Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

Universal Lesion Detector: Deep Learning for Analysing Medical Scans

Author:
Martin Zlocha

Supervisor:
Dr Ben Glocker

Second Marker:
Dr Jonathan Passerat-Palmbach

June 17, 2019

Abstract

Accurate, automated lesion detection is an important yet challenging task due to the large variation of lesion types, sizes, locations and appearances. Most of the recent work only focuses on lesion detection in a constrained setting - detecting lesions in a specific organ. We tackle the general problem of detecting lesions across the whole body and propose approaches which are not limited to a particular dataset. This is done by redesigning RetinaNet to be more applicable to medical imaging, using a general approach for optimising anchor configurations and by generating additional weak labels from the provided ground truth. We evaluate our approach on two different datasets - a large public Computed Tomography (CT) dataset called DeepLesion, consisting of 32,735 lesions, and a much smaller whole-body Magnetic Resonance Imaging (MRI) dataset made up of only 213 scans. We show that our approach achieves state-of-the-art results, significantly outperforming the best reported methods by over 5% on the DeepLesion benchmark, while being generalisable enough to achieve excellent results on the much smaller whole-body MRI dataset.

Acknowledgements

I would like to thank my supervisor Dr Ben Glocker, for his consistent guidance and support. I would also like to thank my collaborator, Dr Qi Dou, for her invaluable advice.

Further, I would like to thank Jan Matas, Inara Ramji and Jayati Sarkar for proof-reading key chapters of the report.

Lastly, I would like to thank my friends and family for giving me advice throughout the project and always supporting me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cancer is one of the most common causes of death in the UK [15] and deaths from neoplasms (cancers and other non-cancerous tissue growths) are one of the leading causes of avoidable deaths in the UK [1]. However, the high workload and fatigue of clinicians means that they might miss a significant number of lesions. 30% of the false-negative errors are considered as scanning errors where the doctor missed the lesion even though it was visible in the scan [24]. To make the task harder, clinicians routinely only spend 4.6 minutes [12] before they have to make a diagnosis. The lack of time, pressure, limited resources and complexity of making a diagnosis introduces errors which may have fatal consequences.

Computer-aided detection/diagnosis (CADe/CADx) systems can be used to reduce the number of human errors during diagnosis. Research shows that these systems detected up to 70% of lung cancers that were not detected by the radiologist but failed to detect about 20% of the lung cancers when they were identified by the radiologist, which suggests that CAD may be useful in the role of second reader [29]. Automated detection systems could also be used to passively look for lesions in patients who initially received a Computed Tomography (CT) or Magnetic Resonance Imaging (MRI) scan for a different reason. Up to 39% of patients with lung cancer are asymptomatic at the time of diagnosis [18], however, it is too costly to look for lesions in every scan if the patient does not display the symptoms.

The main drawback of these systems is that they aren't general enough to be applied to lesions in CT and MRI scans from everywhere in the body, which limits their applicability in industry.

1.1 Objectives

The objective of this project is to develop a deep learning model for detection of lesions. Most of the current research focuses on detecting lesions in a single organ or of a single lesion type. This project aims to produce a universal architecture which can be applied to both CT and MRI scans from anywhere in the body and will reliably be able to detect the lesions.

We will use two datasets. Firstly, the DeepLesion [53] dataset will be used, as it consists of 32,735 lesions in 32,120 CT slices from 10,594 studies of 4,427 unique patients. Most other datasets only focus on collecting only one type of lesion or on only one organ, however, DeepLesion contains lesions from 8 different locations with significantly varying sizes and appearances. To verify that our approach is general enough and can be applied to datasets with other modalities, we will use a whole-body MRI dataset which consists of 213 MRI scans and contains both colorectal and lung lesions. This tests the generalisability of our model, as well as the ability to learn on smaller sample size.

After this goal has been completed, we have also added a stretch goal to make the project demonstrable to the wider public. This makes our research more approachable, tangible and easier to understand.

To achieve this goal, the task is split into several sub-tasks:

1. Consider several standard object detectors and decide which is the most suitable for this task.
2. Preprocess the data so that it can be fed into the object detector and achieves good results.
3. Modify the model so that it achieves state-of-the-art results on the DeepLesion dataset.
4. Verify that our model can generalise and achieves good results on the whole-body MRI dataset.
5. Create a website for non-technical people which gamifies lesion detection and demonstrates the accuracy of our model.

1.2 Challenges

During the project, we found the following to be the biggest challenges:

1. **New datasets** - The DeepLesion dataset has been released recently, and there is a lack of research based on it so far. It is also very diverse with many different lesion types. These two factors make it harder to achieve good results. The whole-body MRI dataset is only used in-house and is much smaller, so we were not sure if it will be feasible to train a model on it without overfitting.
2. **Lack of domain knowledge** - Medical imaging and lesion detection in CT and MRI scans requires a lot of domain knowledge when done manually. Even though this domain knowledge is not required when training object detectors, this deficiency may be a drawback compared to other research and might hinder the performance of our model.
3. **Training times** - The size of the DeepLesion dataset and complexity of the models restrict the iteration times. Currently, it takes a day to train a model which uses 2D convolutions from start to finish, and this may increase to the order of days or weeks if we use 3D convolutions. This requires good time management and planning to avoid wasting time while waiting for training to complete.
4. **Access to sufficient hardware** - This project involves training deep neural networks which require high memory machines to train. As many students are also doing machine learning based projects, the number of available resources provided by the department was limited which, required us to make compromises and limited all of the experimentation which we could do.

1.3 Contribution

1. **Optimize the feature pyramid scheme and anchor configuration** - We have modified the architecture of the network to be more suitable for medical imaging and demonstrated how a differential evolution search algorithm could be used to optimise the anchor configuration, which greatly improves the performance without needing extensive domain knowledge. This approach to optimising the anchor configuration can be applied to any dataset, which makes it suitable for a universal lesion detector and wider use.

2. **Generate Dense Mask from Weak RECIST Labels** - We utilise the response evaluation criteria in solid lesions (RECIST) [8] diameters, which are widely used for annotations of lesions and generate high-quality, dense masks. This is done using the GrabCut [42] algorithm which reduces the workload on doctors as pixel-wise annotations are tedious and expensive to obtain. Using these masks as an additional supervision signal shows great benefit for training the detector.

3. **Employ multiple strategies for boosting the detection performance** - We use the lesion mask predictions and compare them to the bounding box predictions to evaluate the coherence and adjust the prediction score. We also integrate an attention mechanism into our feature pyramids, which further improves the performance.

4. **Demonstrate how the architecture can be applied to different datasets** - A whole-body MRI dataset is utilised to demonstrate that the modified architecture is applicable to other datasets without major modifications. We further demonstrate the improvement in results when transfer learning is used to initialise the model weights when training on the whole-body MRI dataset.

5. **Website** - We built a website which demonstrates our work in an accessible and gamified format. The aim was to make it engaging to non-technical people while preserving the educational value about the difficulties of detecting lesions.[1]

1.4 Publication

The work which focuses on the DeepLesion dataset was submitted and accepted to Medical Image Computing and Computer Assisted Intervention (MICCAI[2]) 2019 conference under the title "Improving RetinaNet for CT Lesion Detection with Dense Masks from Weak RECIST Labels" [54]. After the submission to MICCAI 2019, our work was focused on demonstrating how the architecture can be applied to different datasets such as the whole-body MRI dataset.

1.5 Report Layout

Chapter 2 focuses on research in object detection which is commonly used in industry and also briefly introduces research in object detection specific to medical imaging. A brief comparison of different approaches and motivation for the usage of RetinaNet can also be found in this chapter.

