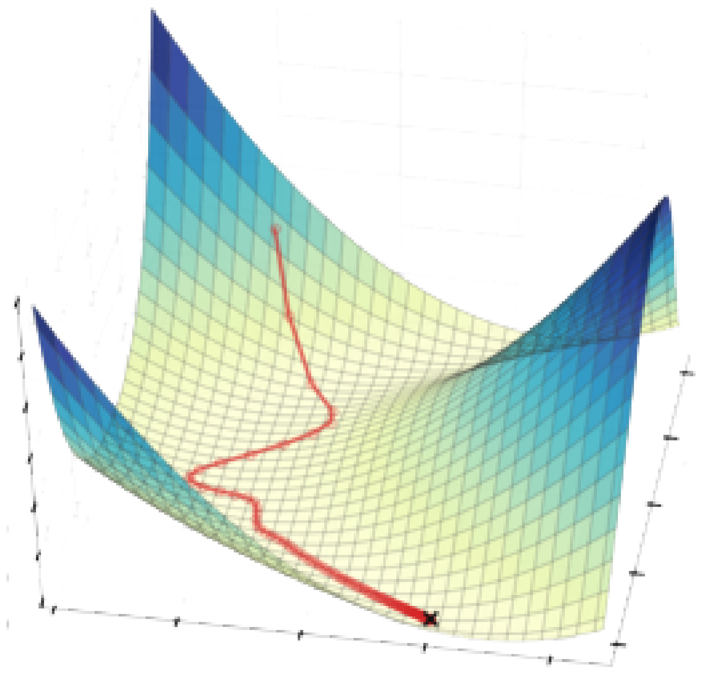


Figure 3.18: Saddle Point

3.5.6.2 Momentum

Due to Stochastic Gradient nature to oscillate between different directions, progressing towards a local / global minimum can be slow. To minimise these oscillations, Momentum is used. Momentum reduces the oscillations by updating the current step to made by adding a fraction of the past step.

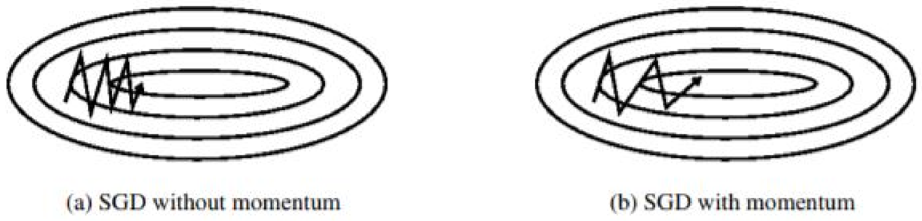


Figure 3.19: Momentum reduced oscillations during training

According to Ruder[37], the momentum term increases for dimensions whose gradients point in the same direction and reduces updates for dimensions whose gradients change directions. This means that a neural network can converge faster by reducing oscillations. The default value for momentum is 0.9.

3.5.6.3 RMSprop

RMSProp is an adaptive learning rate technique proposed by Geoffrey Hinton. Ruder mentions that RMSprop divides the learning rate by exponentially decaying average of squared gradients. This means that the learning rate dynamically decays exponentially as the network trains. This means that the network can converge much better towards the global minimum. A good default value for learning rate for RMSprop is 0.001.

3.5.6.4 Adaptive Moment Estimation (ADAM)

Adam is another method that computes adaptive learning rates. This method was invented by Diederik Kingma and Jimmy Ba[1].
Adam is similar to RMSprop and Momentum where it keeps an exponentially decaying average of the squared gradients and exponentially decaying average gradients. Since these two values are 0 when initialized, Adam uses bias correction for the first and second moment estimation to correct the algorithms initial bias towards 0. These two values are then used to update the parameters. According to Kingma et al[1], Adam when used with Dropout, outperforms the other optimization algorithms. See Figure 3.20

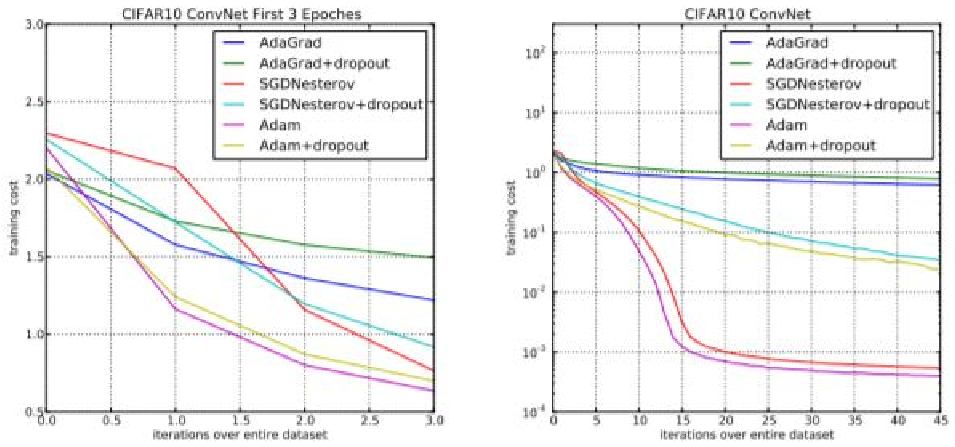


Figure 3.20: Adam Performance on CIFAR-10 Benchmark

3.6 Conclusion

With these intuitions in mind, One would be able to get a better idea of what is going on inside a deep learning model while it's training. There is an element of black-box randomness that happens when training deep learning models but these concepts gives me a better idea of what could be happening when a network doesn't work as intended. In short, it helps to know these concepts when debugging.
In the next chapter the author, outlines the designs for the project.

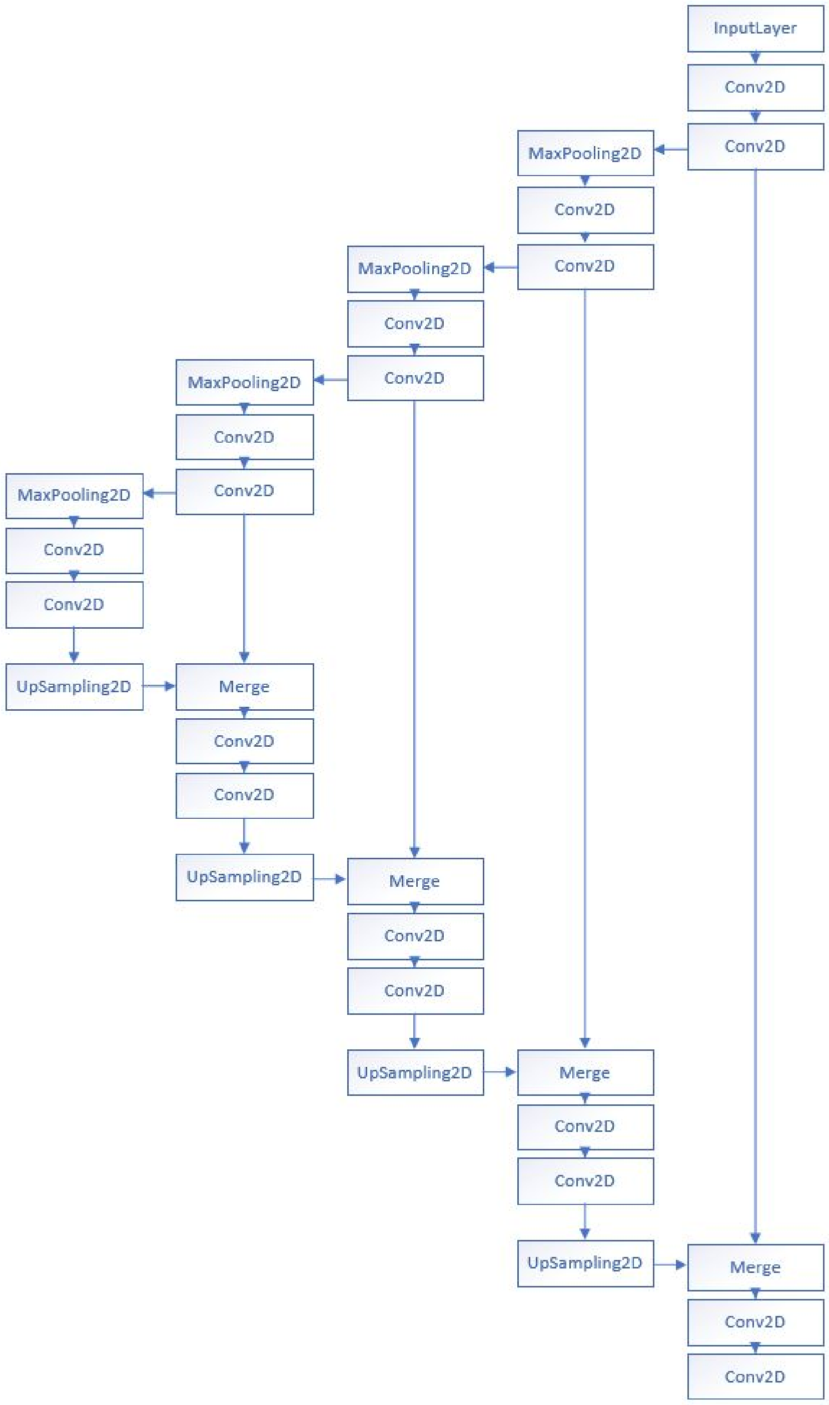
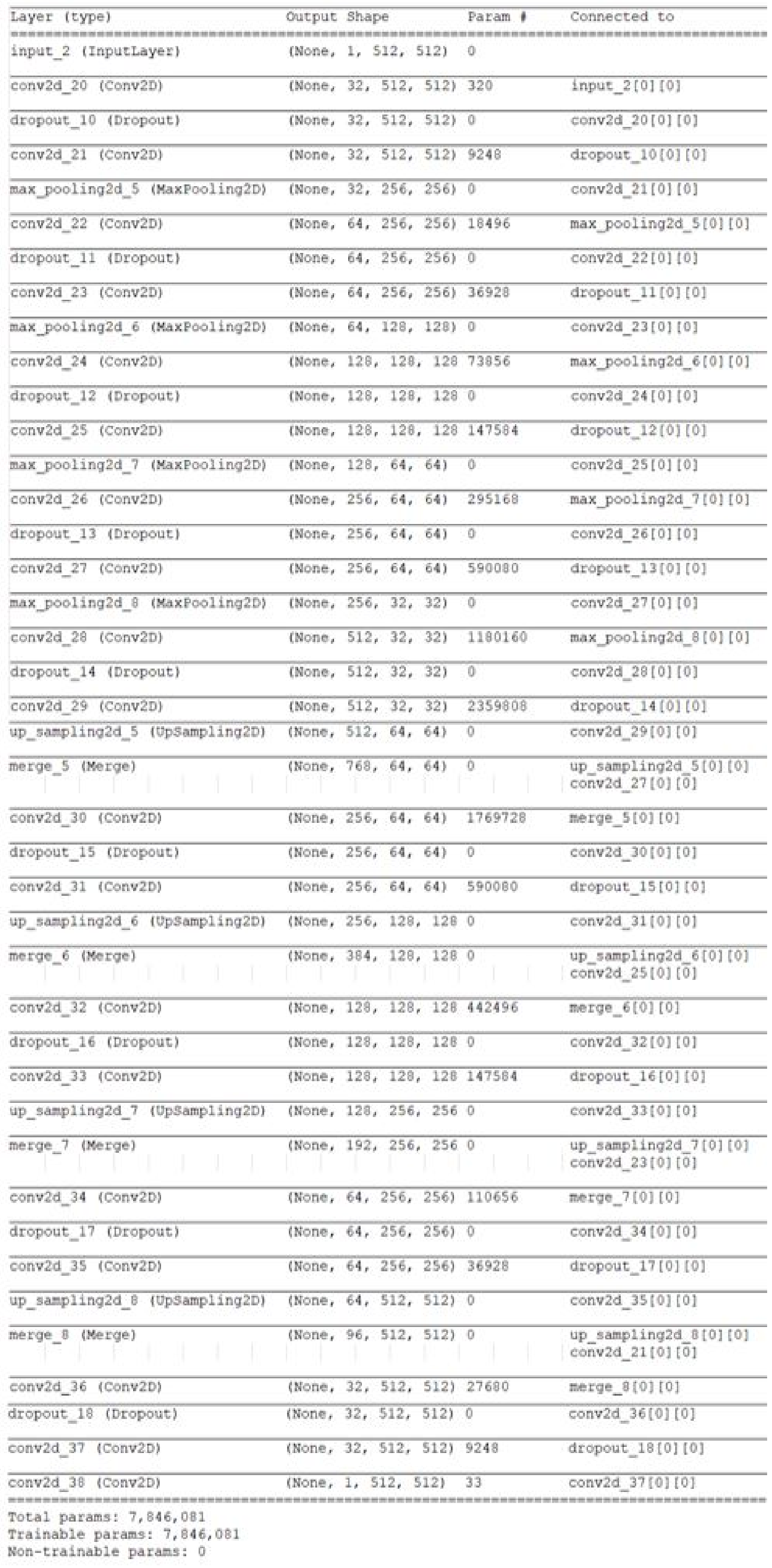


Figure 3.21: U-Net Architecture



```
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_2 (InputLayer)             (None, 1, 512, 512)   0
____________________________________________________________________________________________________
conv2d_20 (Conv2D)               (None, 32, 512, 512)  320         input_2[0][0]
____________________________________________________________________________________________________
dropout_10 (Dropout)             (None, 32, 512, 512)  0           conv2d_20[0][0]
____________________________________________________________________________________________________
conv2d_21 (Conv2D)               (None, 32, 512, 512)  9248        dropout_10[0][0]
____________________________________________________________________________________________________
max_pooling2d_5 (MaxPooling2D)   (None, 32, 256, 256)  0           conv2d_21[0][0]
____________________________________________________________________________________________________
conv2d_22 (Conv2D)               (None, 64, 256, 256)  18496       max_pooling2d_5[0][0]
____________________________________________________________________________________________________
dropout_11 (Dropout)             (None, 64, 256, 256)  0           conv2d_22[0][0]
____________________________________________________________________________________________________
conv2d_23 (Conv2D)               (None, 64, 256, 256)  36928       dropout_11[0][0]
____________________________________________________________________________________________________
max_pooling2d_6 (MaxPooling2D)   (None, 64, 128, 128)  0           conv2d_23[0][0]
____________________________________________________________________________________________________
conv2d_24 (Conv2D)               (None, 128, 128, 128  73856       max_pooling2d_6[0][0]
____________________________________________________________________________________________________
dropout_12 (Dropout)             (None, 128, 128, 128  0           conv2d_24[0][0]
____________________________________________________________________________________________________
conv2d_25 (Conv2D)               (None, 128, 128, 128  147584      dropout_12[0][0]
____________________________________________________________________________________________________
max_pooling2d_7 (MaxPooling2D)   (None, 128, 64, 64)   0           conv2d_25[0][0]
____________________________________________________________________________________________________
conv2d_26 (Conv2D)               (None, 256, 64, 64)   295168      max_pooling2d_7[0][0]
____________________________________________________________________________________________________
dropout_13 (Dropout)             (None, 256, 64, 64)   0           conv2d_26[0][0]
____________________________________________________________________________________________________
conv2d_27 (Conv2D)               (None, 256, 64, 64)   590080      dropout_13[0][0]
____________________________________________________________________________________________________
max_pooling2d_8 (MaxPooling2D)   (None, 256, 32, 32)   0           conv2d_27[0][0]
____________________________________________________________________________________________________
conv2d_28 (Conv2D)               (None, 512, 32, 32)   1180160     max_pooling2d_8[0][0]
____________________________________________________________________________________________________
dropout_14 (Dropout)             (None, 512, 32, 32)   0           conv2d_28[0][0]
____________________________________________________________________________________________________
conv2d_29 (Conv2D)               (None, 512, 32, 32)   2359808     dropout_14[0][0]
____________________________________________________________________________________________________
up_sampling2d_5 (UpSampling2D)   (None, 512, 64, 64)   0           conv2d_29[0][0]
____________________________________________________________________________________________________
merge_5 (Merge)                  (None, 768, 64, 64)   0           up_sampling2d_5[0][0]
                                                                   conv2d_27[0][0]
____________________________________________________________________________________________________
conv2d_30 (Conv2D)               (None, 256, 64, 64)   1769728     merge_5[0][0]
____________________________________________________________________________________________________
dropout_15 (Dropout)             (None, 256, 64, 64)   0           conv2d_30[0][0]
____________________________________________________________________________________________________
conv2d_31 (Conv2D)               (None, 256, 64, 64)   590080      dropout_15[0][0]
____________________________________________________________________________________________________
up_sampling2d_6 (UpSampling2D)   (None, 256, 128, 128  0           conv2d_31[0][0]
____________________________________________________________________________________________________
merge_6 (Merge)                  (None, 384, 128, 128  0           up_sampling2d_6[0][0]
                                                                   conv2d_25[0][0]
____________________________________________________________________________________________________
conv2d_32 (Conv2D)               (None, 128, 128, 128  442496      merge_6[0][0]
____________________________________________________________________________________________________
dropout_16 (Dropout)             (None, 128, 128, 128  0           conv2d_32[0][0]
____________________________________________________________________________________________________
conv2d_33 (Conv2D)               (None, 128, 128, 128  147584      dropout_16[0][0]
____________________________________________________________________________________________________
up_sampling2d_7 (UpSampling2D)   (None, 128, 256, 256  0           conv2d_33[0][0]
____________________________________________________________________________________________________
merge_7 (Merge)                  (None, 192, 256, 256  0           up_sampling2d_7[0][0]
                                                                   conv2d_23[0][0]
____________________________________________________________________________________________________
conv2d_34 (Conv2D)               (None, 64, 256, 256)  110656      merge_7[0][0]
____________________________________________________________________________________________________
dropout_17 (Dropout)             (None, 64, 256, 256)  0           conv2d_34[0][0]
____________________________________________________________________________________________________
conv2d_35 (Conv2D)               (None, 64, 256, 256)  36928       dropout_17[0][0]
____________________________________________________________________________________________________
up_sampling2d_8 (UpSampling2D)   (None, 64, 512, 512)  0           conv2d_35[0][0]
____________________________________________________________________________________________________
merge_8 (Merge)                  (None, 96, 512, 512)  0           up_sampling2d_8[0][0]
                                                                   conv2d_21[0][0]
____________________________________________________________________________________________________
conv2d_36 (Conv2D)               (None, 32, 512, 512)  27680       merge_8[0][0]
____________________________________________________________________________________________________
dropout_18 (Dropout)             (None, 32, 512, 512)  0           conv2d_36[0][0]
____________________________________________________________________________________________________
conv2d_37 (Conv2D)               (None, 32, 512, 512)  9248        dropout_18[0][0]
____________________________________________________________________________________________________
conv2d_38 (Conv2D)               (None, 1, 512, 512)   33          conv2d_37[0][0]
====================================================================================================
Total params: 7,846,081
Trainable params: 7,846,081
Non-trainable params: 0
```

Figure 3.22: U-Net Architecture Layers and Parameters

Chapter 4

Approach and Methodology

4.1 Overview

The two chosen methodologies to develop this project are: Agile Methodology and CRISP-DM. These methodologies were chosen to help achieve data mining and software development goals.

Agile allows for rapid prototyping and being user focused. In this project Agile has allowed the author to appropriately plan the items in order of priority to ensure that main project goals are achieved.

CRISP-DM is a methodology for the development of the deep learning model, as it the provides a greater understanding on how to deliver this model through the steps outlined below.

The next sections will give a brief overview into each methodology and how it is being utilised into this project.

4.2 Agile Methodology

According to the Agile Manifesto[10], Agile values 4 things which are:

- **Individuals and interactions** over processes and tools.
- **Working software** over comprehensive documentation.
- **Customer Collaboration** over contract negotiation.
- **Responding to change** over following a plan.

Agile Methodology puts the focus back on the people rather than on documentation. In agile, tasks are broken down into 'user stories' where the team plans the tasks ahead and estimates how much time they will take.