The work done is split into four chapters. Chapter 3 focuses on the work required to redesign RetinaNet to be more applicable to medical imaging. Chapter 4 first introduces the DeepLesion dataset and evaluates the performance showing that our approach achieves state-of-the-art results, significantly outperforming the best reported methods by over 5%. Afterwards, chapter 5 introduces a whole-body MRI dataset in order to demonstrate how our design can be applied to other datasets and achieves excellent results without significant modifications. Chapter 6 briefly introduces a website which was made to demonstrate our work and gamify the task of detecting lesions in CT scans while educating people.

Chapter 7 concludes this report by summarising the work which was done and suggests several areas of future work.

[1] `https://www.doc.ic.ac.uk/~mz4315/deeplesion/`
[2] `https://www.miccai2019.org/`

Chapter 2

Background

Recent advances in object detection neural networks and availability of new medical imaging datasets enabled automated segmentation of CT and MRI scans both for commercial and research purposes.

This chapter surveys current state-of-art approaches to object detection and localisation both in the medical and non-medical field, how these approaches have evolved.

2.1 Approaches

In the past this was done using traditional machine learning approaches where it was first necessary to define features and then to perform classification using Haar-Feature Classifiers [50] or Histogram of oriented gradients (HOG) [6].

Recently deep learning techniques have gained in popularity because they are able to do end-to-end object detection without having to define custom features [9]. These use convolutional neural networks where each neuron processes data only for its region of space. Examples of this approach include Region-based Convolutional Network (R-CNN [13], Fast R-CNN [14], Faster R-CNN [40], R-FCN [5]), You Only Look Once (YOLO) [39], Single Shot MultiBox Detector (SSD) [32] or Focal Loss for Dense Object Detection (RetinaNet) [30].

These object detection networks usually also work with multiple different backbones such as VGG [43], Inception/GoogLeNet [46] and ResNet [17].

This section will focus on the deep learning techniques as those have shown to achieve better results and are more commonly used in medical imaging.

2.2 Object Detection Networks

When deciding on which object detection network should be used, the key metric was mAP (Mean Average Precision), which is the most used metric to compare object detection networks. Another metric which is widely used is the inference time. For this project, this is not as important because the network would never be used in real-time systems where high throughput is required.

2.2.1 Region-based Convolutional Network

R-CNN

Image classification has been most commonly approached by using HOG [6] or CNN's. Unlike image classification, detection requires finding (many) objects within an image. One approach is to treat this as a regression problem which works well for finding the

bounding box of a single object but does not scale to an arbitrary number of objects. An alternative approach is to use a sliding window approach and feed that region into a CNN. The problem of this approach is that it requires all objects to share a common size and aspect ratio.

To tackle these problems, R-CNN [13] is split into three modules:

1. **Region proposals** - Instead of using regions of fixed size generated by sliding a window across the whole image, selective search [49] is used which reduces the number of regions that have to be considered.

2. **Feature extraction** - This step uses a CNN with an S*S RBG image input which produces a fixed-size feature vector which can be used by the next module. Before doing this the region has to be re-sized so that it is compatible with the CNN, this is done by warping the image.

3. **Class-specific linear SVM** - This module scores the feature vectors for each of the regions. After all of the regions are scored, a region is rejected if it has an IoU (intersection over union) overlap with a higher scoring region larger than a learned threshold.

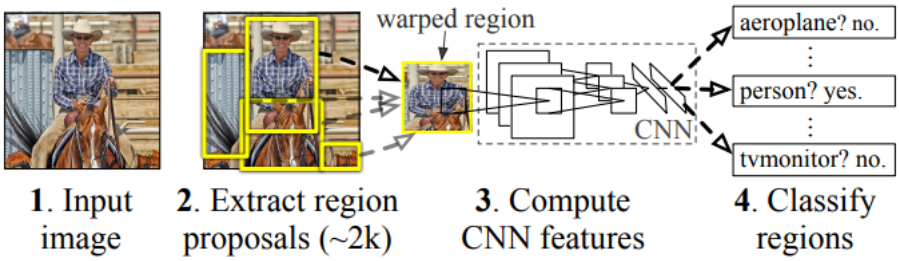


Figure 2.1: Structure of R-CNN [13].

This design was further extended by adding a bounding-box regression stage. After scoring each output from the CNN with a class-specific linear SVM, a new bounding box is predicted using the bounding-box regressor to achieve better IoU. This new bounding box is scored again to obtain the final score. This process can be iterated; however, the results didn't improve noticeably.

Problems with R-CNN

- This process is very slow because you have to classify 2000 regions per image. This is very inefficient as many of the regions overlap so a lot of work is duplicated.

- Selective search is a fixed algorithm that cannot learn, which can lead to bad proposals.

- Training is a multi-stage process because first, the CNN has to be trained, then the SVM has to be fit to the CNN features and in the last stage, the bounding-box regressor is trained.

Fast R-CNN

Fast R-CNN [14] builds on the work of the original R-CNN and resolves the problems of very slow training and testing times and allows training in a single stage.

In Fast R-CNN, the network takes the whole image as an input, applies several convolutional and max-pooling layers and produces a feature map. For each region proposal (which can be generated using selective search), we need to extract a fixed size feature vector, to do this a Region of Interest (RoI) pooling layer is used. Region of Interest pooling uses max-pooling layer to produce a HxW feature (e.g. 7×7).

Each feature map is fed into a sequence of fully connected (dense) layers that finally split into two output layers.

- Layer that produces softmax probability estimates for all of the K object classes plus a "background" class.
- Layer that generates four numbers for each of the K object classes. These four numbers represent the refined bounding-box position for each of the classes.

Each of the parts of Fast R-CNN can be seen in figure 2.2.

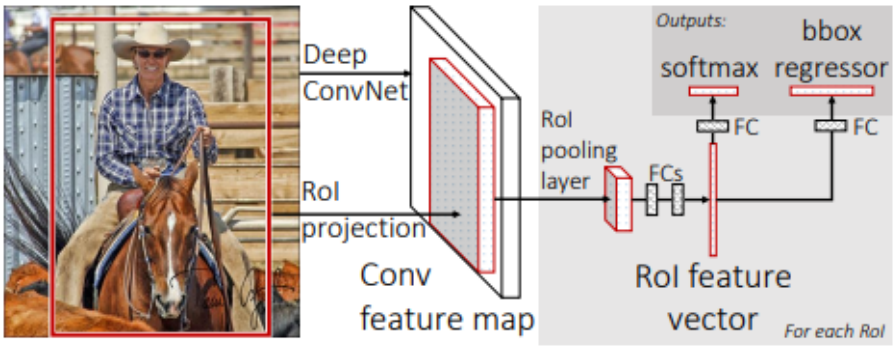


Figure 2.2: Structure of Fast R-CNN [14].

These changes result in training times 9x faster than R-CNN and test times 213x faster while achieving higher mAP. However, not all problems are eliminated such as having to use a fixed algorithm to generate region proposals which cannot learn.

Faster R-CNN

Fast R-CNN has been able to achieve very fast inference times, but the region proposals have become a bottleneck consuming as much running time as the detection network. To tackle this problem, Faster R-CNN is made up of two modules:

1. **Region Proposal Network (RPN)** - Takes an image (of any size) as input and produces a feature map similarly to Fast R-CNN. A small network slides across the feature map. At each of the sliding-window locations, multiple region proposals are generated with a score which defines the likelihood of the region being a foreground vs a background object. These regions are called anchors; this will be explained in a section below.
2. **Fast R-CNN** - This module takes as input the different region proposals and the feature map from the RPN and applies the Region of Interest (RoI) pooling layer and all subsequent layers as in Fast R-CNN. The convolutional and max-pooling layers are shared between the RPN and the Fast R-CNN.

These two modules of the Faster R-CNN can be seen in figure 2.3. This shows the shared convolutional layers and the produced feature map, the RPN and the Fast R-CNN, which works on the two inputs.

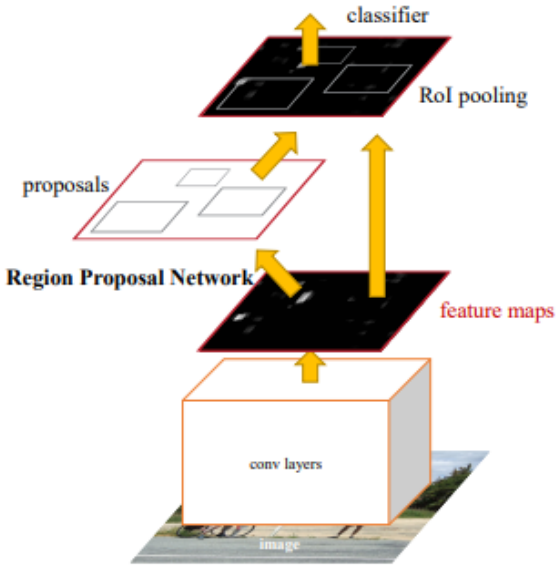


Figure 2.3: Structure of Faster R-CNN [40].

Anchors

At each sliding-window location, multiple region proposals are generated. The regression layer defines the bounding box, and the classification layer scores the probability of the bounding box being foreground vs background. All of the bounding boxes are defined relative to the reference sliding window, which we call anchors. Each anchors position is defined to be at the centre of the sliding window and its dimensions. This can be seen in figure 2.4.

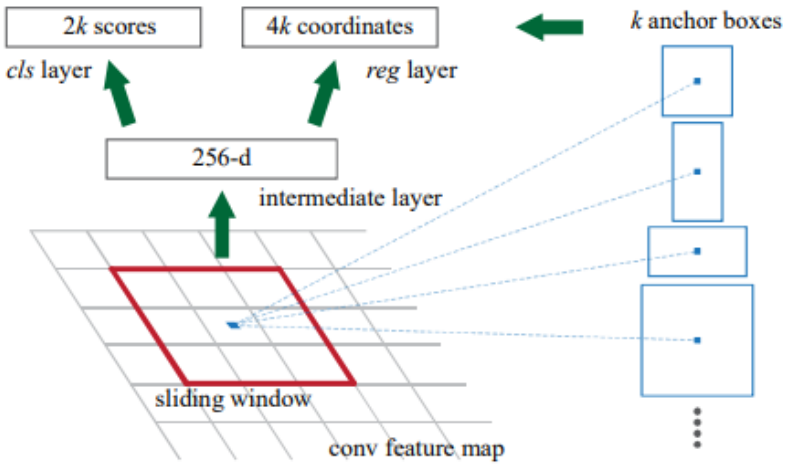


Figure 2.4: Anchor generation in Faster R-CNN [40].

R-FCN (Region-based Fully Convolutional Networks)

The drawback of Fast/Faster R-CNN [14, 40] is that it applies a costly per-region subnetwork which produces the probability estimates and refines the bounding-box. In contrast, R-FCN [5] is fully convolutional with almost all of the computation done on the entire image only once. This is done by generating position-sensitive score maps for each category which address the dilemma between translation-invariance in image classification and translation-variance in object detection. These score maps can be seen in figure 2.5

as layers of different colours (these colours represent all of the categories which are being classified).

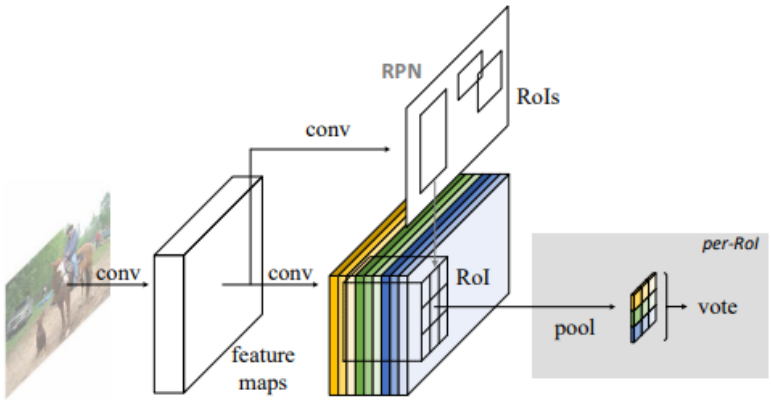


Figure 2.5: Structure of R-FCN [5].

A Region Proposal Network (RPN) is used to generate candidate regions, this is also a fully convolutional network which is shared with the R-FCN.

R-FCN ends with a position-sensitive RoI pooling layer. This layer aggregates the outputs of the position-sensitive score maps and generates scores for each class in the RoI. However, this part uses no convolutional of dense layers. Instead, it averages the values from the score maps.

2.2.2 You Only Look Once

Compared to all of the other object detection methods, YOLO [39] focuses on simplicity and inference time instead of accuracy. It uses a one-step convolutional network which frames object detection as a regression problem. In a single step, it predicts the bounding boxes and the class probabilities. Compared to other methods it makes more localisation errors but its less likely to predict false positives on background, this is because other methods use sliding window or region-proposal based techniques which lack the spatial context during classification.

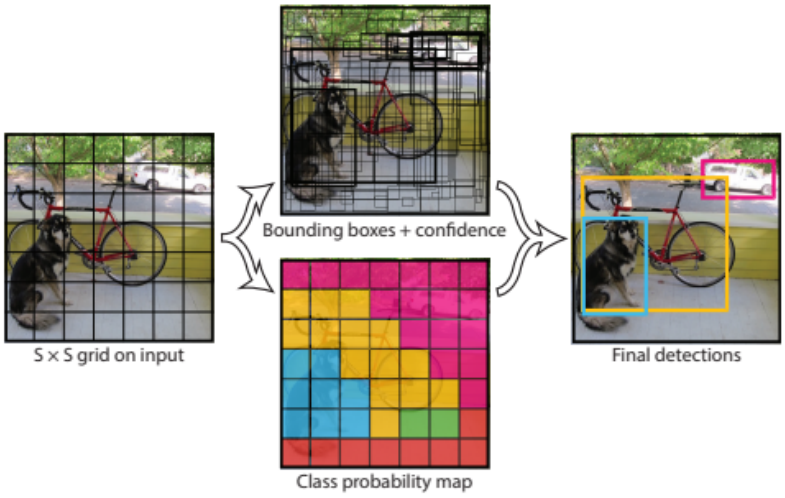


Figure 2.6: Structure of YOLO [39].

However, YOLO is inferior compared to other state-of-art object detection methods inaccuracy. This is because YOLO divides the input image into an $S \times S$ grid. If the

centre of an object falls into a grid cell, that grid cell is responsible for detecting that object. This means that if there are multiple objects within a grid cell, YOLO might not be able to detect all of them. This can be seen in figure 2.6, which shows that each grid square is assigned a single class.