A story could be coding such as code refactor, new feature developments or non coding such as research or design. These stories are then grouped into sprints which are two weeks in duration. The team gets together and agrees on which tasks will be delivered in the sprints. The team can then either conduct a daily scrum meeting or another way like Kanban to discuss briefly about "what they did yesterday","what they are going to do today" and "Any blockers?" to gain a better idea about what each one is doing.

In each sprint, there can also be a Retrospective which is when the team gets together and reviews the sprint(s) and discuss topics like "What went well?", "What went badly?" and "How can we improve?". This way there is a constant effort in becoming better.

4.3 CRISP-DM - CRoss-Industry Standard Process for Data Mining

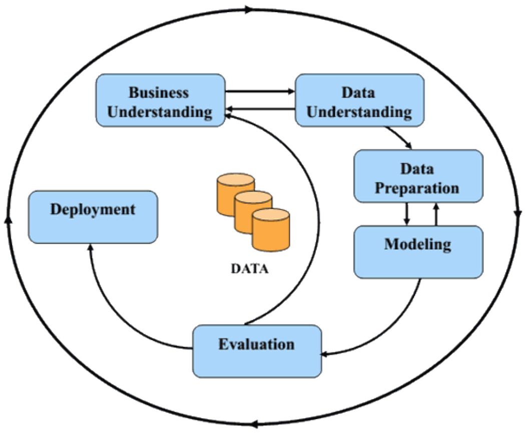
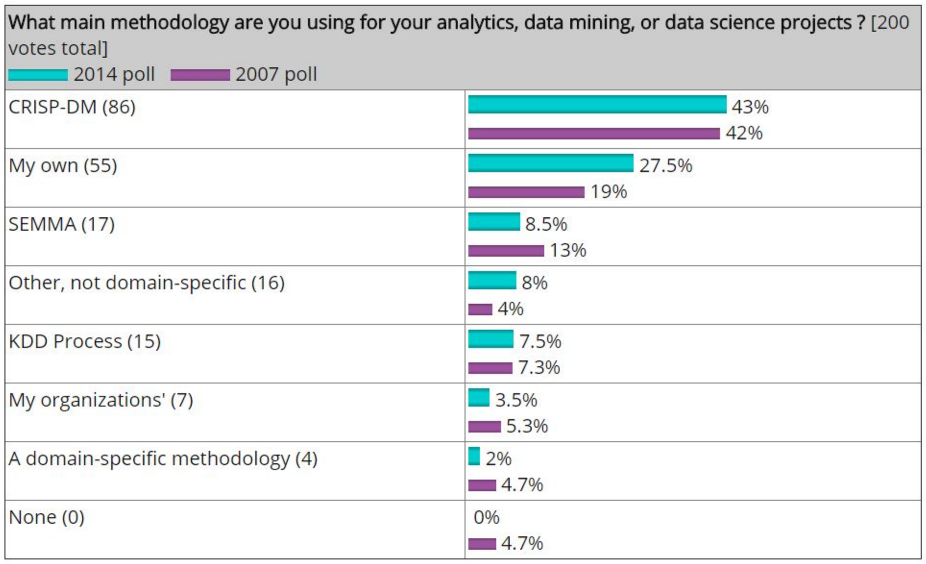


Figure 4.1: CRISP-DM Methodology

According to Shearer(2000)[39]. CRISP-DM is a non proprietary, documented and freely available data mining model. It offers an industry tool and application neutral model that encourages best practices and offers organizations the structure needed to realise better and faster results from data mining. Figure 4.1 displays the 6 phases of CRISP-DM.According to Shapiro,[19], CRISP-DM is the leading methodology for analytics, data mining or data science projects. See Figure 4.2.



What main methodology are you using for your analytics, data mining, or data science projects ? [200 votes total]

🟦 2014 poll 🟪 2007 poll

Methodology	2014 poll	2007 poll
CRISP-DM (86)	43%	42%
My own (55)	27.5%	19%
SEMMA (17)	8.5%	13%
Other, not domain-specific (16)	8%	4%
KDD Process (15)	7.5%	7.3%
My organizations' (7)	3.5%	5.3%
A domain-specific methodology (4)	2%	4.7%
None (0)	0%	4.7%

Figure 4.2: Methodology Poll Results from KD Nuggets[19]

4.3.1 Business Understanding

The most important phase, this is all about using the objective of the data mining problem to suit a business perspective. This involves understanding the problem from a business domain, assessing the business success criteria and assessing the business environment from the perspective of resources, requirements, risks, costs and benefits. For this project this means determining what how a lung cancer detection system will benefit business. This requires a project plan where the data mining goals are aligned with business objectives.

4.3.2 Data Understanding

This phase is about collecting the data, gaining familiarity and ultimately understanding the strengths, opportunities and weakness of the data as data does not always fit the problem that is being solved. The analyst must be aware of the structure of the data and be able to describe it and ultimately verify the quality of the data. Some common questions an analyst may ask himself/herself is ”Is the data complete?, Is there any missing values, Have I explored the data enough?”.

4.3.3 Data Preparation

Data Preparation phase covers the activities regarding the dataset that has been collected. This step is very important before modelling and the steps in this phase are: selecting data that is the most relevant to the data mining goal, cleansing the data to make sure the quality of the data is correct, constructing the data by transforming the data into a new way that fits the model, integrating the data by combining different records to create new one and lastly formatting the data, this could be by converting the data into the correct data structure.

4.3.4 Modelling

The modelling phase is all about selecting a model for your problem for example, a linear regression or classification model, building the model, applying and calibrating different optimal values and assessing the design of the model by calculating an error rate for example to test the validity of the model.

4.3.5 Evaluation

Before deploying the model, it must be evaluated, this phase refers to checking the model to see if it solves the business case. In this phase it is critical to check if the model succeeds in the business success criteria, if the project meets the time and budget constraint and whether the project meets its original business objectives.

4.3.6 Deployment

The deployment phase contains many different actions, it is usually not a data mining task. For deployment, it could mean multiple things like applying the model live for the customer, planning the deployment and monitoring, produce a final report or it could also be reviewing the project again. This could also depend on the business objectives.

4.4 Conclusion

The two chosen methodologies allow the tasks of the project to be broken down and categorised into relevant phases. Agile allows for breaking down tasks into deliverable and manageable tasks while CRISP-DM allows for a greater understanding of the data mining workflow.
In the next chapter, the design artefacts for the project will be detailed.

Chapter 5

Design

5.1 Overview

This chapter deals with the design aspect of the project. Although the project weighs heavily more on the data mining. The author wanted to ensure that the design of application would consider the potential users. In this chapter there will be artefacts regarding user analysis and technical design of the application. These design artefacts were products of conversations with medical experts and research online.

5.2 User Analysis: Persona

5.2.1 Background

Dr. Jim Bob is a 35 year old radiologist at St. Vincent's Hospital. Jim works in a multidisciplinary team of doctor's in the hospital to diagnose lung cancer for patients. Jim obtained his degree in Royal College of Surgeons in Ireland and has been working in lung cancer detection for over 10 years in Dublin Ireland. Jim is responsible for performing CT scans on patients who may have lung cancer and constantly analyses CT scan slices.

Figure 5.1: Persona: Dr. Jim Bob(Image Source:healthcaresalaryworld.com)

5.2.2 Goals, Motivations and Frustrations

Jim's goals is to solve lung cancer, he understands that lung cancer is extremely complex in nature so learning and research is important in his profession.

Jim is motivated by people, he loves working with his team and ensures the very best for his patients. He wants to constantly improve his medical skills and gets excited about ways to improve his skills as a radiologist.

Jim however is frustrated by some aspects of diagnosing lung cancer. The majority of people who get diagnosed with lung cancer are at the most advanced stage which means prognosis is not good. Jim also encounters that a large cohort gets diagnosed with very small spots in their lungs which could be benign or malignant. He cannot recommend this patient to get a biopsy as it is an invasive procedure so he recommends that these cohort of patients get rescanned in 6-12 weeks to look for signs of malignant growth.

Jim analyses hundreds of CT scans every day, this is a necessary but repetitive task for Jim. Having an automated system that filters out irrelevant scans may improve his productivity and reduce observer fatigue.

5.3 User Analysis: Scenarios

5.3.1 Scenario 1: Viewing Patient CT Scans

Jim has just scanned a patient, Jim uses his computer and uploads a CT scan on the website and is shown a gallery of scans that have been made. Jim can see all of the image slices. However Jim would like a better view of the the CT Scans as the gallery view is not large enough.

Jim then clicks on the carousel view which allows him to better view the image slices.

5.3.2 Scenario 2: Positive Prediction

Jim had just encountered a tumour on the lungs in one of his patients. Jim is not sure if it's cancer or not. Jim clicks on the image he is not sure about and uses the deep learning model to predict. The result of the prediction comes up malignant so Jim recommends that this patient take a biopsy to confirm.

This is critical, as early detection of lung cancer means that treatment can start as soon as possible which means better prognosis.

5.3.3 Scenario 3: Negative Prediction

Jim encounters another tumour and he is not sure if it's benign or malignant. Jim uses the web application again to confirm and the model responds with negative. i.e. not cancer. Jim proceeds with standard practice to rescan the patient to confirm.

5.4 User Analysis: Use Cases

5.4.1 Use Case Overview

Figure 5.2 Shows an overview of the lung cancer detection system. Users will be able to upload a CT Scan, view the detection results and view cancer diagnostics, at this current moment I am not yet sure if i will be able to deliver the cancer diagnostics part of my project so it is optional.

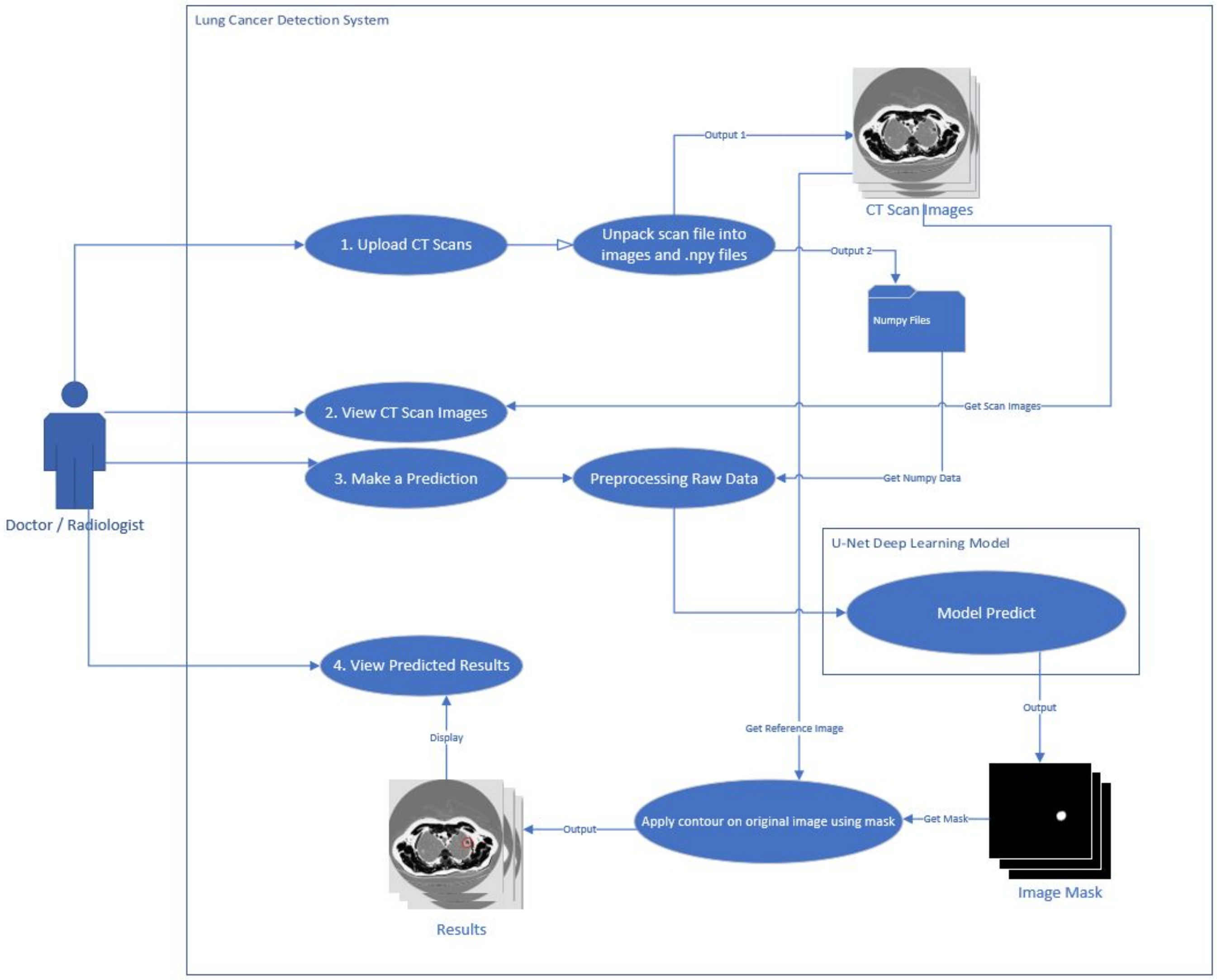


Figure 5.2: Use Case Diagram of the System

5.4.2 Use Case 1: Upload CT Scans

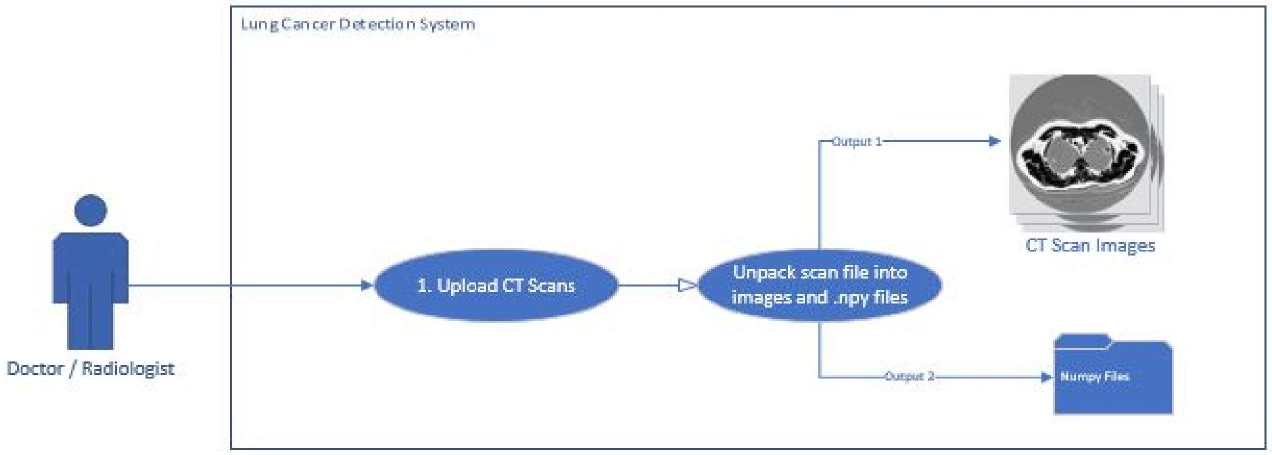


Figure 5.3: Use Case 1: Upload CT Scan

Figure 5.3 shows the first use case. The user uploads a metadata file (.mhd) and a raw file(.raw) to the back end system via POST request. The system takes the metadata and uses it to unpack the raw files which contain the images. The system then takes the image data and saves it into image files (.png) and image data array (.npy) using OpenCV and Numpy.

The reason for this is because the images generated by OpenCV is used to show to the users in the gallery. Saving the images into (.png) alters the image arrays so the model does not react well to the changes in the data. Instead we use create Numpy files for each image to give to the model and use the filenames to reference the image and the numpy files.

5.4.3 Use Case 2: View CT Scans

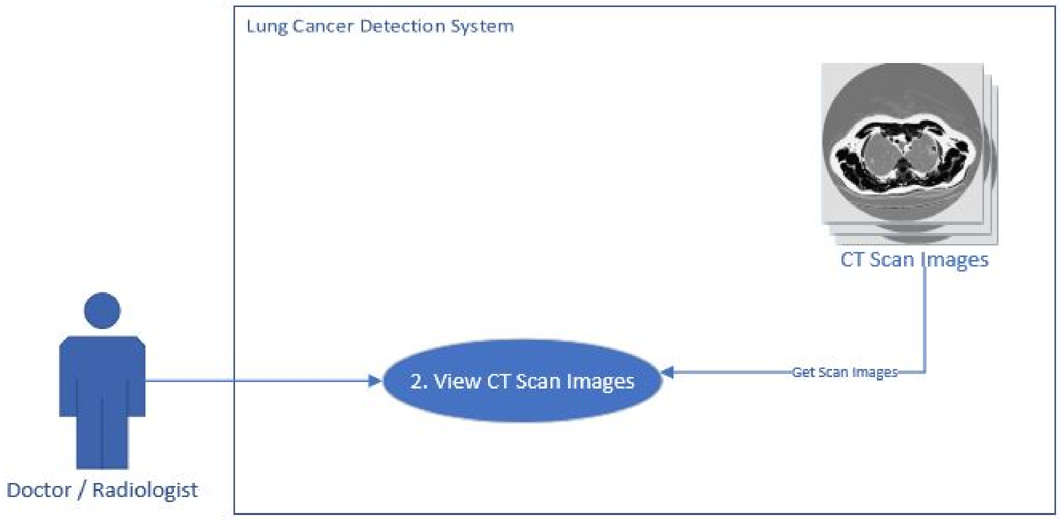


Figure 5.4: Use Case 2: View Detection Results

Figure 5.4 shows the second use case. The system pulls the image files from the back end and displays it in the front end. The system does this by sending GET requests for each image. The images on the front end can be displayed via a carousel image or a gallery style.

5.4.4 Use Case 3: Make Predictions

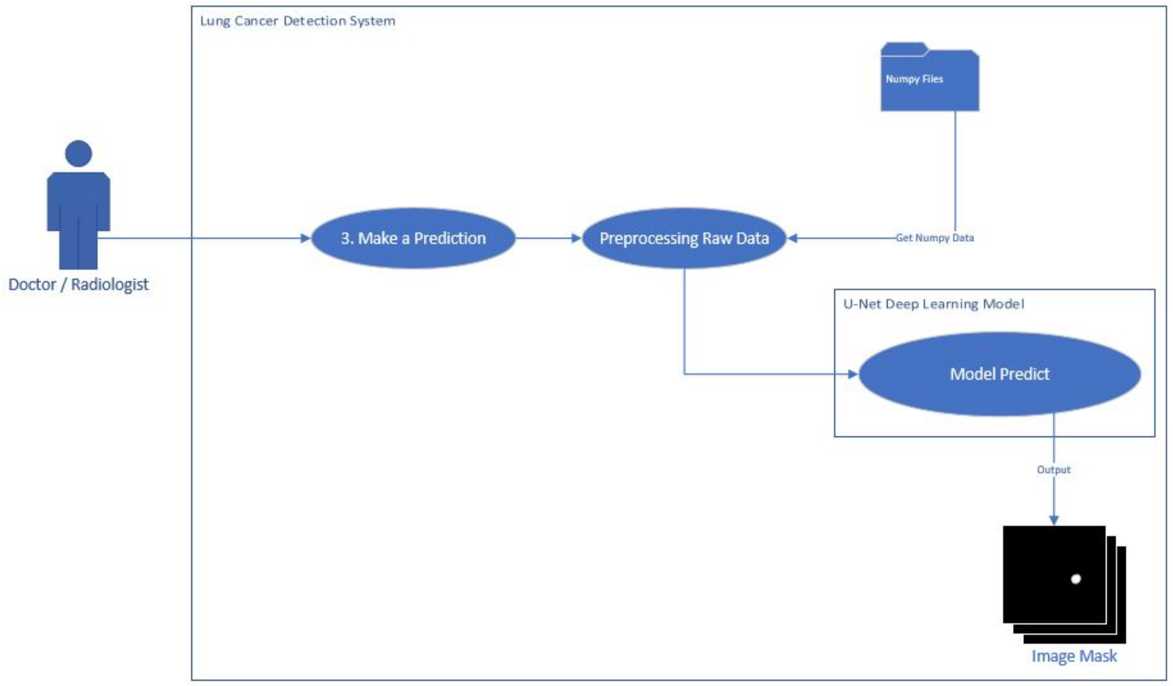


Figure 5.5: Use Case 3: Make Predictions

Figure 5.5 shows the third use case. When a user selects and image that he or she wants to make predictions. The system takes the filenames from the user during selection and uses this filename to reference a numpy file. This numpy file is then preprocessed before its fed to the deep learning model. The model then outputs an image mask as seen on the diagram.