2.2.3 Focal Loss for Dense Object Detection

The highest accuracy object detectors which are most commonly used are based on a two-stage approach of R-CNN [13]. In this case, the classifier is applied to a sparse set of candidate object locations and most commonly does not take into account the spatial context which is included in one-stage object detectors such as YOLO [39]. However, one-stage detectors are applied over a regular, dense sampling of possible locations. This can result in a faster and simpler approach with less false positives, but up till now, these detectors achieved lower accuracy than their two-stage counterpart.

The cause for the inferior accuracy of one-stage detectors is the extreme foreground vs background class imbalance encountered during training. To tackle this problem standard cross-entropy is replaced by Focal Loss, which down-weights the loss assigned to examples with high predicted probability. This prevents a large number of easy negatives from overwhelming the detector during training. Focal Loss adds a $(1 - p_t)^\gamma$ term to cross-entropy where γ is a parameter. The difference between cross-entropy and Focal Loss can be seen in figure 2.7.

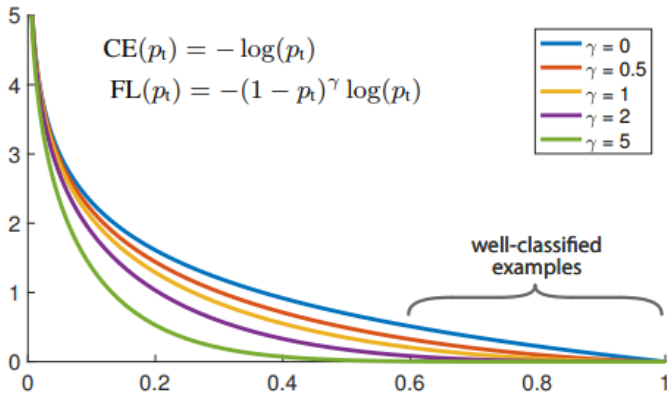


Figure 2.7: Loss calculated by using Focal Loss vs Cross Entropy [30].

The paper also proposes a new network called RetinaNet. This is a single-stage network made up of a backbone and two task-specific subnetworks:

- **Backbone** - A Feature Pyramid Network (FPN) is used as the backbone. FPN augments the standard backbone with a top-down pathway and lateral connections, so the network constructs a multi-scale feature pyramid from a single resolution input image. This allows for better detection of objects at different scales. By default, RetinaNet uses ResNet as the backbone; however, others can be used. Different backbones will be discussed in the next sub-section.
- **Subnetworks** - Similar to Faster R-CNN [40], in RetinaNet there are two subnetworks, one for classifying the region and the second one is used to refine the bounding boxes. Both of these subnetworks have the same design (4 hidden convolutional layers, only the output layer is different) and are applied to the whole feature map produced by the FPN which helps to preserve the spatial context.

This can be seen in the figure 2.8 where (a) and (b) is the backbone with the FPN and (c) and (d) are the two subnetworks used for classification and regression.

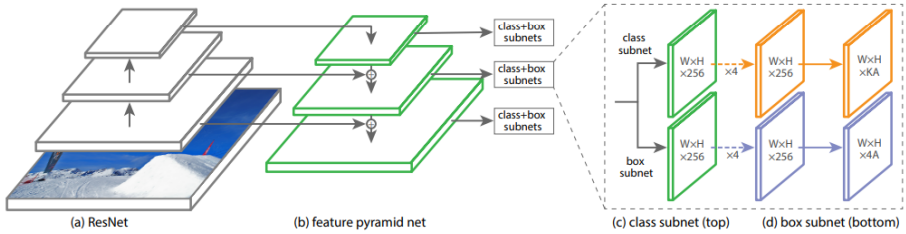


Figure 2.8: Structure of RetinaNet [30].

2.2.4 Comparison

The three main object detectors which were considered for this project were Faster R-CNN, YOLO and RetinaNet. Although YOLO was quickly eliminated as it focuses on inference time instead of accuracy.

The Focal Loss for Dense Object Detection paper contained a very good chart which compares the accuracy of different object detectors, including Faster R-CNN with different backbones. This can be seen in figure 2.9. Based on this RetinaNet was selected as the object detector, which will be used as a basis for the project.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [16]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [20]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [34]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [32]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [27]	DarkNet-19 [27]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [22, 9]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet (ours)	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2

Figure 2.9: Comparison of object detectors [30].

2.3 Backbones

Most of the object detection networks mentioned before rely on a very deep network which generates a feature map that can then be used for region classification and regression. Originally the networks mentioned below have been used for classification of ImageNet images, but if the last fully connected (dense) layers are removed, they can be reused for object detection.

2.3.1 VGG

This is one of the first very deep convolutional models which are still used today and achieve very good results. VGG [43] only uses convolutional and max-pooling layers followed by multiple fully connected (dense) layers; however, these are not used for object detection.

The biggest difference between VGG and previous networks is that it only uses more 3x3 convolutional layers instead of having larger filters (e.g. 7×7). Having three convolutional layers instead of a single 7×7 captures the same amount of area and makes the decision function more discriminative. This also reduces the number of parameters required. Assuming that three 3x3 convolutional layers are used instead of a single 7×7

convolutional layer, we require $3 \times 3^2 \times C^2 = 27 \times C^2$ weights (where C is the number of channels) however for a 7×7 convolutional layer we require $7^2 \times C^2 = 49 \times C^2$ weights.

There are many different configurations in which VGG has been tested and used; however, most commonly the 16 and 19 weight configuration is used. These can be seen in figure 2.10.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 2.10: Structure of VGG [43].

2.3.2 Inception/GoogLeNet

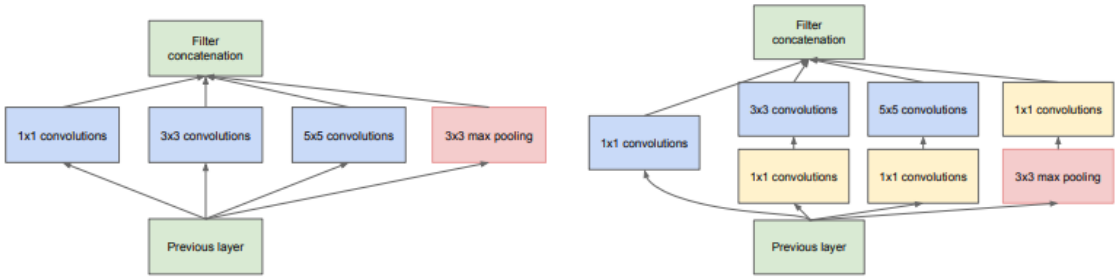
The most common way to increase the accuracy of a model is to either increase its depth (number of layers that it uses) or the width (the number of neurons in each layer). This approach, however, comes with its drawbacks:

- Increasing the size of the network will increase the number of parameters which need to be trained. This requires more training data which might not always be available and make it more prone to overfitting.
- Increasing the number of parameters and the size of the network will increase the training times; however, it might not result in better accuracy.

The way of solving these issues would be to use sparsely connected architectures instead of fully connected ones. However, the current compute resources are not optimised for numerical calculations on sparse matrices.

To solve this, dense layers are used to approximate the sparse local structure in a convolutional network. A combination of 1×1, 3×3 and 5×5 convolutions are used. Having a high number of 5×5 convolutions is too prohibitive, to tackle this, dimension reductions and projections are used which reduce the computational requirements. These blocks of layers are called the Inception modules, which then make up GoogLeNet. The naive and improved Inception modules can be seen in figure 2.4.

There have been many improvements to the design of the Inception block and the GoogLeNet model [47, 45] however those won't be discussed here because they build on the principles mentioned above and aren't used further in the project.



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2.11: Structure of Inception v1 [46].