5.4.5 Use Case 4: View Predictions

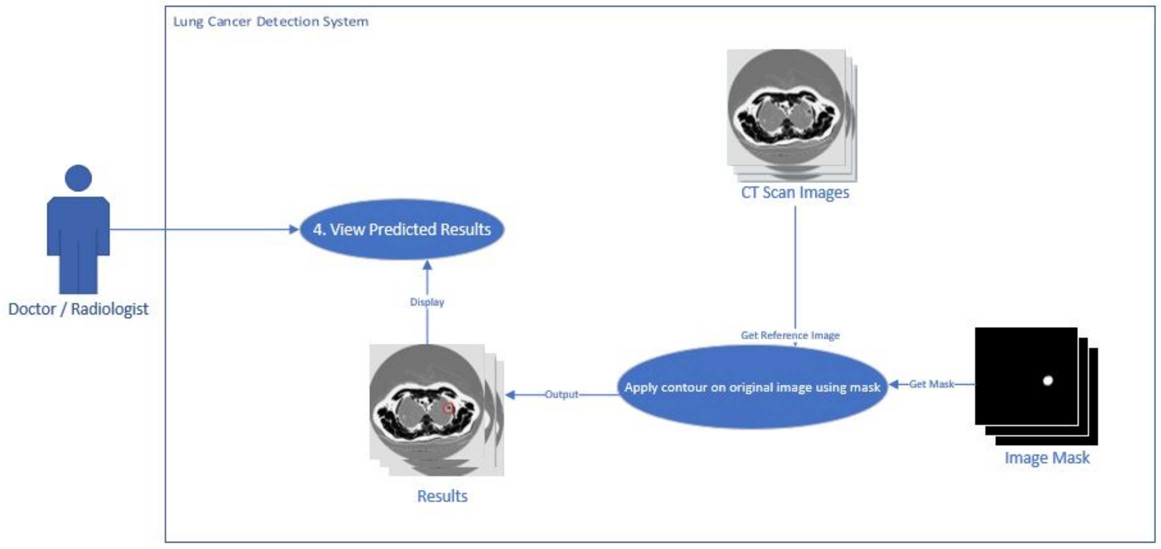

Lung Cancer Detection System
CT Scan Images
Doctor / Radiologist
4. View Predicted Results
Display
Get Reference Image
Apply contour on original image using mask
Get Mask
Output
Results
Image Mask

Figure 5.6: Use Case 4: View Predictions

Figure 5.6 shows the last use case. The system takes the original CT scan reference image and the associated mask and applies an image contour on the original image. section A.5 and section A.6 explains how masks and contours work.

5.5 Design Wireframes

5.5.1 Overview

This section outlines the wireframes designed for the view of the application. These wireframes are used to outline the benefit of the user and to get a general idea of how the application should look and feel prior to development.

5.5.2 Homepage

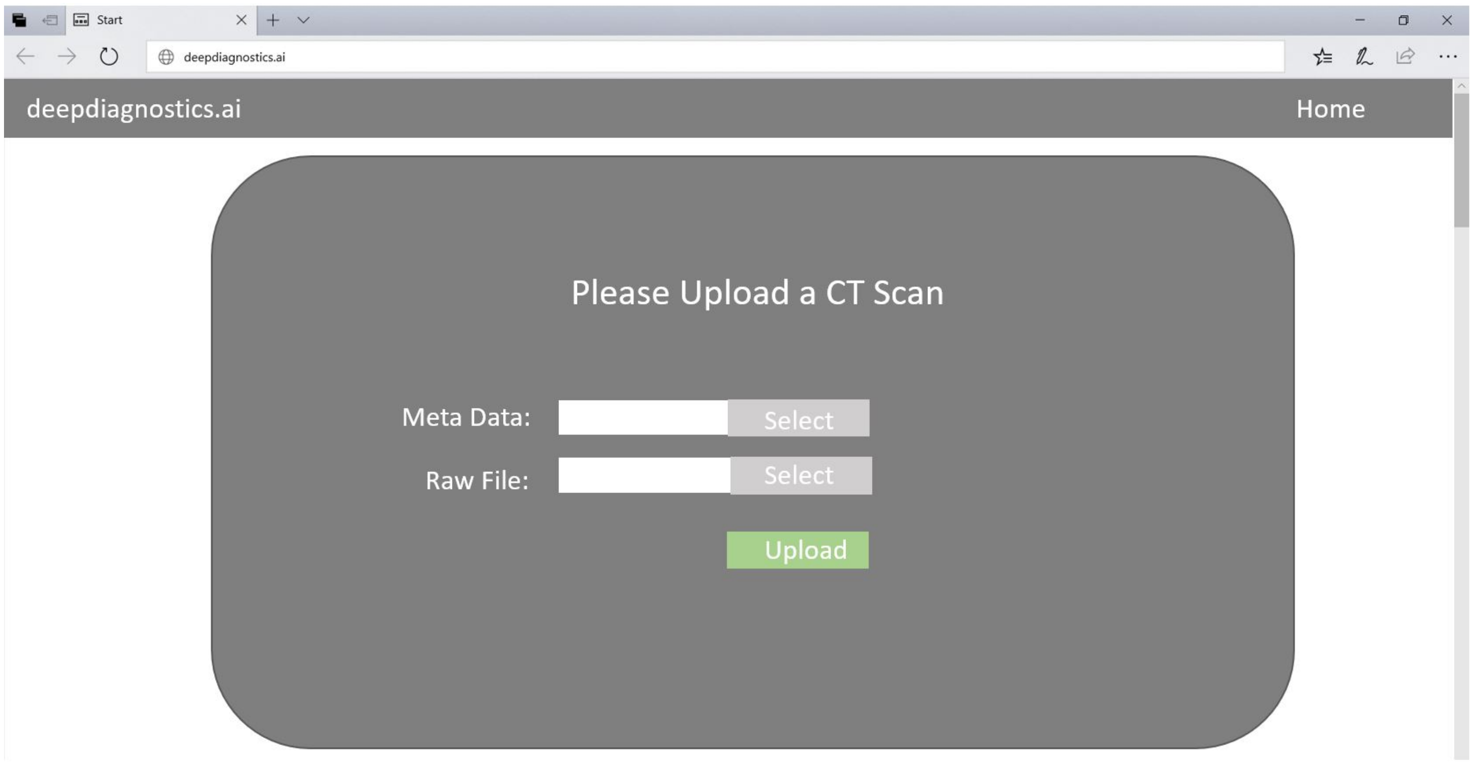
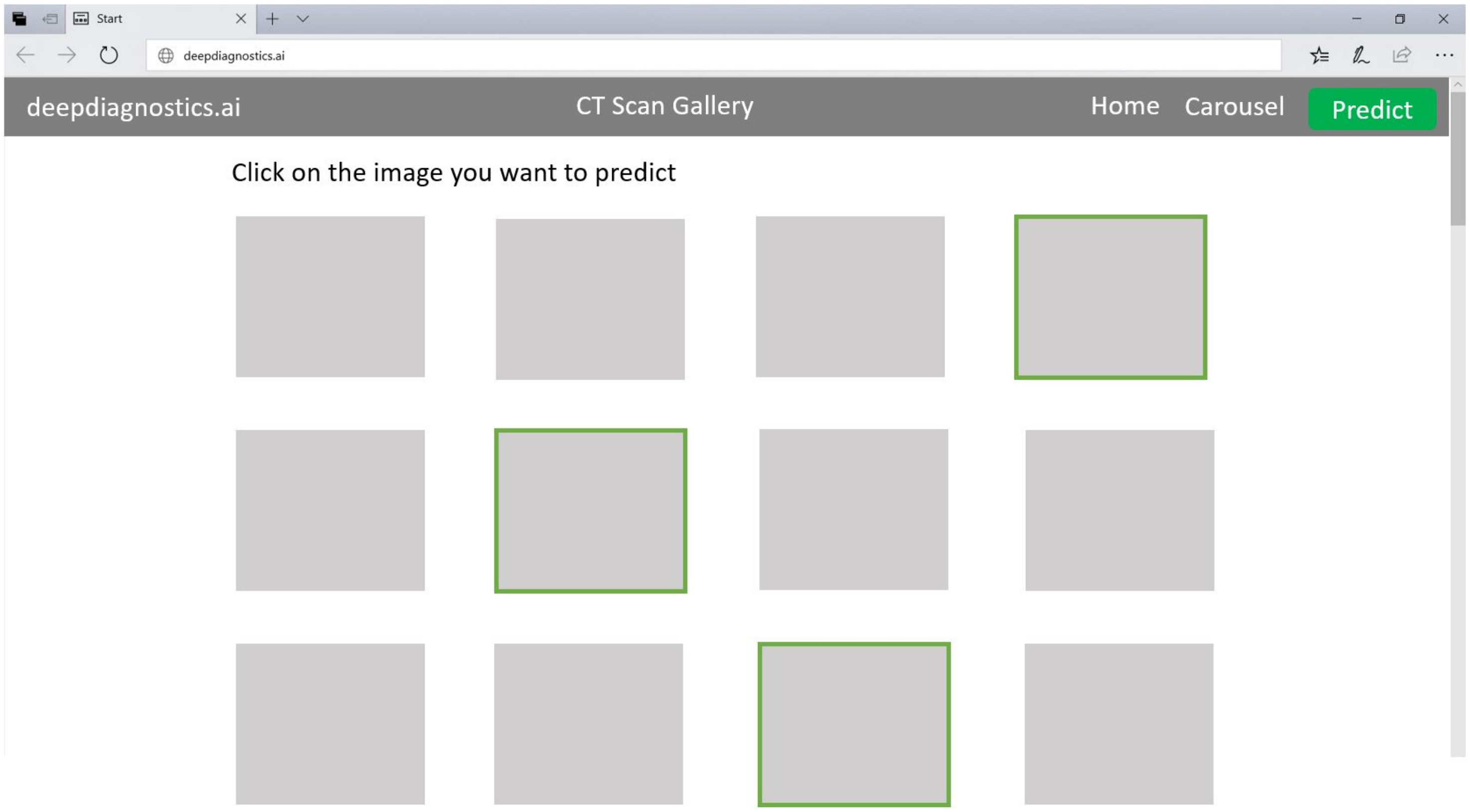


Figure 5.7: Homepage Design

Figure 5.7 shows the wireframe for the first page of the application. The only feature of the homepage is to allow the user to upload a CT Scan in his computer. The interface will ask the user to upload a metadata file and a raw file. The system will capture this data via POST request to the back end server.

5.5.3 CT Scan Gallery



Click on the image you want to predict

Figure 5.8: CT Scan Gallery

Figure 5.8 shows the second wireframe for the CT scan gallery of the application. This wireframe is what's shown when the user uploads the CT scan and the system finishes unpacking the raw file that the user has uploaded. The main feature of this wireframe is that a user will be able to select a scan that he wants to predict and the system will then take those images and feed it to the deep learning model.

5.5.4 CT Scan Carousel

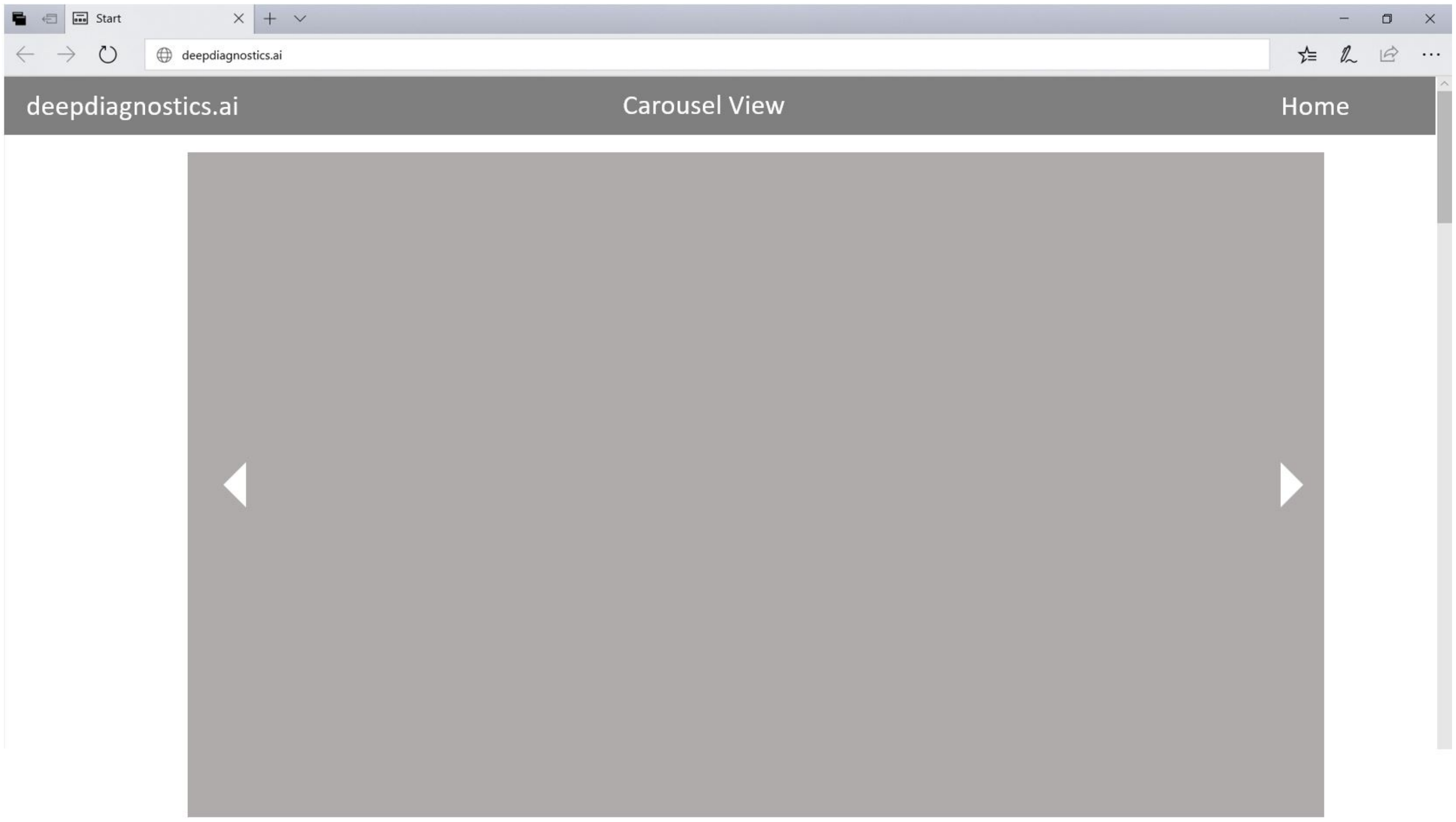


Figure 5.9: CT Scan Carousel

Figure 5.9 shows the carousel wireframe of the application. This is for when the user needs a closer lookup of the CT scan. The blank box in the middle is a container for a sample CT Scan image. The main feature here is that the user will be able to see a sequential view of the images by clicking on left and right of the image or left or right on the keyboard.

5.5.5 Prediction Output Gallery

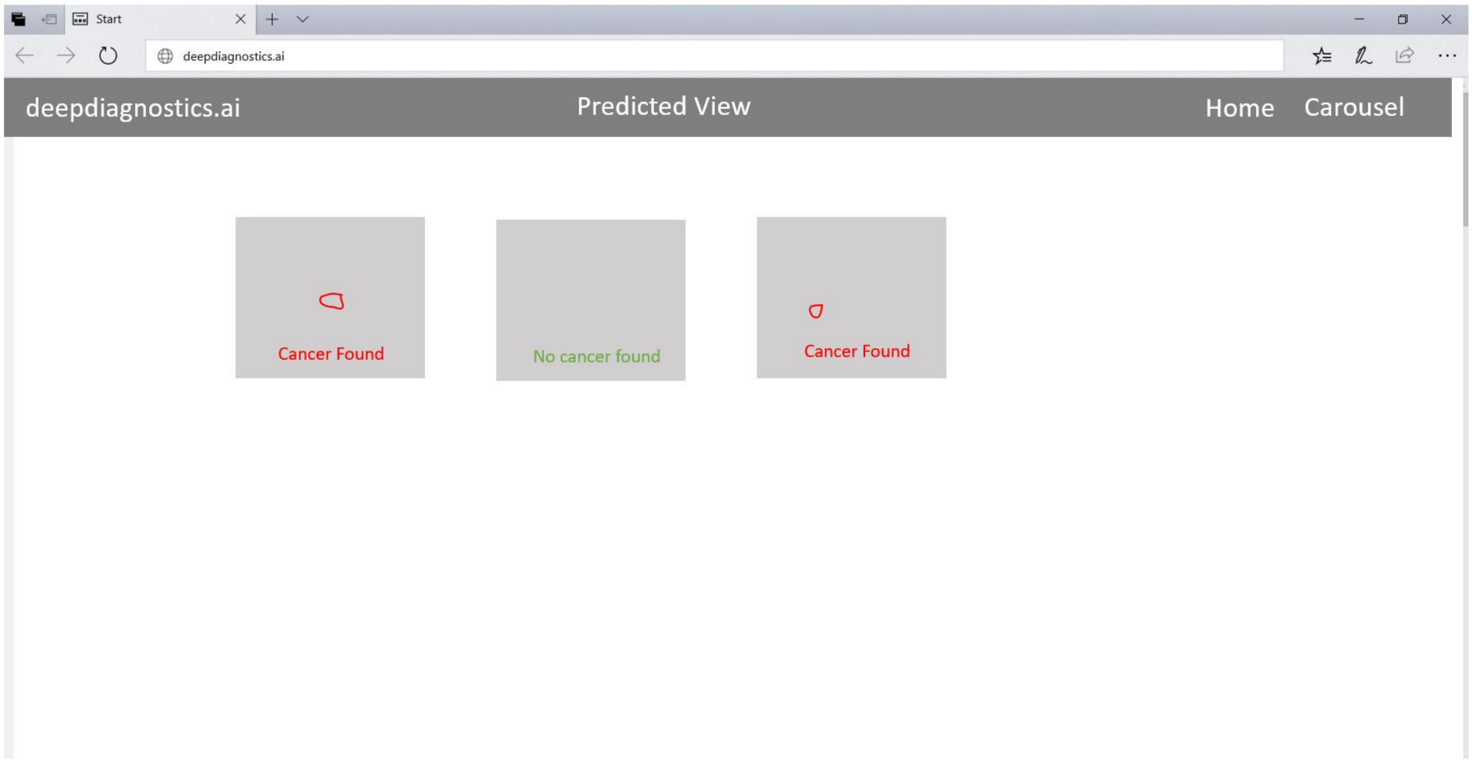


Figure 5.10: Prediction Output Gallery

Figure 5.10 shows the wireframe for the output of the model. It is similar to the CT scan gallery however the difference is that the user cannot select the image and will just view it. The images also are tagged with labels to see and is either tagged with cancer found or no cancer found.

5.5.6 Prediction Output Carousel

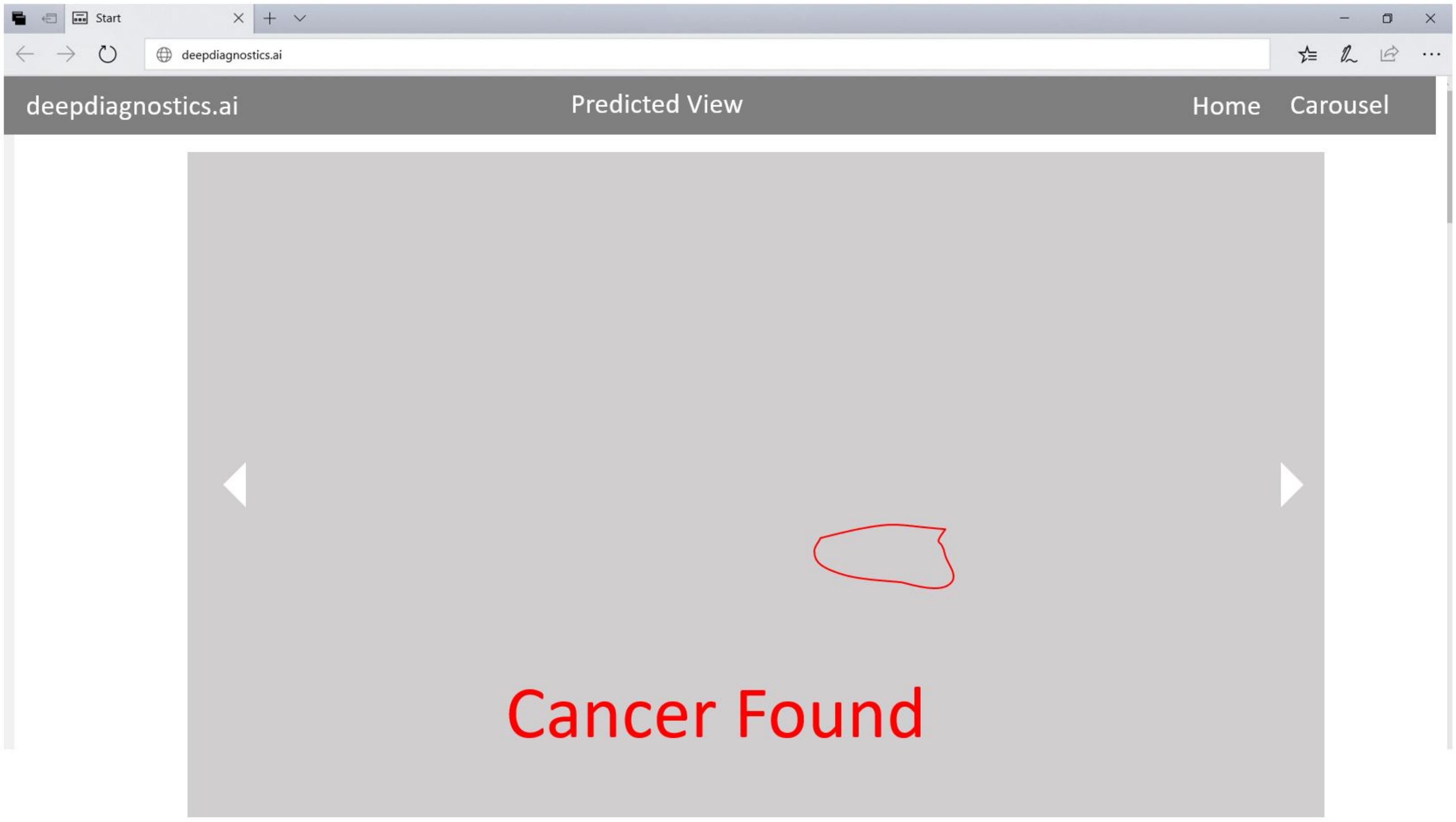


Figure 5.11: Prediction Output Carousel

Figure 5.11 shows the last wireframe of the application. This is another carousel view that gives the user an option to have a more close up view of the prediction images created by the model.

5.6 System Architecture

5.6.1 Model-View-Controller

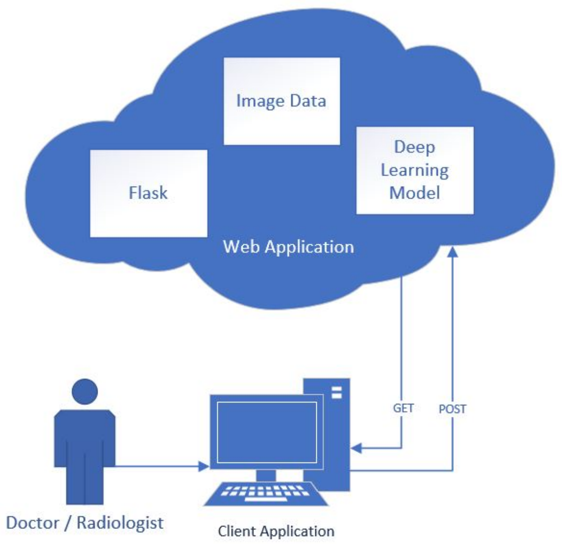


Figure 5.12: System Architecture

Figure 5.12 shows a system architecture design. The application is a standard web application that holds the deep learning model and image data that the user uploads via POST request. The user can then use GET request to get images from Flask to either view the CT scan images or view predictions made by the model.

The system utilises a MVC (Model-View-Controller) software design pattern that splits the code into 3 abstract components: model, view and controller.

5.6.1.1 Components

The model in the application is the images that we take from the user and the deep learning model itself. The model drives the main functionality and is central to the entire application.

The view in the application is the front-end and is what the user sees, the view uses HTML, CSS, Bootstrap and JavaScript to display to the user the output of the model.

The controller itself is the Flask back-end code. This is responsible for manipulating data to and from the model and view. In the application, Flask is responsible for the heavy lifting including showing the relevant front-end code through routing to using the model to make predictions.

5.7 Conclusion

This chapter outlines the design artefacts used for the project, with these artefacts the author would be able to better understand the user and the technical aspect prior to implementation. This ensures that the project is agile as possible, possibly giving more value to users and not heavily focused on the technical detail of the application.

The next chapter outlines the implementation of the system.

Chapter 6

Implementation

6.1 Overview

This chapter deals with the implementation process of the project. This chapter covers the data mining process and the software development process.
The tasks of the project have been broken down into the following work flow diagram. See Figure 6.1.
The Data Understanding and Data Preparation steps have been inspired and adapted from Kaggle by Mulholland et al [29] and Zuidhof [24].

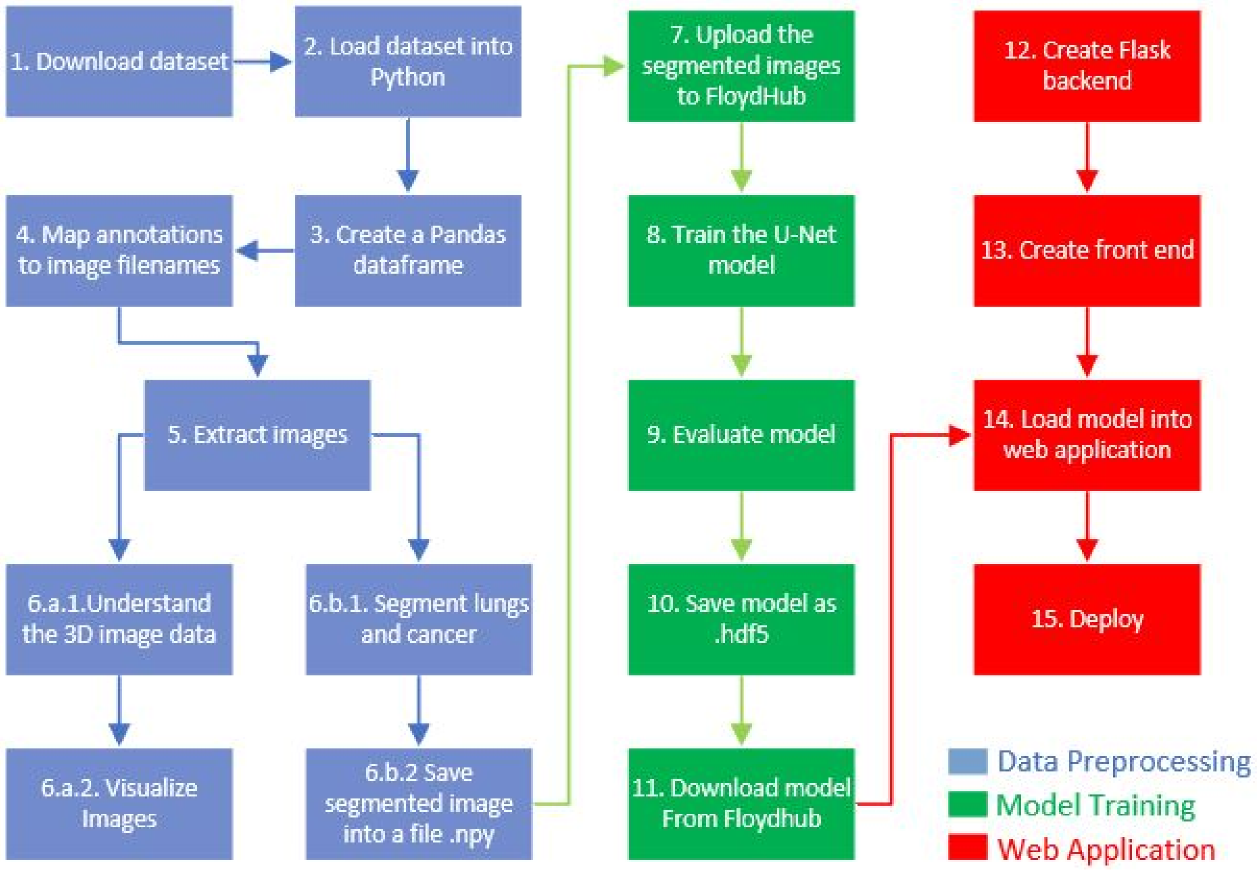

1. Download dataset
2. Load dataset into Python
3. Create a Pandas dataframe
4. Map annotations to image filenames
5. Extract images
6.a.1.Understand the 3D image data
6.a.2. Visualize Images
6.b.1. Segment lungs and cancer
6.b.2 Save segmented image into a file .npy
7. Upload the segmented images to FloydHub
8. Train the U-Net model
9. Evaluate model
10. Save model as .hdf5
11. Download model From Floydhub
12. Create Flask backend
13. Create front end
14. Load model into web application
15. Deploy
Data Preprocessing
Model Training
Web Application

Figure 6.1: Project Workflow

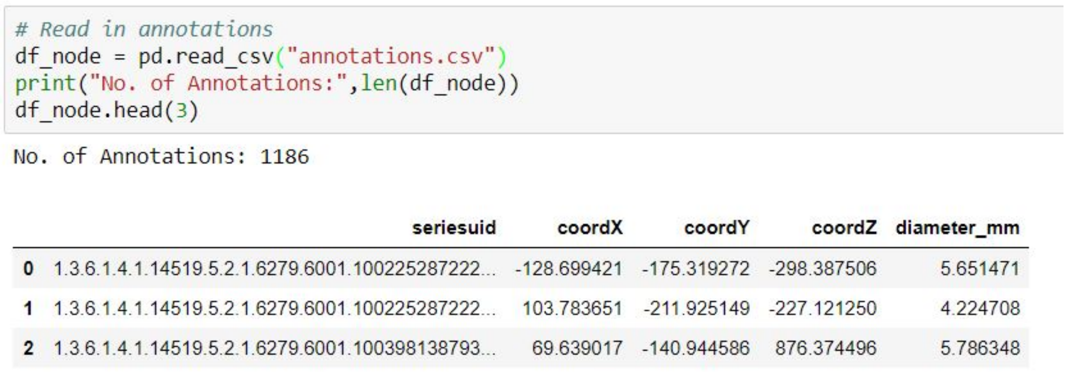
The goal of the data mining process is to produce a deep learning model which takes in an image and outputs a prediction. The software development process is used to output a web application as a proof of concept.

6.2 Data Understanding

This sections deals with the different ways the dataset is visualized and understood. Initially a subset was downloaded from LUNA2016. The subset contains annotations on the CT scans (annotations.csv), metadata file (*.mhd), and a raw file (*.raw).

6.2.1 Annotations

The annotations file give is more description of the cancer found in the dataset. The annotations.csv give the following label information: ID of the CT scan image, X coordinate,Y coordinate,Z coordinate, Diameter in millimetres, number of cancer instances.



```
# Read in annotations
df_node = pd.read_csv("annotations.csv")
print("No. of Annotations:",len(df_node))
df_node.head(3)
```

```
No. of Annotations: 1186
```

	seriesuid	coordX	coordY	coordZ	diameter_mm
0	1.3.6.1.4.1.14519.5.2.1.6279.6001.100225287222...	-128.699421	-175.319272	-298.387506	5.651471
1	1.3.6.1.4.1.14519.5.2.1.6279.6001.100225287222...	103.783651	-211.925149	-227.121250	4.224708
2	1.3.6.1.4.1.14519.5.2.1.6279.6001.100398138793...	69.639017	-140.944586	876.374496	5.786348

Figure 6.2: annotations.csv file

The annotations.csv can be analysed to get a better idea of the contents of the entire dataset. Figure 6.3 shows the cancer sizes in the dataset. The graph shows that the graph sizes is skewed to the left and is not a normal distribution. The figure also shows that cancer sizes range from 3mm - 32mm.

```
**************General Lung Cancer Data in Luna 2016**************
Average Size of Nodules in diameter(mm) = 8.306527315553957
Largest Sized Nodule in diameter(mm)= 32.27003025
Smallest Sized Nodules in diameter(mm)= 3.253443196
```

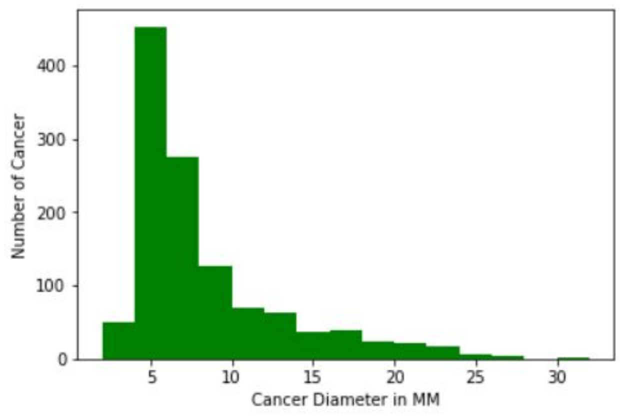
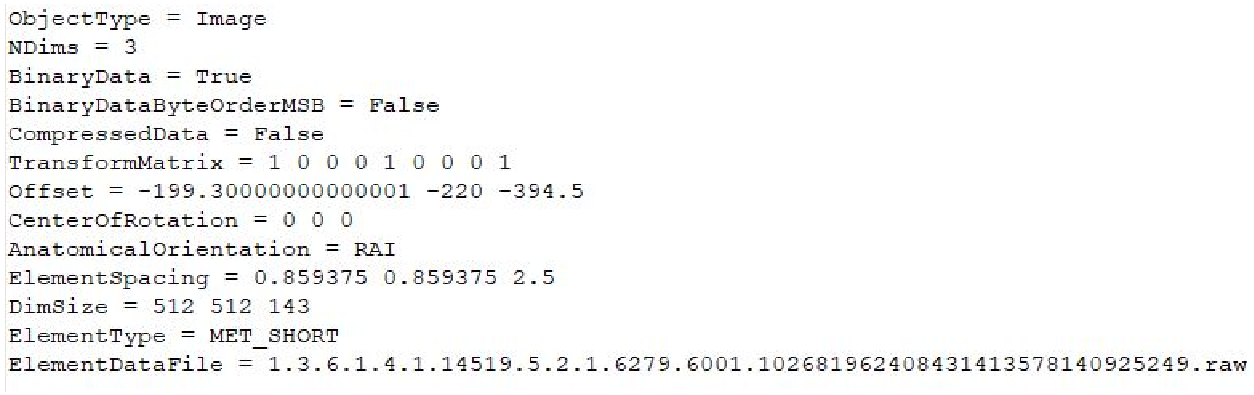


Figure 6.3: Cancer Sizes

6.2.2 Metadata

The metadata file gives more information about the raw CT scan images, see Figure 6.4. With this metadata file, the data can be loaded and be processed.



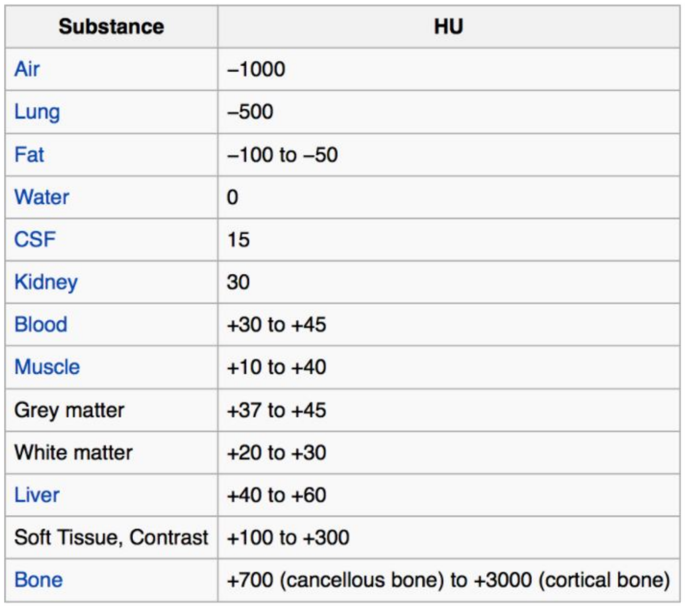
```
ObjectType = Image
NDims = 3
BinaryData = True
BinaryDataByteOrderMSB = False
CompressedData = False
TransformMatrix = 1 0 0 0 1 0 0 0 1
Offset = -199.30000000000001 -220 -394.5
CenterOfRotation = 0 0 0
AnatomicalOrientation = RAI
ElementSpacing = 0.859375 0.859375 2.5
DimSize = 512 512 143
ElementType = MET_SHORT
ElementDataFile = 1.3.6.1.4.1.14519.5.2.1.6279.6001.102681962408431413578140925249.raw
```

Figure 6.4: Sample Metadata File

6.2.3 Visualizations

6.2.3.1 Hounsfield Scale

The 3D images can be processed and segmented using the Hounsfield Scale. The Hounsfield Scale describes radiodensity[6], i.e. what matter is inside the CT Scans. Figure 6.5 shows the Hounsfield scale that helps radiologists analyse CT scans. For this project it will be used to filter out regions of interest.



Substance	HU
Air	–1000
Lung	–500
Fat	–100 to –50
Water	0
CSF	15
Kidney	30
Blood	+30 to +45
Muscle	+10 to +40
Grey matter	+37 to +45
White matter	+20 to +30
Liver	+40 to +60
Soft Tissue, Contrast	+100 to +300
Bone	+700 (cancellous bone) to +3000 (cortical bone)

Figure 6.5: Hounsfield Scale[7]

Figure 6.6 shows taking one instance of the 3D Image and plot what kind of substance is inside the images using the Hounsfield Scale and Matplotlib[6].
Figure 6.6 shows the substances, there is substance of foreign value -3000 due to the black circlar border, -1000 shows that there is air present in the lungs and a large peak at the 0 value which is mostly liquid substance.