2.3.3 ResNet

Previous work has shown that deeper networks have been able to achieve better accuracy. This has been shown with the introduction of VGG-16 and VGG-19, which was deeper than the previous networks. However improving the accuracy is not as simple as adding more layers because the network will suffer from the problem of vanishing/exploding gradients, which impedes the convergence from the beginning. Another problem of increasing the depth of a neural network its the degradation, which leads to lower accuracy and higher training error.

To tackle this problem ResNet [17] introduces residual learning blocks as shown in figure 2.12. Lets say that the layer should learn a function $F(x)$, in the residual block we instead learn $H(x)$ where $F(x) = H(x) + x$. The hypothesis is that it is easier to optimize $F(x)$ than the original $H(x)$.

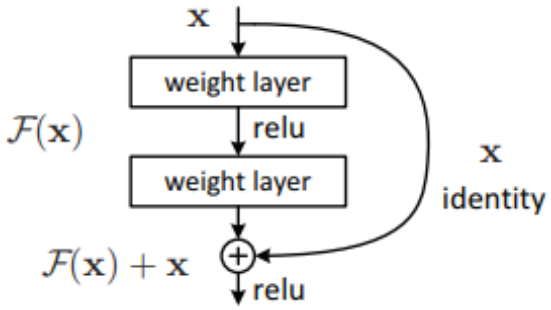


Figure 2.12: Design of the residual learning block in ResNet [17].

This residual learning blocks allow building much deeper networks, the deepest having 152 layers (ResNet-152) while having lower complexity than VGG and achieving better accuracy.

2.3.4 Comparison

When deciding which backbone to use with RetinaNet, VGG and ResNet have been considered. This was because they were both implemented in the RetinaNet library, which was used as a starting point for this project. When testing both implementations, ResNet always performed poorer than VGG. This result is also supported by other research [52, 51] which uses the DeepLesion dataset and achieves better mAP with VGG; however, that research uses R-CNN or other object detectors, so it is not directly comparable.

2.4 U-Net based Object Detectors in Medical Imaging

In many cases, medical imaging requires specialised networks as it comes with a different set of challenges. The first difference is the vastly different input data format. Traditional object detection and segmentation uses RGB images, but in medical imaging, we mostly deal with 3D monochrome volumes and in some cases with multiple volumes with different modalities. The data itself also has different characteristics, in traditional imaging there is a large range of object sizes and the objects are usually locationally invariant and independent. This is not the case in medical imaging as we usually know the voxel sizes, and most objects are usually a similar size across different data points. Another common issue in medical imaging is a lack of large clean datasets. Medical images can only be annotated by experts, which reduces the dataset sizes and in general, they contain more noise than traditional images. Below are a few networks that were built for medical datasets and were used within the project.

2.4.1 U-Net

All of the previously discussed networks rely on large datasets such as ImageNet [7] to achieve good results and usually only produce bounding boxes. U-Net [41] consists of a contracting path to capture context and a symmetric expanding path that enables precise localisation and generates a semantic segmentation mask. The contracting path of the network has a very similar design to VGG however the convolutions are not padded which reduces the output feature map size after each convolution, this can be seen in figure 2.13. To compensate for this, the input image has to be extended by applying "mirroring" at the border.

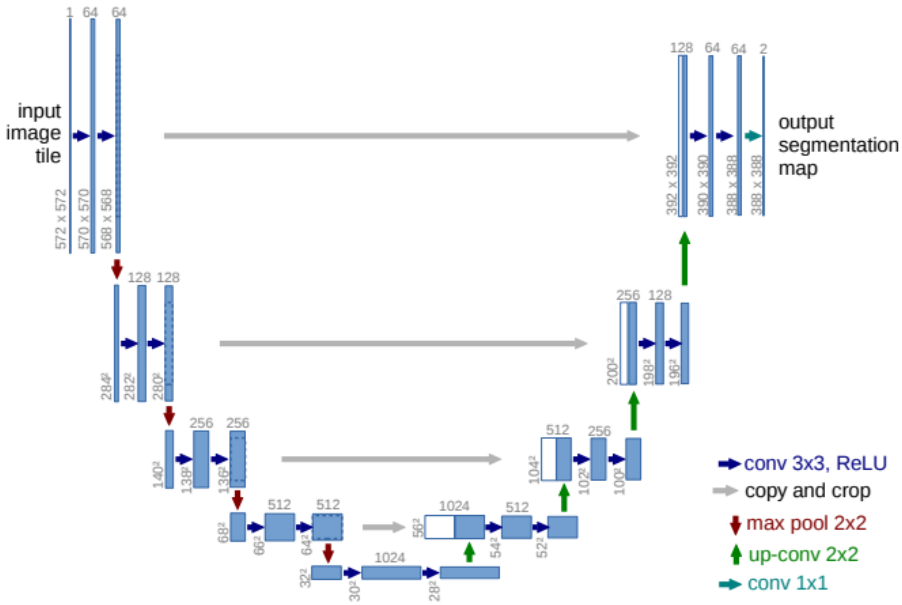


Figure 2.13: Structure of U-NET [41].

The paper also presents a training strategy that relies on a strong use of data augmentation, which allows the network to be trained end-to-end from a few images and outperforms the prior best methods. The key to better results was to apply random elastic deformations using random displacement vectors on a coarse 3 by 3 grid. The displacements are sampled from a Gaussian distribution with 10 pixels standard deviation.

2.4.2 Retina U-Net

In medical imaging, the most common task is semantic segmentation, where each pixel is labelled. This, however, relies on ad-hoc heuristics when attempting to calculate object-level scores. On the other hand, it is possible to generate bounding boxes from the ground truth masks and treat it as an object detection problem. However, in this case, we lose the ability to exploit the full pixel-wise supervision signal. Retina U-Net [21] paper proposes how the RetinaNet [30] architecture can be fused with U-Net [41] which allows the architecture to use both pixel-level and object-level annotations without introducing additional complexity.

To do this, the sub-networks which normally operate on P3-P7 are shifted by one level and use P2-P6. This makes it possible to detect small objects, but it comes at a computational price as there are more anchors produced. Then additional pyramid levels P0 and P1 are added to the top-down pathway and skip connections are added which connect them to the respective levels in the bottom-up path; these changes can be seen in figure 2.14.

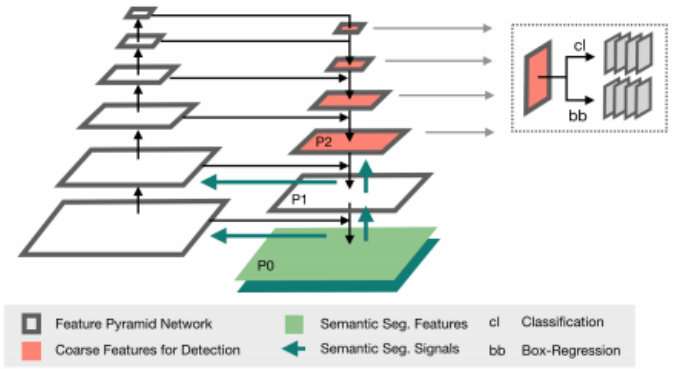


Figure 2.14: Structure of Retina U-Net [21].

The segmentation loss is a combination of pixel-wise cross-entropy and a soft Dice loss which is introduced to stabilise training on highly imbalanced data sets.

The network was tested on a lung nodule dataset and compared with multiple different networks, including RetinaNet. It generally performs better in both 2D and 3D scenarios demonstrating the importance of utilising the additional training signals.