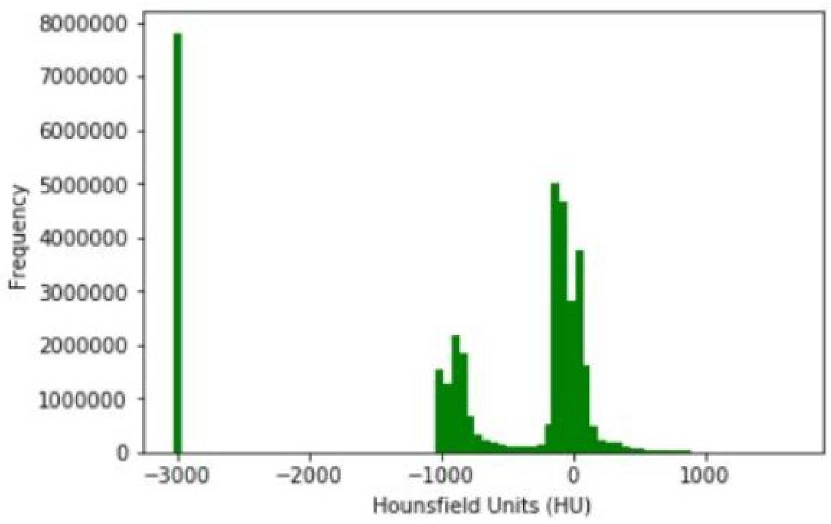
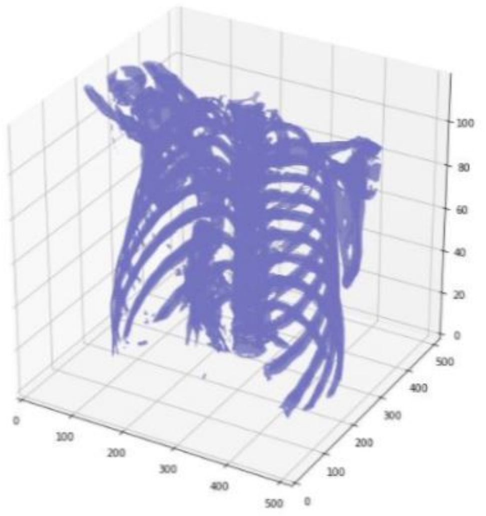


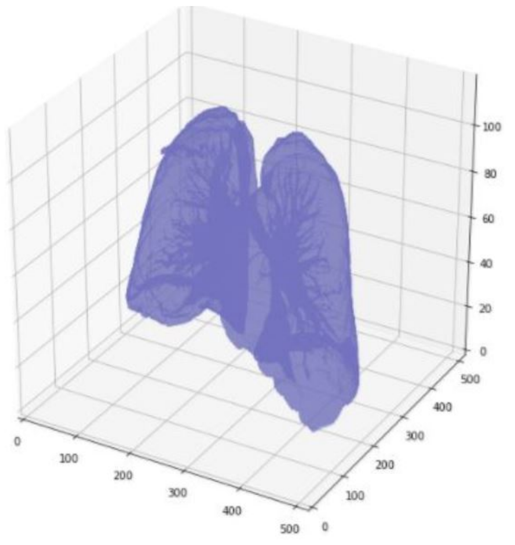
Figure 6.6: Substances within the Images using Hounsfield Scale

6.2.3.2 Hounsfield Segmentation

Using skimage and mpltoolkits helps to display the 3D image. Figure 6.7a shows the segmentation of Bones and Lungs using the Hounsfield Unit Scale. With these techniques, the complex nature of the data can be understood.



(a) Segmenting Bones



(b) Segmenting Lungs

Figure 6.7: Segmentation using Hounsfield Scale

6.2.3.3 CT Scan Slices

Figure 6.8 shows a set of images for a single CT scan. The dimensions for the CT scan is about 512 height, 512 width and approximately 200 images, although there also exists scans which are over 300 images.

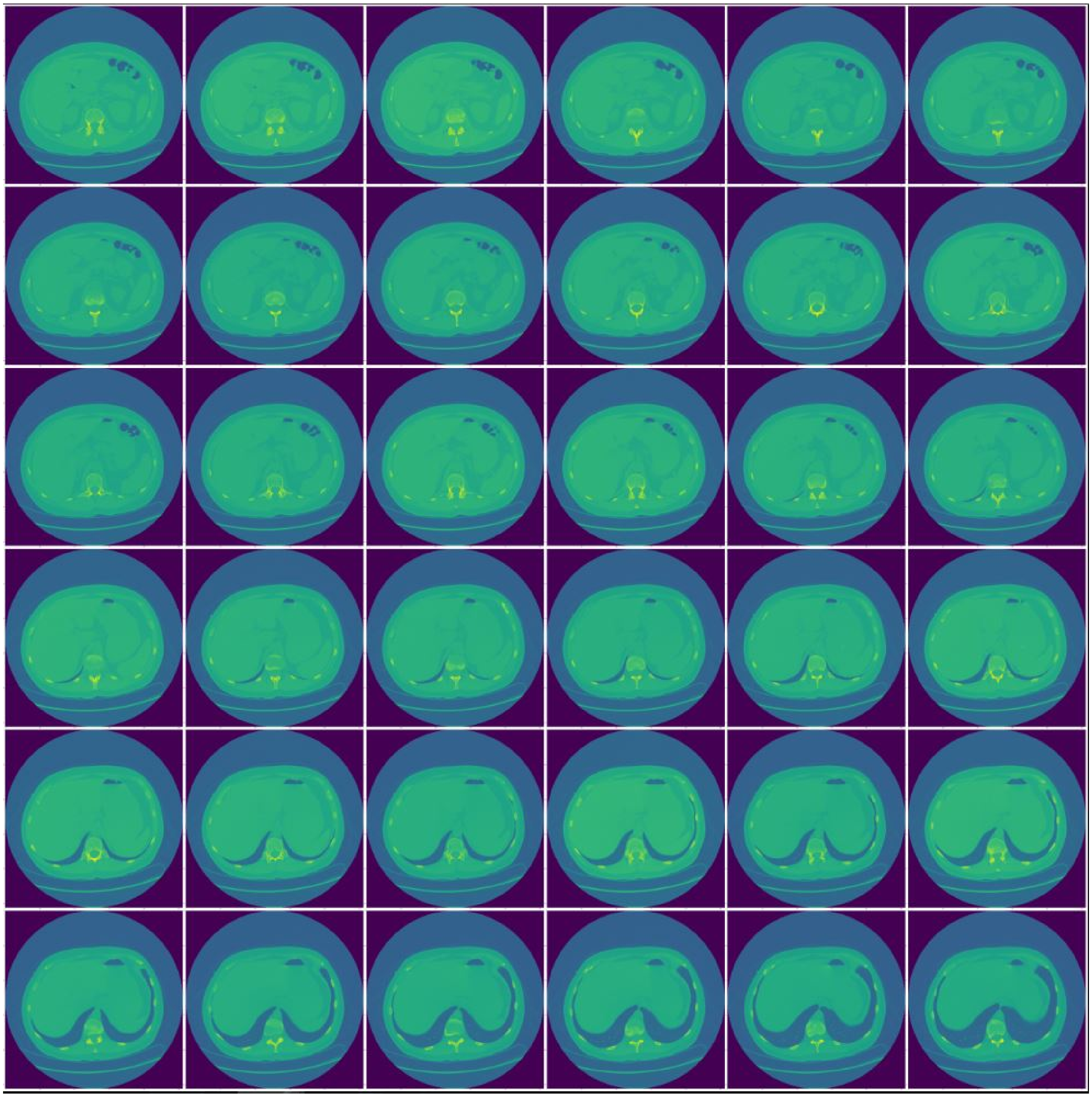


Figure 6.8: CT Scan Slices

6.3 Feature Engineering

This section details the feature engineering of the CT scans before it's used to train the model. The task for this section is as follows: 1. Create nodule mask, 2. Extract the lungs from original image. This process ensures that we have the necessary data to teach the model: cancer nodule masks and lung images segmented out of the CT scan slices.

6.3.1 Creating Nodule Mask

With the use of the annotations and Mulholland et al's makemask algorithm [29]. The author was able to extract a boundary around cancer nodules. A short explanation of masks and the makemask algorithm used is shown in the appendix.

Figure 6.9 shows sample images of cancer masks, the majority of which is small and some are large.

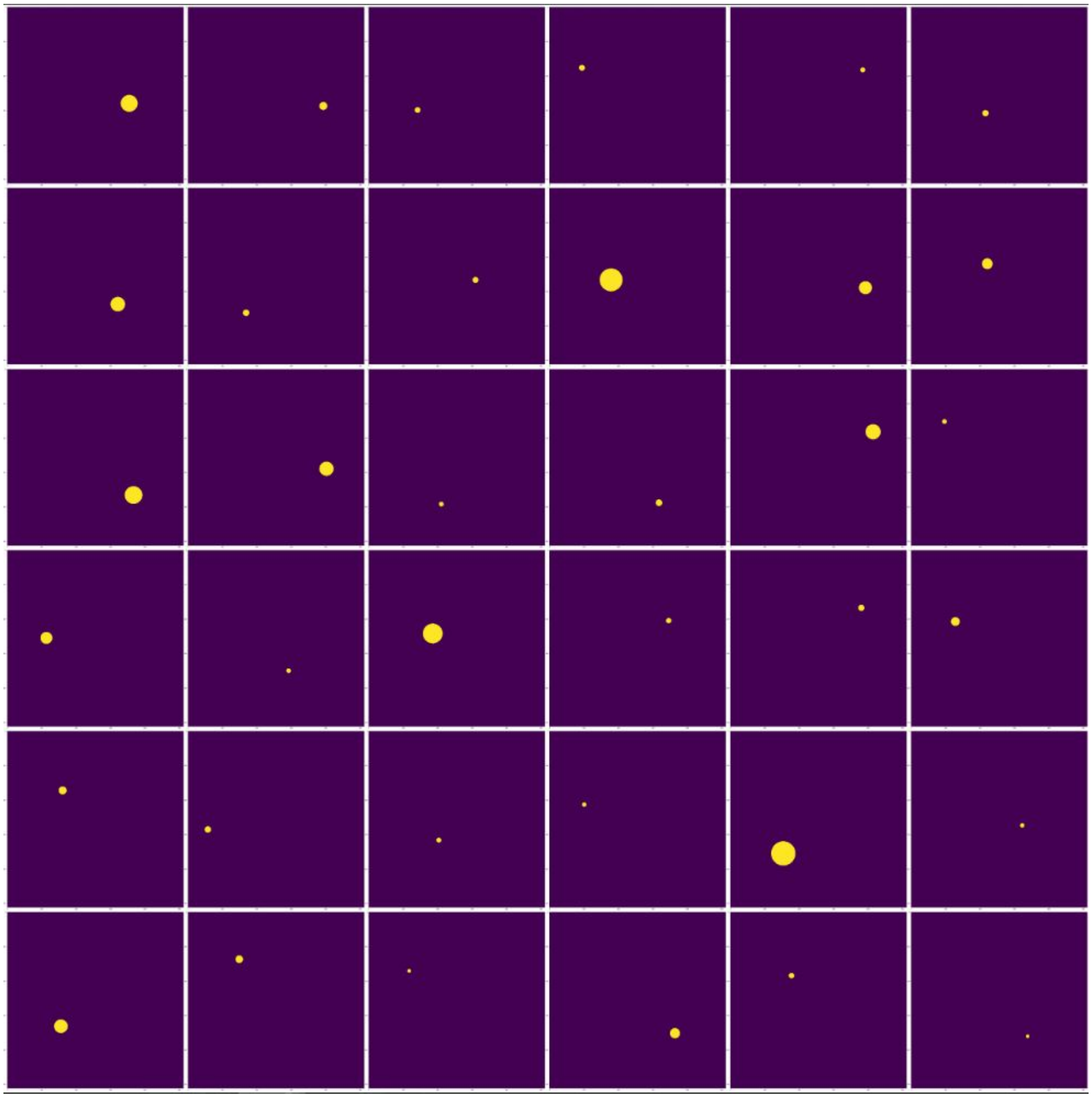


Figure 6.9: Cancer Masks (Label Set)

6.3.2 Creating Lung Images

The next step is to create our lung images segmented from our original image. This is also done via Mulholland et al's algorithm shown in the appendix section.

Figure 6.10 shows a sample images of segmented lungs with cancer, we can see some of the cancer is very small and hard to determine visually but some are very large and are clearly malignant.

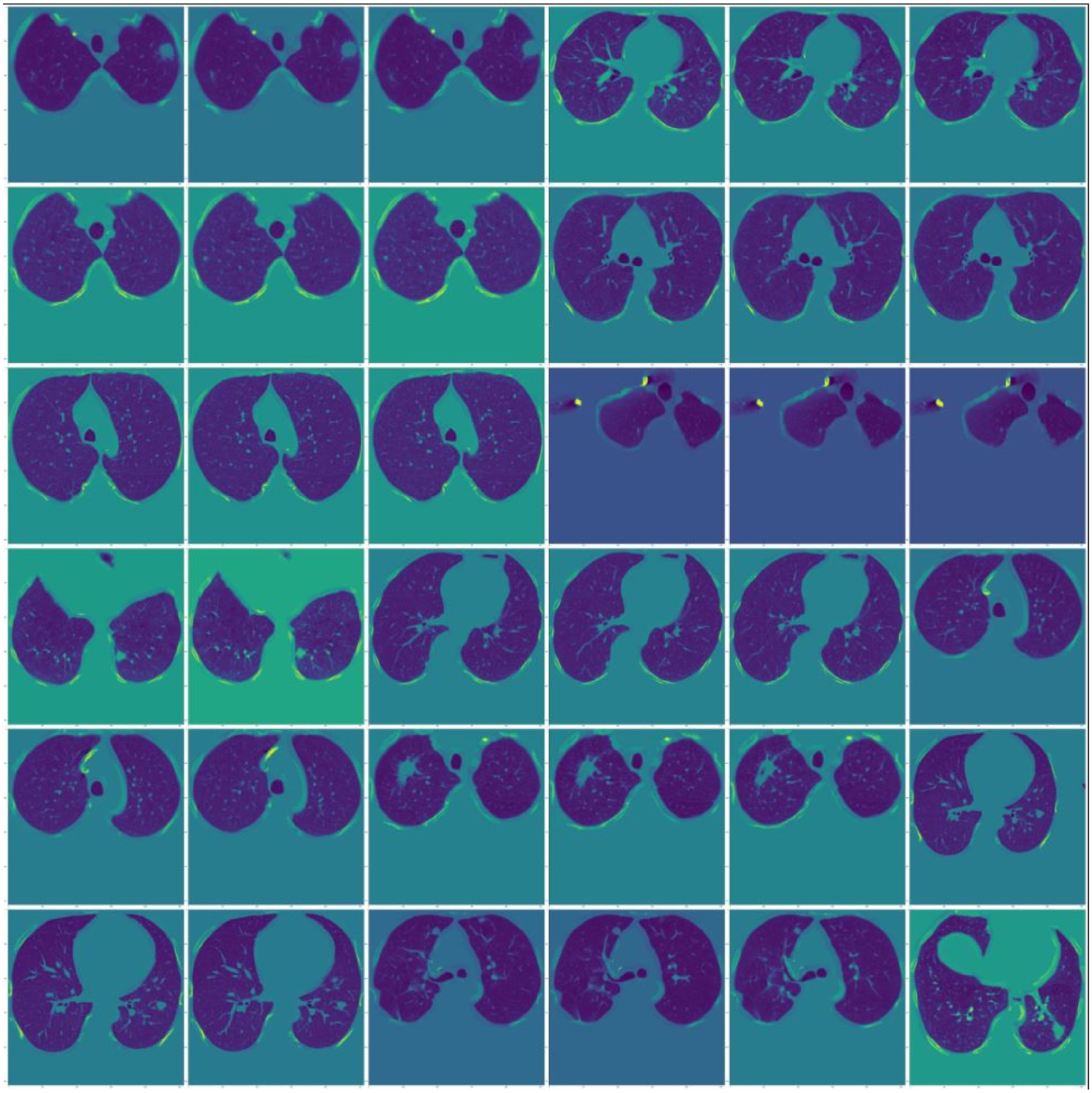


Figure 6.10: Lungs Segmented (Feature Set)

6.3.3 Data Preparation

After creating these cancer masks and lung images, these images are saved into a numpy array and uploaded into Floydhub as data sets ready for deep learning modelling. There are overall 1803 lung images and cancer masks (label) and we split this dataset into 98% training(1767 image and mask), 1% validation(18 image and mask), 1% test sets(18 image and mask). This way ensures we give the neural network as much data as possible and get enough data for evaluating the model.

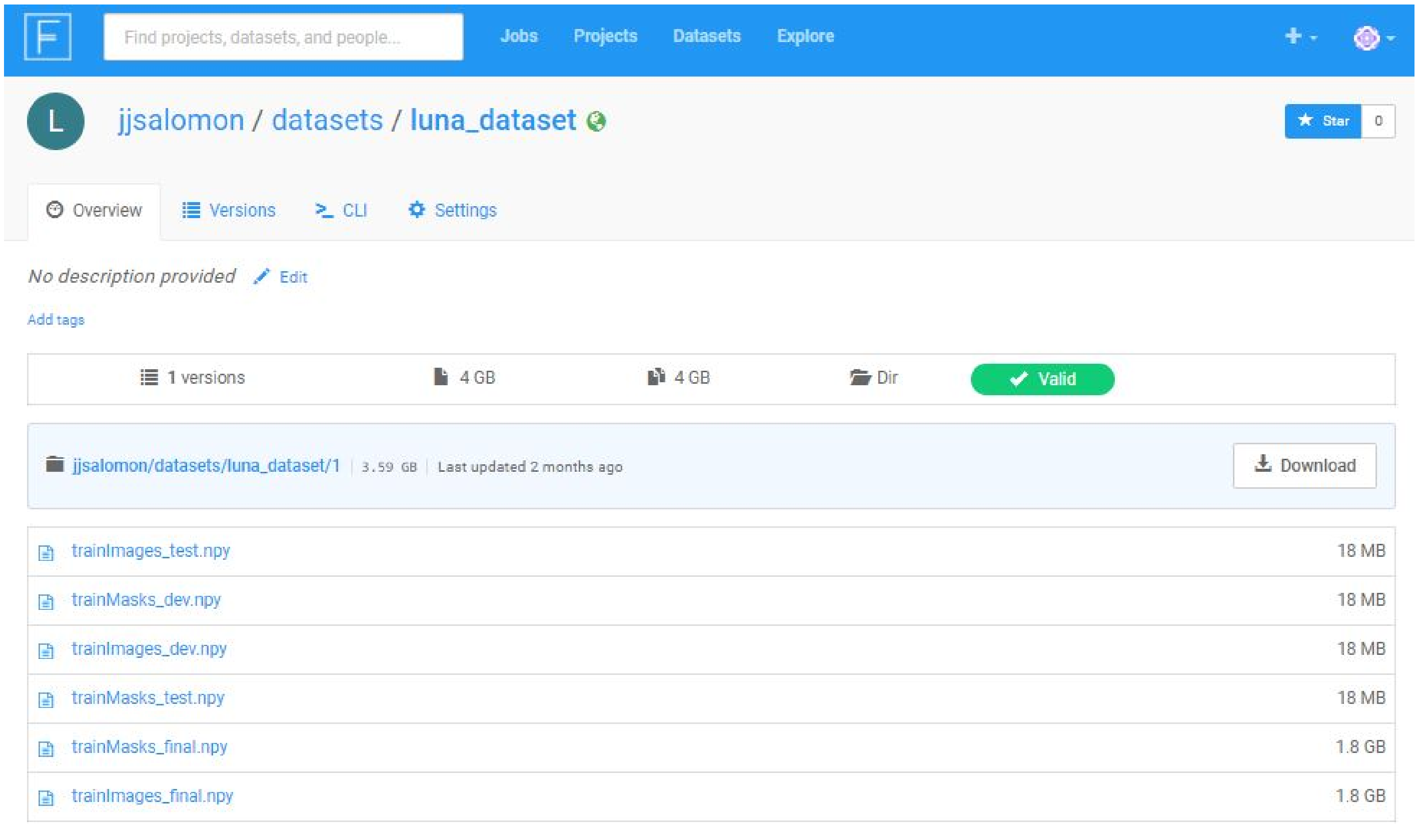
Creating these two datasets ensures that we teach the u-net model to find cancer masks in lung areas only. The network will not learn to search for cancer masks outside of the lungs. This reduces the search space for the neural network.