2.4.3 Attention U-Net

Attention has traditionally been used in machine translation or generation where the length of both the input and output is unbounded. Attention allows the model to focus and remember long term dependencies between relevant inputs. However recently it has also been used in imaging to focus the network on a specific part of the image.

Oktay et al. [36] demonstrated how an attention module can be used in a U-Net [41] to automatically learn to focus on target structures, in their case the goal was to focus on the pancreas.

The Attention Gate (AG), which can be seen in figure 2.15, produces attention coefficients (α) and then using element-wise multiplication the attention coefficients are used to prune feature responses in the input feature-maps. In the default case with only one semantic class, a single scalar attention value is computed for each pixel vector x, which makes the attention module very lightweight. In the case of multiple semantic classes, the paper proposes the use of multi-dimensional attention coefficients so that each coefficient can focus on a subset of the target structures. Traditionally for image captioning and classification softmax is used as the activation function to normalise the attention coefficients,

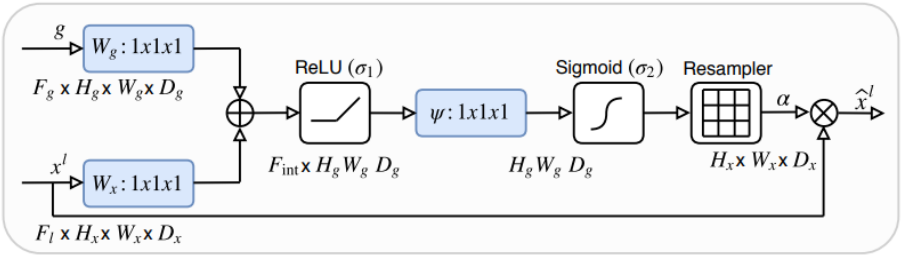


Figure 2.15: Schematic of the Attention Gate (AG), x is the feature response propagated through the skip connection and g is the gating signal, in our case those are the features upsampled from coarser scale [36].

but the paper notes that it produces sparser output so sigmoid activation function is used instead.

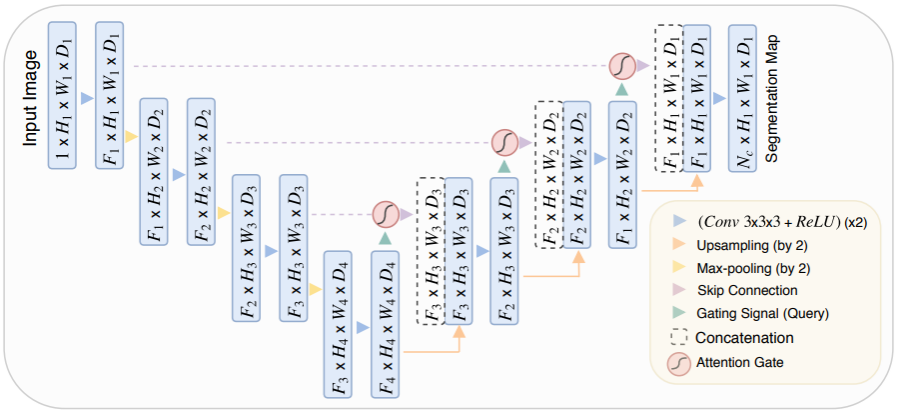


Figure 2.16: Structure of Attention U-Net [36].

The AG described above is incorporated into the standard U-Net model by inserting it before the concatenation operation which merges the features from the skip connection and the up-sampled features from a coarser scale, this can be seen in figure 2.16. The features from a coarser scale are used as the gating signal to prune the feature map propagated by the skip connection.

The Attention U-Net model was evaluated on two different CT abdominal datasets, one with 150 scans and the second with 82 scans. In both cases, small improvement to the results has been achieved, demonstrating that Attention U-Net achieves better results even when trained on smaller datasets compared to the DeepLesion [53] dataset.

Chapter 3

Universal Lesion Detector

This chapter focuses on the work required to redesign RetinaNet to be more applicable to medical imaging. First, we introduce the tools and libraries used throughout the project in section 3.1 and 3.2. Using these tools, in section 3.3, we show how the detection performance can be significantly improved without modifying the architecture of the model just by optimising the anchor configuration. In section 3.4 and 3.5, where we demonstrate how RECIST labels which are provided in many datasets can be used to generate additional weak labels and used during training to improve the performance of the model. Lastly, in section 3.6, where we show how attention can be introduced to RetinaNet to improve the results without the sizeable computational overhead.

3.1 Key Tools and Libraries

Before we can dive into the implementation of our model, we first had to decide on which tools and programming languages will be used. Python was used as the programming language as it is the industry standard for machine learning and offers a massive ecosystem of libraries and a large user community. The next important decision was which machine learning library to use, the decision came down to two main contenders:

- **Keras** [4] is a high-level neural network API capable of running on top of TensorFlow, CNTK and Theano. It has gained popularity as it is easier to use compared to the other libraries and allows fast prototyping. Only using the high-level API can however have its drawbacks and be limiting. To overcome this problem, Keras has an excellent integration with TensorFlow, which allows building of custom layers directly using the TensorFlow API. This can be useful when using non-standard loss functions such as Focal Loss.
- **PyTorch** [37] on the other hand, is a lower-level API focused on direct work with array expressions. It has gained a lot of interest recently becoming a prevalent solution for academic research and applications of machine learning, which require custom layers and expressions. Another advantage of PyTorch compared to TensorFlow is the relative ease of debugging; however, this will change with TensorFlow v2, which should make debugging much easier.

A decision was made to use Keras as fast prototyping and a gradual learning curve is vital during a busy university schedule and slightly longer training duration won't be an issue if time is managed well.

Other libraries which were used extensively include `OpenCV` [2] and `SciPy` [22] which are useful when dealing with images. When dealing with MRI scans, we mainly used `SimpleITK` [33] - an Insight Segmentation and Registration Toolkit (ITK) which facilitates rapid prototyping and simple preprocessing of MRI scans.

3.2 Model Design

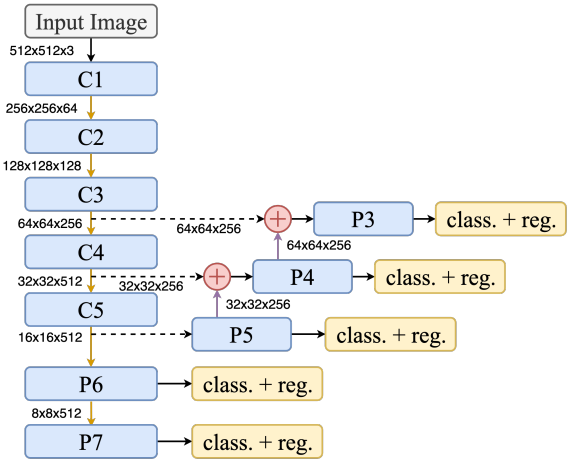


Figure 3.1: Network architecture of RetinaNet without any modifications. C1 to C5 are the 5 convolutional blocks of the backbone. P3 to P7 are the convolutional layers introduced by RetinaNet.

RetinaNet implemented in the `keras-retinanet`[1] library is used as a basis of the project. It has over 2500 stars on GitHub and a very active community of users who answer GitHub Issues and their Slack channel has over 600 users. The library is also very well maintained with continuous integration (CI), good documentation and easy extensibility. The library supports several backbones; the default is ResNet-50 yet ResNet-101 and ResNet-152 [17] have also been implemented. There are also several community-contributed backbones such as VGG [43], MobileNet [19] and DenseNet [20], which make this library an ideal starting point as it allows quick experimentation.