6.4 Modelling

This section details the creation of the U-Net model. Floydhub works by uploading a dataset and scripts into the platform. The platform then uses a GPU (Tesla K80) to run processes to train neural networks. This ensures that training is over 20 times faster compared to a CPU.

6.4.1 Floydhub Setup

Before the model can be trained, an account in Floydhub has to be created, Floydub cli installed and the prepared datasets need to be uploaded. See Figure 6.11. The Floydhub cli needs to be installed on the machine so that a job can be run which is further explained in the next section.



jjsalomon / datasets / luna_dataset

No description provided

1 versions | 4 GB | 4 GB | Dir

jjsalomon/datasets/luna_dataset/1 | 3.59 GB | Last updated 2 months ago

trainImages_test.npy	18 MB
trainMasks_dev.npy	18 MB
trainImages_dev.npy	18 MB
trainMasks_test.npy	18 MB
trainMasks_final.npy	1.8 GB
trainImages_final.npy	1.8 GB

Figure 6.11: Train,Dev,Test Sets Uploaded

6.4.2 Training

After the setup is completed, A Floydhub job can be run to train a model. The parameters ensure that a GPU is used with a tensorflow environment and the correct dataset loaded.

Listing 6.1: Running a Floydhub Job

```
floyd init unet-training
floyd run --gpu
          --env tensorflow-1.1
          --data jjsalomon/datasets/luna_dataset/1:input
          --tensorboard
          --mode jupyter
```

The U-Net model was created using Keras with a tensorflow backend. **??** shows the U-Net model code using Keras. This model was run in Floydhub using the uploaded dataset. The network is considered deep with 10 layers overall. The main features in this neural network are: convolution, maxpooling, merge layers, with relu and sigmoid activation functions. The model also uses an adam optimizer, a mini-batch parameter of 2 and a dice coefficient loss function. The goal is to increase the percentage of the cost function by changing different hyper parameters.

Listing 6.2: Training the Model

```
def train_model():
    # Training Set
    imgs_train = np.load('/input/trainImages_final.npy').astype(np.float32)
    imgs_mask_train = np.load('/input/trainMasks_final.npy').astype(np.float32)

    # Mean for data centeringm STD for data normalization
    mean = np.mean(imgs_train)
    std = np.std(imgs_train)

    # Standardize Data
    imgs_train -= mean
    imgs_train /= std

    # Load Model
    model = get_unet()
    model.summary()

    # Saving weights to unet.hdf5 at checkpoints
    model_checkpoint = ModelCheckpoint(output_path+'//unet_weights.hdf5',
                                       monitor='loss',
                                       save_best_only=True,
                                       save_weights_only = True)
    # Train
    model.fit(imgs_train, imgs_mask_train, batch_size=2, nb_epoch=40, verbose=1,
        shuffle=True, callbacks=model_checkpoint)
```

6.4.3 Training Results

The U-Net Model was trained using Jupyter Notebook on Floydhub. Training time was very time consuming even with a GPU instance in Floydhub. The author had only allocated about 110 GPU hours on Floydhub for this project.

Num	Initialization	Regularizer	Optimizer	Batch Size	Learning Rate	Epochs	Training Time	Mean Dice Coef
1	Random	None	Adam	2	1.0e-5	10	10 hours	0.0016%
2	Random	None	Adam	2	1.0e-5	20	20 hours	0.0016%
3	Random	None	Adam	2	0.001	6	6 hours	0.0016%
4	Random	Dropout(0.2)	Adam	2	0.001	10	10 hours	0.0016%
5	He	Dropout(0.2)	Adam	2	1.0e-5	20	20 hours	0.387%
6	He	Dropout(0.2)	Adam	2	1.0e-5	40	40 hours	0.657%

Table 6.1: Experiments with Hyperparameters and Accuracy Result: Model Evaluation.ipynb

Table 6.1 shows the results of training the U-Net model with different hyper parameters.
Initially the author had thought that the learning rate was the issue and the number of epoch. The learning rate was then decreased and the number of epoch increased. Even with these changes the model would not converge.

The author suspected the model was stuck on a saddle point initially so it had a very difficult time in converging. The results leads to believe that it was because the random weights initialization that caused it to converge slowly. When He initialization was implemented it converged more per iteration.

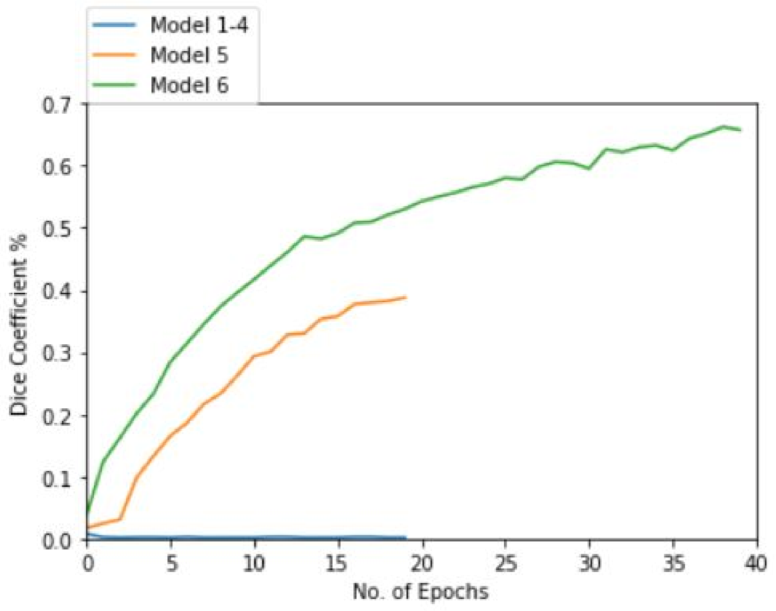


Figure 6.12: Performance per Iteration

After exhausting all the GPU hours at Floydhub, model 6 was the best performing model overall. See Figure 6.12. The graph seems to be converging as expected which leads the author to believe that more updates to the weights was needed as once the model broke out of the saddle point and starting learning, it kept getting more accurate the more epochs it achieved. There were many more experiments the author wanted to do such as increasing the rate of dropout from 0.2 to 0.5 and increasing the batch sizes of the mini-batch gradient descent algorithm. The author also suspects that because the batch size was only 2 the training was slower and be more highly likely to get stuck on a saddle point. However a small batch size allowed the model to have no fluctations.

The author decided to evaluate model 6 on the validation and test set to check for overfitting in the next section.

6.4.4 Model Evaluation

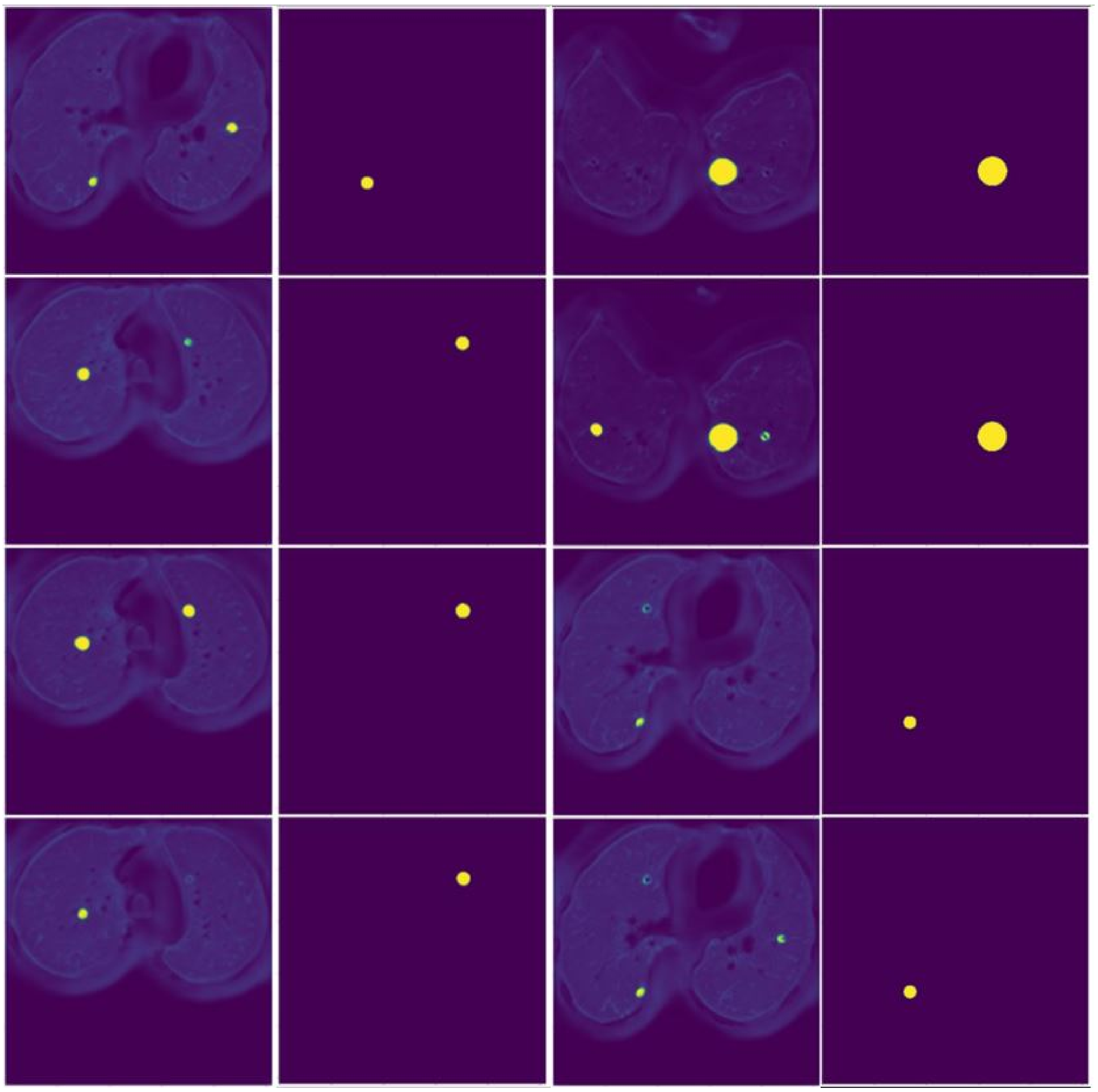


Figure 6.13: U-Net Predictions vs Cancer Masks(Label) Examples - Positive Samples

Figure 6.13 shows the model predictions beside the label. There are a high number of false positives that appear. The shape of the lungs appear also. This could be because the accuracy is only around 65% however this could be easily removed by rounding the image pixel values in a post processing step to create a mask.

Table 6.2 shows the metrics: TPR, TNR, FPR, FNR and Mean Dice Coefficient for the Dev Set and Test Set including results for post processing. An interesting observation is that the model is able to gain predict the true positive cancer nodule within the scan. The data consists of about 18 positive labels (with cancer) on each dev and training set and about 20 negative images (without cancer) which approximates to about 28 images per set. The positive samples were taken from the subsets of the training set and the negative ones were taken from random CT scans.

Data	Post Processing	TPR	TNR	FPR	FNR	Dice Coefficient
Dev Set	None	0.833%	0.256%	0.743%	0.166%	0.0003%
Test Set	None	0.928%	0.312%	0.687%	0.0714%	0.00062%
Dev Set	Int Rounding	0.388%	0.833%	0.166%	0.61%	0.183%
Test Set	Int Rounding	0.529%	0.833%	0.166%	0.470%	0.105%

Table 6.2: Model Evaluation: True Positive Rate, True Negative Rate, False Positive Rate, False Negative Rate and Mean Dice Coefficient

The metrics given at Table 6.2 give an indication that the model is able to a high percentage of accuracy when detecting lung cancer nodules. However it also detects a large amount of false positives. The data further indicates that the model is able to distinguish between a positive scan and a negative scan as indicated by a low percentage of low true negative rate.

The method of post processing the predicted image give an output of a higher dice coefficient compared to the sets without post processing. This can indicate that a Dice Coefficient would not be a suitable metric for evaluating the predicted outputs. However further research by Setio et al[38] suggests that using a FROC (Free-Response Receiver Operating Characteristic) analysis is the better option.

The goal for the post processing is to manually filter out false positives that arrive in the CT Scan. It works however they do filter out many of the true positives also. The current post processing method is not valid and it would be better to use different kinds of image filters to manually reduce false positives.

6.4.5 File Artefacts

When the model is trained, the model is serialized into a JSON file. This acts as a blueprint of the model. Listing 6.3 shows how the model is serialized to JSON.
The weights are also serialized into a HDF5 file which is loaded with the model file. Listing 6.4 shows how the weights are saved during training of the network.

With these two artefacts, the deep learning model can be integrated into an application explained in the web application development sections of this chapter.

Listing 6.3: Serializing Model Blueprint

```
model = get_unet()                    # Load model blueprint
model_json = model.to_json()          # Serialize Keras model to JSON
with open("model.json", "w") as json_file:
    json_file.write(model_json)       # Output to a file
```

Listing 6.4: Saving Weights During Training Using Keras.Callbacks

```
from keras.callbacks import ModelCheckpoint

model_checkpoint = ModelCheckpoint(output_path+'//unet_weights.hdf5',
                                   monitor='loss',
                                   save_best_only=True,
                                   save_weights_only = True)
```

6.5 Software Development

In this section, the steps for developing the web application is outlined. The features are detailed with reference to their underlying technologies. The main controller in this application is Flask which interacts with the front end built by HTML,CSS,Javascript, Bootstrap, jQuery. Tensorflow is also integrated with Flask to ensure that the deep learning functions as intended in the application. This is detailed in further sections.

6.5.1 Setup

Before development a virtual environment is created. A virtual environment ensures that this project contains their own isolated Python environment and packages. This can be done using Anaconda. The packages can then be saved into a requirements.txt file and can be installed. Listing 6.5 shows how to create a virtual environment using Anaconda and installing a text file of Python packages.

Listing 6.5: Python Environment Setup

```
conda create −−name py35 python=3.5          # Creating Python 3.5 VE
pip install −r requirements.txt              # Installing Packages
source activate py35                         # Activate Environment
```

6.5.2 Deep Learning Integration

Listing 6.6 shows how to load a serialized deep learning model, their associated weights and return the necessary variables for the application to run model functions. The graph variable in line 8 allows applications to integrate deep learning methods into an application and ensure that the application runs in the appropriate thread.

Listing 6.6: Initializing a Deep Learning Instance In Flask App

```
1 def initialize():
2     # Loading in the model
3     json_file = open(model_path + ’model.json’,’r’)
4     loaded_model_json = json_file.read()
5     json_file.close()
6     loaded_model = model_from_json(loaded_model_json)
7     loaded_model.load_weights(model_path+”weights.hdf5”)
8     graph = tf.get_default_graph()
9     return loaded_model, graph
```

When predicting, the graph variable is called to override Python’s original context manager. The code ensures that the deep learning object works in sync with Flask’s thread.

6.5.3 Feature 1: CT Scans Upload

The upload functionality is the first feature implemented in the web application.
The front-end allows the user to select files in their computer and this file is sent to the back-end via POST request. Listing 6.7 outlines how a file upload functionality is created in the front-end.

Listing 6.7: Upload Functionality in Frontend(index.html)

```
<form action="/upload" method=post enctype=multipart/form-data>
        <div class="row">
                <div class="col-md-4" id="pupload">
                        <p id="inputb" > Select Mhd File </p>
                        <input class="btn btn-primary"  type=file name=mhd>
                </div>
                <div class="col-md-4" id="pupload">
                        <p id="inputb"> Select Raw File </p>
                        <input class="btn btn-primary"  type=file name=raw>
                </div>
                <div class="col-md-4 text-center">
                        <p id="inputb">&nbsp</p>
                        <input id="uploadbutton" class="btn btn-primary" type=submit">
                </div>
        </div>
</form>
```

Listing 6.8 show's the Flask backend code for capturing the files. The POST request triggers a route function called upload within the Flask routes and the files sent by the user are captured via python requests and saved locally within Flask server side. The file is then loaded in using SimpleITK which is a python library used working with medical imaging. The parameters it takes is a the filename for the CT scan and the output is a raw 3 dimensional array of image values. These values can be further decomposed to create the individual CT scan slices images and also their numpy array files.

Listing 6.8: Flask Capture Route(server.py)

```
@app.route('/upload',methods=['POST'])
def upload():
    if request.method == "POST":
        # Get the image from the request
        file_mhd = request.files['mhd']
        file_raw = request.files['raw']

        # Get filenames
        filename_mhd = secure_filename(file_mhd.filename)
        filename_raw = secure_filename(file_raw.filename)
        # Save files in a folder
        file_mhd.save(os.path.join(app.config['UPLOAD_FOLDER'], filename_mhd))
        file_raw.save(os.path.join(app.config['UPLOAD_FOLDER'], filename_raw))
        # Read in the 3 Dimensional Dataset
        itk_img = sitk.ReadImage(str(uploads_path+filename_mhd))
        img_array = sitk.GetArrayFromImage(itk_img)
        # Number of image, Height, Width
        num, height, width = img_array.shape

        # Create images and filenames
        imgs, names,imgs_filenames_paths = createImages(num,img_array,imgs)
        # Add a timestamp on the filenames to help the front end renew the cache
        imgs_filenames = [name +"?"+str(datetime.datetime.now().time()) for name in names]
        return render_template("displayalpha.html",urls=zip(imgs_filenames,names)
```

6.5.4 Feature 2: CT Scan Gallery

The CT Scan gallery is triggered at the end of the routes of the upload function. Flask returns a HTML template with certain url parameter which is seen in Listing 6.8 line 27.

These url parameter are important as they allow the front-end template to perform GET requests for the image resources on Flask. Listing 6.9 shows a controller on the front-end that renders the bootstrap gallery. The controller on the front-end side is Jinja templates that is available with Flask. Jinja allows the front-end to hold data from the back-end and apply some basic logic. In this case a for loop is applied to send a GET request for each CT scan image and render it shown in line 1.

Listing 6.9: Rendering CT Scan Gallery Images using Jinja and Bootstrap

```
1 {% for image in urls %}
2 <div id=”selectable_images” class=”col-lg-3 col-md-4 col-xs-6”>
3         <img id=”galleryimage” class=”thumbnail_img”
4                  src= ”../static/alpha_view/{{image.0}}”
5                  name=”{{image.0}}”
6                  onclick=”addImageToArray(''+this.name+'')”>
7         <p class=”m-0 text-center”>{{image.1}}</p>
8 </div>
9 {% endfor %}
```

This feature is also available for predicted images created by the model which is further explained.

6.5.5 Feature 3: Carousel View

A carousel was implemented to help doctor's better visualize the CT scan in a sequential manner. The carousel was implemented using Bootstrap 3. The carousel view can only be viewed when users upload a CT scan and are able to see scans using the gallery. The user starts this flow by clicking on view carousel on the navbar. This triggers a new route from Flask for a new template with the image file names as parameters. The file names populate an image tag on the front end which then trigger GET requests for the images. Bootstrap carousel allows the user to click on left and right or left and right on their keyboards to change images. Listing 6.10 outlines how a Bootstrap carousel can be loaded using Jinja.

Listing 6.10: Rendering Bootstrap Carousel using Jinja

```
<div id=”myCarousel” class=”carousel” data-ride=”carousel”>
<div class=”carousel-inner”>
    {% for image in urls2 %}
        {% if loop.first %}
          <div class=”item active”>
            <img src=”../static/alpha_view/{{image.0}}” style=”width:100%;”>
            <div class=”carousel-caption”>
              <h3>{{ image.1 }}</h3>
            </div>
          </div>
        {% else %}
          <div class=”item”>
            <img src=”../static/alpha_view/{{image.0}}” style=”width:100%;”>
            <div class=”carousel-caption”>
              <h3>{{ image.1 }}</h3>
            </div>
          </div>
        {% endif %}
    {% endfor %}
</div>
</div>
```

6.5.6 Feature 4: Predictions

This is the final feature of the web application. When a user wants to run a prediction. A user can click on multiple images within the gallery view. These images are chosen on click and their file names are appended into a JavaScript array.

After a user has completed choosing, he or she can click on the predict button which triggers a number of events. First, the file names array sent to Flask via a POST request this is done using Ajax in jQuery.

Listing 6.11 shows the back-end for predicting the images sent by the user. Flask retrieves the corresponding numpy array files. These files are the pure image array data of the CT scan that the user has chosen and is then piped into the model for predictions seen on line 26. Line 25 shows how to integrate the tensorflow model into the application, without this the model predict function will not work.

The output of the model is then prepared and saved into an image folder. Once the images has been saved, triggers a new route where it takes all the predicted images and sends their file names to the predicted gallery or predicted carousel which then loads the image via GET requests.

Listing 6.11: Predicting Images Chosen by User

```
1  @app.route('/getImageFilenames',methods=['POST'])
2  def getImageFilenames():
3      if request.method == "POST":
4          # Retrieves filenames from the POST request
5          data = request.get_json()
6          filenames_array = data['filenames']
7
8          numpy_array_data = [s.replace('.png','.npy') for s in filenames_array]
9
10         # Check if folder has data, if data -- delete
11         files = glob(predicted_path+"*.png")
12         if files != 0:
13             for f in files:
14                 os.remove(f)
15
16         pr_path_images = [image_data_path + i  for i in numpy_array_data]
17         # Container for our images before feeding to model
18         images_for_model = np.ndarray([len(filenames_array),1,512,512],dtype=np.float32)
19
20         # Preprocess the data on the container, this is loading the image correctly..
21         for count, img_path in enumerate(pr_path_images):
22             images_for_model[count,0] = np.load(img_path)
23
24         # Predict using model and tensorflow graph
25         with graph.as_default():
26             predicted = model.predict(images_for_model,verbose=0)
27
28             # Reshape the predicted, write out image, get paths to the images
29             for count, data in enumerate(predicted):
30                 image_predicted = data.reshape((512,512))
31                 formatted = (image_predicted * 255 / np.max(image_predicted)).astype('uint8
32                 img = Image.fromarray(formatted)
33                 path = predicted_path+filenames_array[count]
34                 img.save(path)
35             return jsonify(images_for_model.shape)
```

The deep learning model outputs a mask that is saved and used to make create a contour for the original image. A.5 and A.6 explain how masks and contours behave. Listing 6.12 shows how to use the predicted mask and apply a contour on the original image.

Listing 6.12: Applying Contour(server.py)

```
for i in masks_filenames:
    origin = cv2.imread(alpha_data_path + str(i))        # Loading the alpha image -4 chann
    mask = cv2.imread(predicted_path + str(i))           # Load Masks
    imgray = cv2.cvtColor(mask,cv2.COLOR_BGR2GRAY)       # Preprocessing
    ret, thresh = cv2.threshold(imgray,10,255,0,)

    image, contours, hierarchy = cv2.findContours(thresh,
    cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)              # Create Contours
    contour_img = cv2.drawContours(origin,
    contours, contourIdx=-1, color=(0,0,255), thickness=2)
        font = cv2.FONT_HERSHEY_SIMPLEX
    if hierarchy is None:                                # if contours doesnt exist
        # Draw green no cancer
        cv2.putText(origin,'No Cancer',(10,500), font, 1,(0,0,255),2,cv2.LINE_AA)
    else:
        # Draw red found cancer
        cv2.putText(origin,'Cancer Found',(10,500), font, 1,(0,0,255),2,cv2.LINE_AA)
    cv2.imwrite(results_data_path+str(i),origin)
```

6.5.7 Application Evaluation: Unit Testing

The web application is evaluated using unit tests. Unit tests test individual blocks of code in the application. The project uses Python unittest library. The test cases used for the web application is shown in Table 6.3. These tests were conducted to ensure that the basic functionality and structure of the web application is working as intended. Like the other technologies used for this project, Python's unittest library is also quite simple and minimalist to use.

Num	Test Case	Test Result
1	Check Image Folder Exists	PASS
2	Load Index Route	PASS
3	Load Gallery Route	PASS
4	Load Carousel Route	PASS
5	Load Predict Gallery Route	PASS
6	Load Predict Carousel Route	PASS
7	Load Upload Page	PASS
8	Image File Parser	PASS
9	Load CT Scan	PASS
10	Predict Functionality Route	PASS

Table 6.3: Test Cases

6.6 Conclusion

This chapter details technical implementation of the project. Although the deep learning model only performs on a 65% accuracy on the training set it is still able to create masks and find cancer within new instances given the risk of high false positives. The model was then integrated into a web application and performs the main functionality for this project.

In the next chapter the project plan will be discussed.

Chapter 7

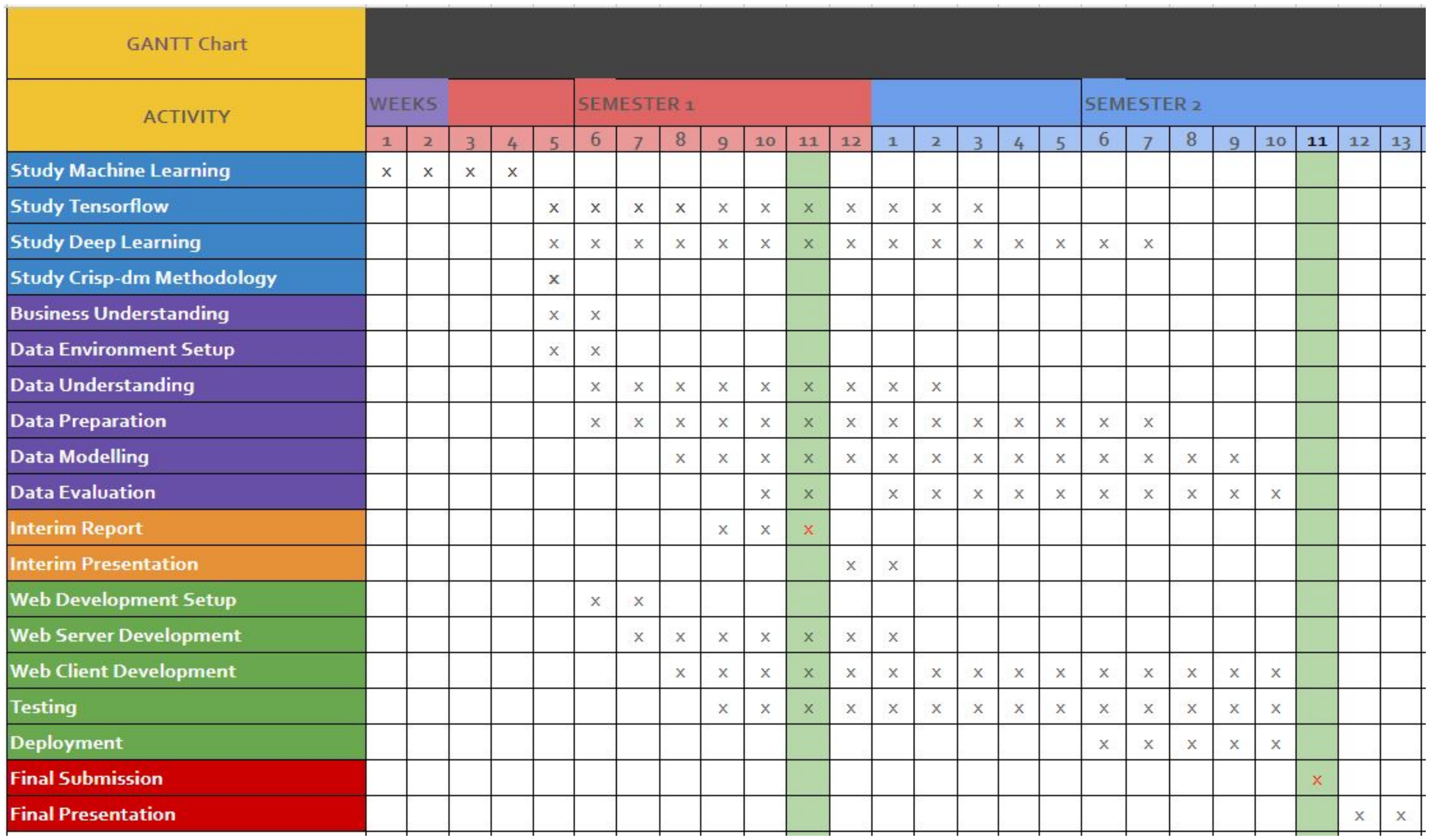
Project Plan

7.1 Introduction

This chapter details the project plan and reviews the different changes that occurred within the entire development lifecycle of the project.

7.2 Gantt Chart Plan

Figure 7.1 shows how the projects tasks were broken down using a Gannt Chart. The chart allowed a brief estimation for the amount of work that had to be done for a task. Some suggestions put forward in the Gantt chart were not implemented however the main objectives were met.



GANTT Chart																									
ACTIVITY	WEEKS					SEMESTER 1												SEMESTER 2							
	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12	13
Study Machine Learning	x	x	x	x																					
Study Tensorflow					x	x	x	x	x	x	x	x	x	x	x										
Study Deep Learning					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x						
Study Crisp-dm Methodology					x																				
Business Understanding					x	x																			
Data Environment Setup					x	x																			
Data Understanding						x	x	x	x	x	x	x	x	x											
Data Preparation						x	x	x	x	x	x	x	x	x	x	x	x	x	x						
Data Modelling								x	x	x	x	x	x	x	x	x	x	x	x	x	x				
Data Evaluation										x	x		x	x	x	x	x	x	x	x	x	x			
Interim Report									x	x	x														
Interim Presentation												x	x												
Web Development Setup						x	x																		
Web Server Development							x	x	x	x	x	x	x												
Web Client Development								x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			
Testing									x	x	x	x	x	x	x	x	x	x	x	x	x	x			
Deployment																		x	x	x	x	x			
Final Submission																							x		
Final Presentation																								x	x

Figure 7.1: Project Gantt Chart

7.3 Project Plan Review

The data science tasks took much longer to achieve for the project. Although there is a generous amount of time allocated for it, much focus was taken from the web development tasks in the project initially.The main reason is the time taken for Data Modelling. Running deep learning experiments is a very iterative and time consuming process. This is because models require hyperparameter tuning and hours of training time. My lack of experience training deep neural networks also impacted this. The web development integration had to be postponed at a much later date but the features planned to be implemented on the website took less time to be completed. This was due to the minimalist nature of Flask which enables developers to rapidly develop prototypes. A task that was not estimated properly was the final year report as the documentation took much longer than anticipated to finish which led to deployment not being done.

7.4 Conclusion

This section details how the author estimated and de constructed the tasks for the project. The gantt chart was heavily used to estimate the duration of activities and with this artefact the great majority of the tasks has been achieved.
The next chapter, concludes the entire project.

Chapter 8

Conclusion

8.1 Introduction

This chapter aims to discuss the model results, an evaluation of the proof of concept, future work to improve the application and a personal statement.

8.2 Model Results

The model resulted in a 65.7% accuracy using the dice coefficient on the training set. The dice coefficient is much lower on the training set however the confusion matrix outputs a high true and false positive rate on a set that contains positive and negative samples. This indicates that the model is great at distinguishing between CT scan slices with no cancer nodules compared to the ones with cancer. I believe with more hyperparameter tuning and model training the accuracy could be increased.

8.3 Concept Evaluation

When doctors find small nodules (less than 3mm) the current practice suggests that they should wait and rescan in 6-12 weeks to see signs of growth. Depending on the tumour, a tumour can grow up to double its size and evolve to a more advanced form of cancer. It is also important to note that the second most frequent diagnosis is small tumours. The project demonstrates that it would be possible for Doctor's to use deep learning applications to aid their decision making process regarding whether a patient with a small tumour should perform a biopsy or rescan in a few weeks which to a patient could mean early treatment and a better prognosis.

8.4 Future Work

Doctor's who work in this field are prone to observer fatigue from viewing so many CT scan images. The research on that suggests that observer fatigue increases the risk of errors that can be made by doctors while analysing these scans. Many images in a CT scan also are irrelevant to Doctors e.g. for 200-300 images only 3 scans would show cancer depending on the stage of the patient. Although this feature was not implemented on the website, a more efficient deep learning model would be capable of alleviating these additional challenges.

8.5 Personal Statement

During the course of the entire project I have learned new skills in areas of deep learning, machine learning, image processing, web development and also research. Being able to blend multiple skills in computer science and produce a proof of concept to try and solve a real world problem is really challenging but also provides the best learning experience. I do believe that anyone who gets involved in Computer Science has a large ability to solve real problems and make the world a better place.

Appendix A

Appendix Chapter

A.1 U-Net Code

Listing A.1: U-Net Model Code in Keras

```
def get_unet():
# Input Layer
    inputs = Input((1,img_rows, img_cols))

# Layer 1
    conv1 = Convolution2D(32, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(inputs)
    conv1 = Dropout(0.2)(conv1)
    conv1 = Convolution2D(32, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

# Layer 2
    conv2 = Convolution2D(64, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(pool1)
    conv2 = Dropout(0.2)(conv2)
    conv2 = Convolution2D(64, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

# Layer 3
    conv3 = Convolution2D(128, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(pool2)
    conv3 = Dropout(0.2)(conv3)
    conv3 = Convolution2D(128, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)

# Layer 4
    conv4 = Convolution2D(256, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(pool3)
    conv4 = Dropout(0.2)(conv4)
    conv4 = Convolution2D(256, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)


```

```
# Layer 5
    conv5 = Convolution2D(512, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(pool4)
    conv5 = Dropout(0.2)(conv5)
    conv5 = Convolution2D(512, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(conv5)

# Layer 6
    up6 = merge([UpSampling2D(size=(2, 2))(conv5), conv4], mode='concat', concat_axis=1)
    conv6 = Convolution2D(256, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(up6)
    conv6 = Dropout(0.2)(conv6)
    conv6 = Convolution2D(256, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(conv6)

# Layer 7
    up7 = merge([UpSampling2D(size=(2, 2))(conv6), conv3], mode='concat', concat_axis=1)
    conv7 = Convolution2D(128, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(up7)
    conv7 = Dropout(0.2)(conv7)
    conv7 = Convolution2D(128, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(conv7)

# Layer 8
    up8 = merge([UpSampling2D(size=(2, 2))(conv7), conv2], mode='concat', concat_axis=1)
    conv8 = Convolution2D(64, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(up8)
    conv8 = Dropout(0.2)(conv8)
    conv8 = Convolution2D(64, 3, 3, activation='relu', border_mode='same')(conv8)

# Layer 9
    up9 = merge([UpSampling2D(size=(2, 2))(conv8), conv1], mode='concat', concat_axis=1)
    conv9 = Convolution2D(32, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(up9)
    conv9 = Dropout(0.2)(conv9)
    conv9 = Convolution2D(32, 3, 3, activation='relu', border_mode='same',
        kernel_initializer='he_normal')(conv9)

# Layer 10
    conv10 = Convolution2D(1, 1, 1, activation='sigmoid',
        kernel_initializer='he_normal')(conv9)

    model = Model(input=inputs, output=conv10)
    model.compile(optimizer=Adam(lr=0.001), loss=dice_coef_loss, metrics=[dice_coef])
    return model
```

A.2 Make Cancer Mask Algorithm

Listing A.2: Make Mask Algorithm by Mulholland et al.

```
def make_mask(center,diam,z,width,height,spacing,origin):
    '''
Center : centers of circles px -- list of coordinates x,y,z
diam : diameters of circles px -- diameter
widthXheight : pixel dim of image
spacing = mm/px conversion rate np array x,y,z
origin = x,y,z mm np.array
z = z position of slice in world coordinates mm
    '''
    mask = np.zeros([height,width]) # 0's everywhere except nodule swapping x,y to match im
    #convert to nodule space from world coordinates

    # Defining the voxel range in which the nodule falls
    v_center = (center-origin)/spacing
    v_diam = int(diam/spacing[0]+5)
    v_xmin = np.max([0,int(v_center[0]-v_diam)-5])
    v_xmax = np.min([width-1,int(v_center[0]+v_diam)+5])
    v_ymin = np.max([0,int(v_center[1]-v_diam)-5])
    v_ymax = np.min([height-1,int(v_center[1]+v_diam)+5])

    v_xrange = range(v_xmin,v_xmax+1)
    v_yrange = range(v_ymin,v_ymax+1)

    # Convert back to world coordinates for distance calculation
    x_data = [x*spacing[0]+origin[0] for x in range(width)]
    y_data = [x*spacing[1]+origin[1] for x in range(height)]

    # Fill in 1 within sphere around nodule
    for v_x in v_xrange:
        for v_y in v_yrange:
            p_x = spacing[0]*v_x + origin[0]
            p_y = spacing[1]*v_y + origin[1]
            if np.linalg.norm(center-np.array([p_x,p_y,z]))<=diam:
                mask[int((p_y-origin[1])/spacing[1]),int((p_x-origin[0])/spacing[0])] = 1.0
    return(mask)
```

A.3 Lung Segmentation Algorithm

Listing A.3: Lung Segmentation Algorithm by Mulholland et al.

```
for img_file in file_list:
    # I ran into an error when using Kmean on np.float16, so I'm using np.float64 here
    imgs_to_process = np.load(img_file).astype(np.float64)
    print ("on image", img_file)
    for i in range(len(imgs_to_process)):
        img = imgs_to_process[i]
        #Standardize the pixel values
        mean = np.mean(img)
        std = np.std(img)
        img = img-mean
        img = img/std
        # Find the average pixel value near the lungs
        # to renormalize washed out images
        middle = img[100:400,100:400]
        mean = np.mean(middle)
        max = np.max(img)
        min = np.min(img)
        # To improve threshold finding, I'm moving the
        # underflow and overflow on the pixel spectrum
        img[img==max]=mean
        img[img==min]=mean
        #
        # Using Kmeans to separate foreground (radio-opaque tissue)
        # and background (radio transparent tissue ie lungs)
        # Doing this only on the center of the image to avoid
        # the non-tissue parts of the image as much as possible
        #
        kmeans = KMeans(n_clusters=2).fit(np.reshape(middle,[np.prod(middle.shape),1]))
        centers = sorted(kmeans.cluster_centers_.flatten())
        threshold = np.mean(centers)
        thresh_img = np.where(img<threshold,1.0,0.0)  # threshold the image
        #
        # I found an initial erosion helful for removing graininess from some of the region
        # and then large dialation is used to make the lung region
        # engulf the vessels and incursions into the lung cavity by
        # radio opaque tissue
        #
        eroded = morphology.erosion(thresh_img,np.ones([4,4]))
        dilation = morphology.dilation(eroded,np.ones([10,10]))
        #
        #  Label each region and obtain the region properties
        #  The background region is removed by removing regions
        #  with a bbox that is to large in either dimnsion
        #  Also, the lungs are generally far away from the top
        #  and bottom of the image, so any regions that are too
        #  close to the top and bottom are removed
        #  This does not produce a perfect segmentation of the lungs
        #  from the image, but it is surprisingly good considering its
        #  simplicity.
        #
        labels = measure.label(dilation)
        label_vals = np.unique(labels)
        regions = measure.regionprops(labels)
        good_labels = []
        for prop in regions:
            B = prop.bbox
            if B[2]-B[0]<475 and B[3]-B[1]<475 and B[0]>40 and B[2]<472:
                good_labels.append(prop.label)
        mask = np.ndarray([512,512],dtype=np.int8)
        mask[:] = 0
        #
        #  The mask here is the mask for the lungs--not the nodes
        #  After just the lungs are left, we do another large dilation
        #  in order to fill in and out the lung mask
        #
        for N in good_labels:
            mask = mask + np.where(labels==N,1,0)
        mask = morphology.dilation(mask,np.ones([10,10])) # one last dilation
```

A.4 Sorensen Dice Coefficient

Sorensen Dice Coefficient is a statistical evaluation metric for calculating how similar 2 samples are [15]. It is commonly used in image segmentation to compare the output of a mask is to the reference image. Figure A.1 shows an example of how the Sorensen Dice Coefficient is used in image segmentation

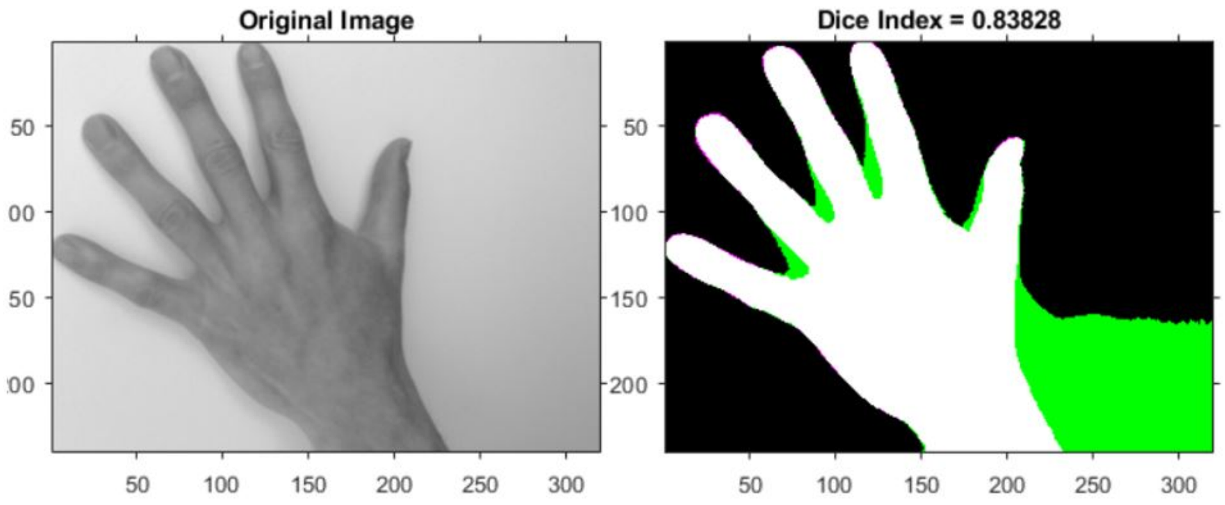


Figure A.1: Sorensen Dice Coefficient[15]

A.5 Image Masks

Masking is a technique used in Image Segmentation. J. Courtney 2017[21] mentions that it's widely used to eliminate areas that are not of interest by retaining the region of interest and ignoring everything else. Figure A.2 displays an example of a zebra mask taken from the reference image.