We use VGG-19 as the backbone for RetinaNet as it achieved the best performance on the DeepLesion [53] dataset. The usage of multiple variations of ResNet and DenseNet were also explored, but its performance was worse on the DeepLesion dataset. This result is in line with results reported in [52]. We find that the default implementation works relatively well without any modification, use of focal loss addresses the problem of class imbalance; however, the sensitivity on small lesions was low. This is caused by the fact that RetinaNet requires intersection over union (IoU) of 0.5 and we find the default anchor sizes (32, 64, 128, 256 and 512), aspect ratios (1:2, 1:1 and 2:1), scales ($2^{\frac{0}{3}}$, $2^{\frac{1}{3}}$ and $2^{\frac{2}{3}}$) and strides (8, 16, 32, 64 and 128) are ineffective for small lesions or lesions with large ratios.

3.3 Optimized Anchor Configuration

To achieve good results using RetinaNet on any dataset it is important that the bounding boxes of our objects have similar dimensions as the anchor boxes of the RetinaNet model. In general, the default anchor configuration works well on images; however, tumours are usually much smaller.

Initially, we attempt to manually find a better anchor configuration using trial and error however this is a very slow method because each change is tested by training a new model on the whole dataset which can take one or two days if trained till convergence.

Instead of using trial and error, we frame the anchor optimisation problem as a maximisation of the average IoU between the "best" anchor and the lesion bounding box. For each position, there are $s \times r$ anchors where s is the number of scales which are used and

[1] `https://github.com/fizyr/keras-retinanet`

r is the number of ratios. The "best" anchor is one of the anchors taken from the set of $s \times r$ anchors which has the highest IoU with the lesion bounding box.

The bounding boxes are defined by a set of 4 parameters, these are the anchor scales (s), aspect ratios (r), strides and sizes. The aim is to optimise the anchor scales and aspect ratios as anchor sizes and strides are fixed and defined by the architecture of the model. In the default implementation of RetinaNet only 3 scales and 3 strides are used, but we find that 5 scales achieve a much better IoU as some of the lesions have large ratios. This comes at a computational cost as the number of anchors per anchor centroid is increased from 9 to 15. Another disadvantage of increasing the number of anchors is that the number of positive samples per anchor is decreased so this trade-off might produce better fitting bounding boxes but might result in lower confidence of the model. When the version with 9 anchors was tested compared to the version with 15 anchors we indeed saw an improvement in the results, the main gain came from lesions with large ratios and there was no loss in precision in the other cases. This indicates that the dataset is large enough to support more anchors.

To reduce the search space when optimising the anchor configuration we fix one ratio as $1:1$, and define other ratios as reciprocal pairs (i.e., if one ratio is $1:\gamma$ then another is $\gamma:1$). Thus, we need to optimise only five variables, i.e., two ratio pairs and three scales instead of five ratio pairs and three scales.

To find the best set of variables we employ a differential evolution search algorithm [44], implemented in the `scipy` [22] library, which is a global optimisation algorithm. This algorithm works by randomly initialising a population of candidate solutions, then iteratively improves the population by combining the solutions to produce new ones. This optimisation method is known as metaheuristic as it does not make any assumption about the problem which it is trying to solve and can search a large solution space. A local optimisation algorithm is not used as some combination of variables might generate anchors which do not match any lesions and a slight modification to these variables might not produce any changes which could get the value stuck in a local minimum.

Running this algorithm only takes several hours compared to the original method, which required training the model for each anchor configuration. It also removes the guess-work and finds the anchor configuration, which achieves the best overlap with our dataset. The whole approach can also be applied to new datasets as it makes no assumptions about the samples.

3.4 Generating Dense Masks from Weak RECIST Labels

The standard RetinaNet architecture only requires bounding boxes for training. However "weak" labels such as RECIST diameters which are routinely generated in clinical practice and have been provided in the DeepLesion dataset as well. To leverage this additional information, we automatically generate dense lesion masks from the RECIST diameters using GrabCut [42].

The algorithm works by labelling the pixels in the image as background (T_B), foreground (T_F) and unclear (T_U). This initialisation is used to generate a Markov random field over the pixel labels. The unclear pixels are iteratively set to (T_B) and (T_F) by solving the min-cut of this graph.

Cai et al. [3] demonstrated how GrabCut can be used to generate lesion masks from the RECIST diameters for weakly-supervised lesion segmentation in 2D. They then further use a CNN for mask propagation to generate a 3D mask from the mask generated using GrabCut.

Their T_B is set as pixels outside the bounding box defined by RECIST axes, and T_F is obtained by dilation of the diameters so that the new diameters takes up 10% of the area

in the bounding box.

Such initialisation may be sub-optimal because a large portion of the image is not labelled as T_F or T_B, specifically, for large lesions, where a considerable number of lesion pixels, which are quite sure to belong to the foreground, are not covered by the dilated diameters and omitted in T_F. For small lesions, the dilation has the risk of hard-labelling background pixels into T_F, which cannot be corrected in the optimisation.

Yi [28] argued that lesions are often elliptical objects, so instead of detecting bounding boxes, it is possible to detect lesion bounding ellipses by representing them as 2D Gaussian distributions on the image plane.

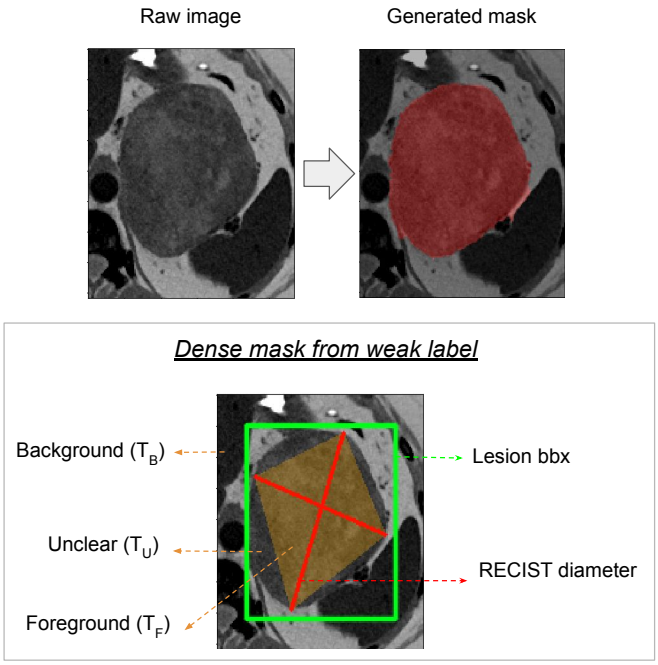


Figure 3.2: An example of a mask generated from RECIST labels. The top images show the original image and the resulting mask. Below we can see the different labels which are used to initialize the GrabCut algorithm to produce the mask.

To achieve higher-quality masks using GrabCut we propose a different strategy which uses the assumption that lesions generally have a convex and elliptical outline. Instead of dilating the RECIST axes we consecutively connect the four vertices of the axes to build a quadrilateral. If a pixel falls inside that quadrilateral it is labelled as T_F, if it falls outside of the bounding box provided with the DeepLesion dataset it is labelled as T_B and all other pixels are labelled as T_U as seen in figure 3.2. This approach dramatically reduces the number of unknown pixels, which improves the results when GrabCut is applied while keeping the simplicity of the strategy.

Some images contain multiple lesions; in those cases, the masks are merged into one. This might result in lesion boundaries being merged; however, for our use-case, this is not an issue as our model also generated bounding boxes and only having a single mask per training sample reduces the model complexity.