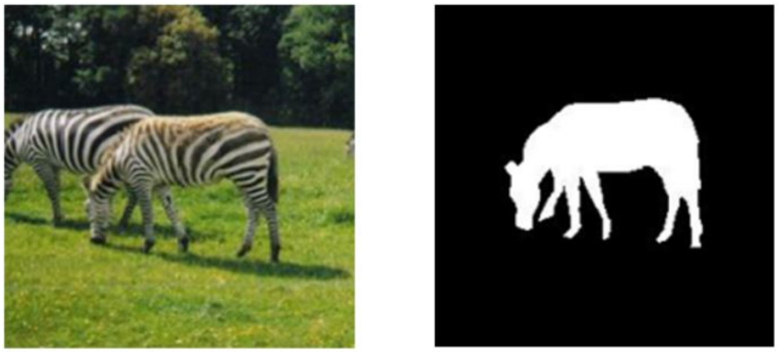


Figure A.2: Binary Mask of a Zebra[2]

A.6 Image Contours

Contours are lines joining continuous points having the same color or intensity. This technique is widely used in object detection, object recognition and analysis of shapes for image processing. Figure A.3 shows an image of a contour found on a mask (left) and applied on the reference image(right).

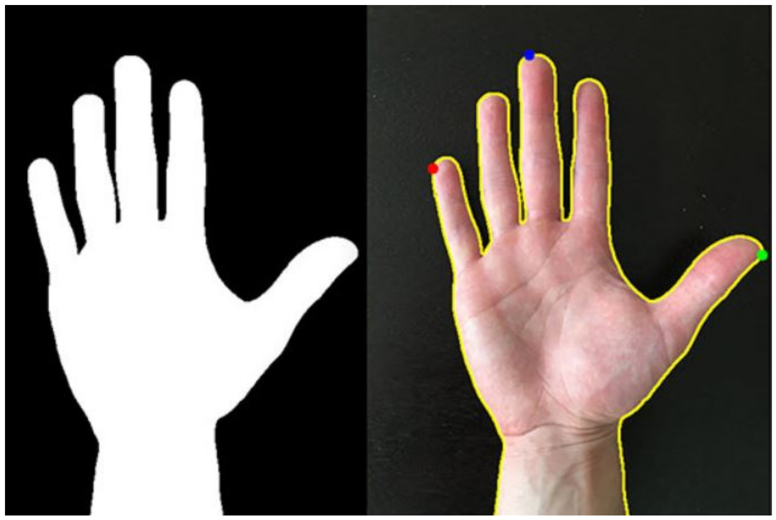
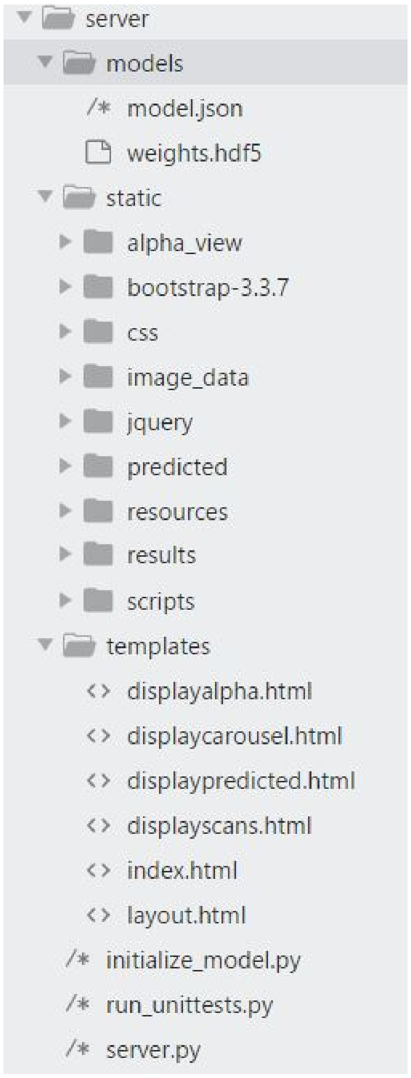


Figure A.3: Contour Feature Example(Image Source: pyimagesearch.com)

A.7 Flask Source Code Layout



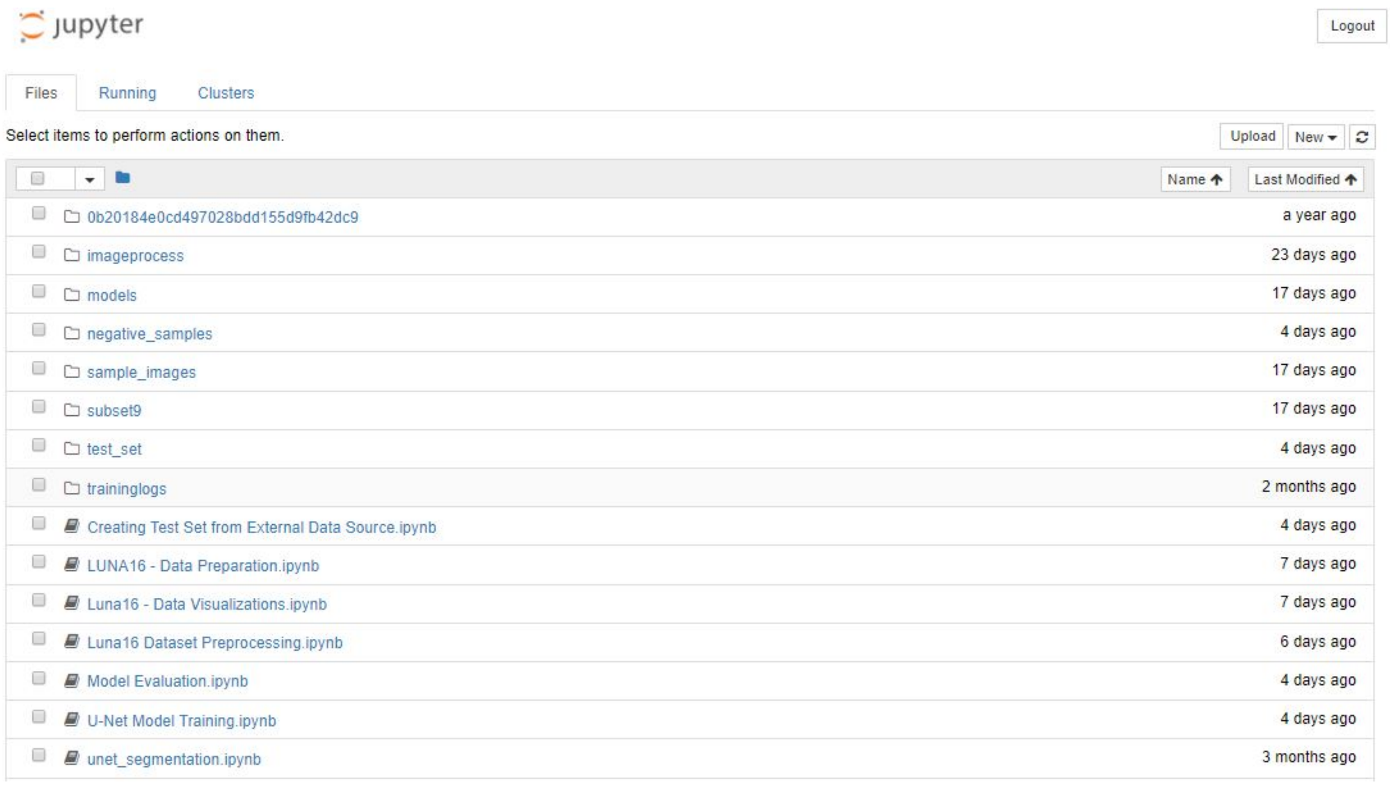
```
server
  models
    model.json
    weights.hdf5
  static
    alpha_view
    bootstrap-3.3.7
    css
    image_data
    jquery
    predicted
    resources
    results
    scripts
  templates
    displayalpha.html
    displaycarousel.html
    displaypredicted.html
    displayscans.html
    index.html
    layout.html
  initialize_model.py
  run_unittests.py
  server.py
```

Figure A.4: Flask App Source Code Layout

Figure A.4 shows the source code layout for the application. The layout follows a standard Flask setup where the HTML templates are placed in templates folder and static files such as libraries, images and numpy data are stored there. The deep learning model is stored in the model folder and the python scripts are stored in root of the folder.

The initialize_model script loads in the model files and creates the necessary configurations to perform deep learning predictions on the application. The server script is the main controller that contains the majority of the back-end functionality. Lastly the run_unittests script includes all of the unit tests performed for this project.

A.8 Jupyter Source Code Layout



Select items to perform actions on them.

0b20184e0cd497028bdd155d9fb42dc9	a year ago
imageprocess	23 days ago
models	17 days ago
negative_samples	4 days ago
sample_images	17 days ago
subset9	17 days ago
test_set	4 days ago
traininglogs	2 months ago
Creating Test Set from External Data Source.ipynb	4 days ago
LUNA16 - Data Preparation.ipynb	7 days ago
Luna16 - Data Visualizations.ipynb	7 days ago
Luna16 Dataset Preprocessing.ipynb	6 days ago
Model Evaluation.ipynb	4 days ago
U-Net Model Training.ipynb	4 days ago
unet_segmentation.ipynb	3 months ago

Figure A.5: Jupyter Source Code Layout

Figure A.5 displays the jupyter notebook files for the data mining aspect of the project. There exists a notebook for each phase of the data mining process.

Bibliography

[1] [1412.6980v8] Adam: A Method for Stochastic Optimization, . URL https://arxiv.org/abs/1412.6980v8.

[2] Analyzing The Papers Behind Facebook's Computer Vision Approach Adit Deshpande CS Undergrad at UCLA ('19), . URL https://adeshpande3.github.io/Analyzing-the-Papers-Behind-Facebook%27s-Computer-Vision-Approach/.

[3] Computer Vision - deeplearning.ai, . URL https://www.coursera.org/learn/convolutional-neural-networks/lecture/Ob1nR/computer-vision.

[4] Data Science Bowl 2017 | Kaggle, . URL https://www.kaggle.com/c/data-science-bowl-2017/kernels.

[5] FloydHub - Deep Learning Platform - Cloud GPU, . URL https://www.floydhub.com/.

[6] Hounsfield unit, . URL https://medical-dictionary.thefreedictionary.com/Hounsfield+unit.

[7] Hounsfield Scale LITFL Life in the Fast Lane Medical Blog, . URL https://lifeinthefastlane.com/funtabulously-frivolous-friday-five-164/screen-shot-2016-10-14-at-11-19-30/.

[8] LUNA16 Grand Challenge, . URL https://luna16.grand-challenge.org/.

[9] Lung cancer: Reduce your risk by quitting smoking, . URL http://www.mayoclinic.org/diseases-conditions/lung-cancer/basics/definition/con-20025531.

[10] Manifesto for Agile Software Development, . URL http://agilemanifesto.org/.

[11] Neural Network Foundations, Explained: Activation Function, . URL https://www.kdnuggets.com/2017/09/neural-network-foundations-explained-activation-function.html.

[12] Object Localization and Detection Artificial Inteligence, . URL https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/object_localization_and_detection.html.

[13] Overview of neuron structure and function, . URL https://www.khanacademy.org/science/biology/human-biology/neuron-nervous-system/a/overview-of-neuron-structure-and-function.

[14] Project Jupyter, . URL http://www.jupyter.org.

[15] Srensen-Dice similarity coefficient for image segmentation - MATLAB dice - MathWorks United Kingdom, . URL https://uk.mathworks.com/help/images/ref/dice.html.

[16] TensorFlow, . URL https://www.tensorflow.org/.

[17] Welcome to Python.org, . URL https://www.python.org/.

[18] Welcome | Flask (A Python Microframework), . URL http://flask.pocoo.org/.

[19] What main methodology are you using for your analytics, data mining, or data science projects? Poll, . URL https://www.kdnuggets.com/polls/2014/analytics-data-mining-data-science-methodology.html.

[20] F. Chollet. Keras as a simplified interface to TensorFlow: tutorial. URL https://blog.keras.io/keras-as-a-simplified-interface-to-tensorflow-tutorial.html.

[21] J. Courtney. Segmentation of Images.

[22] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *arXiv:1406.2572 [cs, math, stat]*, June 2014. URL http://arxiv.org/abs/1406.2572. arXiv: 1406.2572.

[23] M. Firmino, A. H. Morais, R. M. Mendoa, M. R. Dantas, H. R. Hekis, and R. Valentim. Computer-aided detection system for lung cancer in computed tomography scans: Review and future prospects. *BioMedical Engineering OnLine*, 13:41, Apr. 2014. ISSN 1475-925X. doi: 10.1186/1475-925X-13-41. URL https://doi.org/10.1186/1475-925X-13-41.

[24] Guido Zuidhof. Full Preprocessing Tutorial. URL https://www.kaggle.com/gzuidhof/full-preprocessing-tutorial.

[25] K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *arXiv:1502.01852 [cs]*, Feb. 2015. URL http://arxiv.org/abs/1502.01852. arXiv: 1502.01852.

[26] G. Huang, G.-B. Huang, S. Song, and K. You. Trends in extreme learning machines: A review. *Neural Networks*, 61(Supplement C):32 – 48, 2015. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2014.10.001. URL http://www.sciencedirect.com/science/article/pii/S0893608014002214.

[27] L. Jacobson. Introduction to Artificial Neural Networks - Part 1. URL http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7.

[28] A. Jemal, R. Siegel, and K. D. Miller. American Cancer Society |. URL http://cancerstatisticscenter.cancer.org/.

[29] Jonathan Mulholland and Aaron Sander. Data Science Bowl 2017. URL https://www.kaggle.com/c/data-science-bowl-2017.

[30] E. A. Krupinski, K. S. Berbaum, R. T. Caldwell, K. M. Schartz, and J. Kim. Long Radiology Workdays Reduce Detection and Accommodation Accuracy. *J Am Coll Radiol*, 7(9):698–704, Sept. 2010. ISSN 1546-1440. doi: 10.1016/j.jacr.2010.03.004. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2935843/.

[31] M. Lustman. Udacity Self-Driving Car Nanodegree: vehicle detection and tracking + lane detection - YouTube. URL https://www.youtube.com/watch?v=sWkt3vRiCZ0.

[32] H. MacMahon, D. P. Naidich, J. M. Goo, K. S. Lee, A. N. C. Leung, J. R. Mayo, A. C. Mehta, Y. Ohno, C. A. Powell, M. Prokop, G. D. Rubin, C. M. Schaefer-Prokop, W. D. Travis, P. E. Van Schil, and A. A. Bankier. Guidelines for Management of Incidental Pulmonary Nodules Detected on CT Images: From the Fleischner Society 2017. *Radiology*, 284(1):228–243, Feb. 2017. ISSN 0033-8419. doi: 10.1148/radiol.2017161659. URL https://pubs.rsna.org/doi/full/10.1148/radiol.2017161659.

[33] S. Mirsadraee, D. Oswal, Y. Alizadeh, A. Caulo, and E. J. van Beek. The 7th lung cancer TNM classification and staging system: Review of the changes and implications. *World J Radiol*, 4(4):128–134, Apr. 2012. ISSN 1949-8470. doi: 10.4329/wjr.v4.i4.128. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3351680/.

[34] A. Ng. Why is Deep Learning taking off? - deeplearning.ai. URL https://www.coursera.org/learn/neural-networks-deep-learning/lecture/praGm/why-is-deep-learning-taking-off.

[35] R. Pieters. Python for Image Understanding: Deep Learning with Convolutional Neur. URL https://www.slideshare.net/roelofp/python-for-image-understanding-deep-learning-with-convolutional-neural-nets.

[36] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv:1505.04597 [cs]*, May 2015. URL http://arxiv.org/abs/1505.04597. arXiv: 1505.04597.

[37] S. Ruder. An overview of gradient descent optimization algorithms. Sept. 2016. URL https://arxiv.org/abs/1609.04747.

[38] A. A. A. Setio, A. Traverso, T. de Bel, M. S. Berens, C. v. d. Bogaard, P. Cerello, H. Chen, Q. Dou, M. E. Fantacci, B. Geurts, R. v. d. Gugten, P. A. Heng, B. Jansen, M. M. de Kaste, V. Kotov, J. Y.-H. Lin, J. T. Manders, A. Sora-Mengana, J. C. Garca-Naranjo, E. Papavasileiou, M. Prokop, M. Saletta, C. M. Schaefer-Prokop, E. T. Scholten, L. Scholten, M. M. Snoeren, E. L. Torres, J. Vandemeulebroucke, N. Walasek, G. C. Zuidhof, B. v. Ginneken, and C. Jacobs. 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, Dec. 2017. ISSN 13618415. doi: 10.1016/j.media.2017.06.015. URL http://linkinghub.elsevier.com/retrieve/pii/S1361841517301020.

[39] C. Shearer. The CRISP-DM model: the new blueprint for data mining. *J Data Warehouse*, 5:13–22, Jan. 2000.

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[41] vdumoulin. conv_arithmetic: A technical report on convolution arithmetic in the context of deep learning, Nov. 2017. URL https://github.com/vdumoulin/conv_arithmetic. original-date: 2016-02-24T15:18:33Z.

[42] Xavier Glorot, Yoshua Bengio. glorot10a.pdf. URL http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf?hc_location=ufi.