3.5 Dense Mask Supervision

To use the generated masks from the previous section, the architecture has to be modified first as RetinaNet only outputs regressed bounding boxes. The design has been modified in the following steps, which can also be seen in figure 3.3:

1. The sub-networks which classify and regress the bounding boxes are shifted from P3-P7 to P2-P6. This retains a higher resolution of the feature maps and reduces the sizes of the anchors (16, 32, 64, 128, 256) and strides (4, 8, 16, 32, 64) by a half. Due to this we also have to multiply the scales by two. This improves the precision for smaller lesions; however, it comes at a cost as the overall number of bounding boxes is higher because the input feature maps have a higher resolution.

2. Two more upsampling layers and multiple convolutional layers are added (called P1 and P2) as well as a segmentation prediction layer (P0) where the P0 to P2 layers follow the same structure of P3-P6. Skip connections are employed by fusing features obtained from C1 and C2 (via a 1×1 convolution which is the same as the other skip connections) and input (via two 3×3 convolutions). We used two convolutions between the input and the segmentation mask to increase the receptive field while keeping a large feature map. These modifications are similar to what Jaeger et al. [21] described in Retina U-Net except for the skip connection from the input to the segmentation mask.

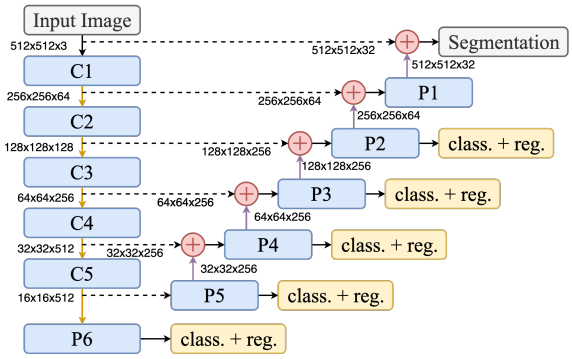


Figure 3.3: Network architecture of RetinaNet with Dense Mask Supervision. C1 to C5 are the 5 convolutional blocks of the backbone. P1 to P6 are the convolutional layers introduced by RetinaNet. Compared to 3.1 Segmentation, P1 and P3 blocks have been added and P7 has been removed. The classification and regression blocks have been shifted by one.

As we have added another output to the network, we have to decide on the loss function which should be used. We have experimented with three different loss functions for the mask output:

- **Cross Entropy** is often used in semantic segmentation but does not work well for our use-case where we have a large imbalance of classes. The output is 512×512 pixels and in most cases, the lesion is smaller than 40×40 pixels, which is less than 1% of the output. If cross entropy was used, then the network would achieve very good loss and accuracy just by classifying everything as negative, but the precision would be very low.

- **DICE Loss** is more suitable when there is a significant class imbalance as it only takes positive samples into account. DICE loss is also very intuitive and aligns with our objective, which is to get the best overlap between the prediction and ground truth; however, it does come with its drawbacks. The Dice Coefficient (D) can be written as

$$D = \frac{2 \sum_i^N p_i t_i}{\sum_i^N p_i + \sum_i^N t_i}$$

$$\frac{\partial D}{\partial p} = \frac{2\sum_i^N t_i^2}{(\sum_i^N p_i + \sum_i^N t_i)^2}$$

where the sum is computed over N voxels, with the prediction labelled as p and ground truth as t. The formula for calculating its derivative has squared terms in the denominator, which could cause training instability when the values are small. In addition during testing we found that the predicted values would usually be very close to the extremes (either close to 0 or 1) and in many cases all of the values would be 0 which wasn't suitable for our use-case as we wanted to see the most probable areas with lesions instead of finding a very accurate segmentation.

- **Focal Loss** was chosen as the loss function for the mask output as it works well for classes with a large imbalance and the derivative doesn't suffer from the same instability as the DICE loss. During testing, we found that the results were much better than when DICE loss was utilised, the mask had a better distribution of predicted probabilities and generated a heatmap instead of a binary output. We found that in some cases, the mask scores for a lesion might be very high even though the classification score for the bounding boxes is very low. For this reason, we rely on the classification score and only if it is high, we display the mask for the bounding box.

This makes it is possible to use this mask prediction as another supervision signal and improve the sensitivity of the model. Compared to networks such as Mask R-CNN [16], our modified model generates a single mask per detection instead of generating a mask per object. This approach does not apply well to other domains where you have many objects which are nearby but works well for lesion detection as in most cases the number of lesions is low and it is more important to find the approximate position and not a perfect outline.

Although the masks and predicted bounding boxes are not always the same because they don't share the entire network we would expect them to correlate. To capture this correlation and use it to achieve better results, we do the following:

1. Normalise the mask to the range of $[0, 1]$ and round all values so that they are either 0 or 1.
2. Generate bounding box for each object in the mask.
3. For each bounding box generated by the classification and regression sub-networks we find a bounding box from the mask with the highest IoU between the pair.
4. The prediction probability of a lesion is updated using $\tilde{p} = p \times (1 + \text{IoU})$ where p was the probability predicted by the classification sub-network.

Higher IoU indicates high confidence in the lesion prediction and hence, the prediction probability is increased. This mainly benefits sensitivity at low FP rates.

3.6 Attention Mechanism for Gated Feature Fusion

We are using the attention gate (AG) model proposed by Oktay et al. [36] which implicitly learns to suppress irrelevant regions in an input image while highlighting useful features by producing an attention map. In their research, it focuses on finding the pancreas and eliminates the necessity of using explicit external tissue/organ localisation modules. In our case, the hope is that the AGs will highlight areas in the body where it is likely to find lesions.

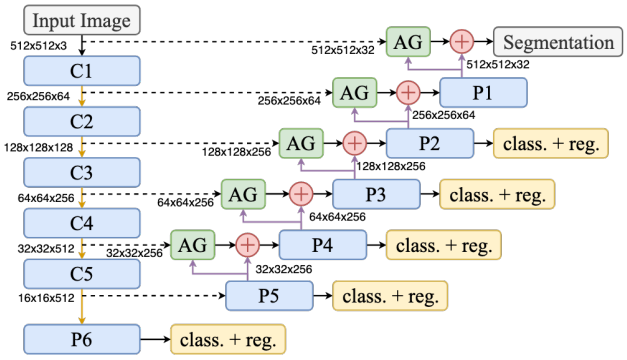


Figure 3.4: Network architecture of RetinaNet with Attention Mechanism for Gated Feature Fusion. C1 to C5 are the 5 convolutional blocks of the backbone. P1 to P6 are the convolutional layers introduced by RetinaNet. AG is the attention gate block. Compared to 3.3 AG blocks have been added. This is the final architecture used.

We introduce the AGs into each of the skip connections before the skip connection is concatenated with the features upsampled from a coarser scale, as shown in figure 3.4. However, compared to the classification and regression sub-networks the AG don't share weights between each other. The AG is used to filter feature responses propagated through skip connections and use features upsampled from a coarser scale as the gating signal.

The AG module only introduces a single new 1×1 convolution and produces a single attention map, which makes it computationally light-weight. Oktay et al. [36] noted that in the case of multiple classes, multi-dimensional attention coefficients should be used. The output of AG is the element-wise multiplication of the attention map and the feature map from the skip connection. The whole design of the attention gate can be seen in figure 3.5, however, there are two differences in our design, we do not use resampling after the ψ convolution as the sizes of both input feature maps are identical and we also introduce batch normalisation after the convolutions.

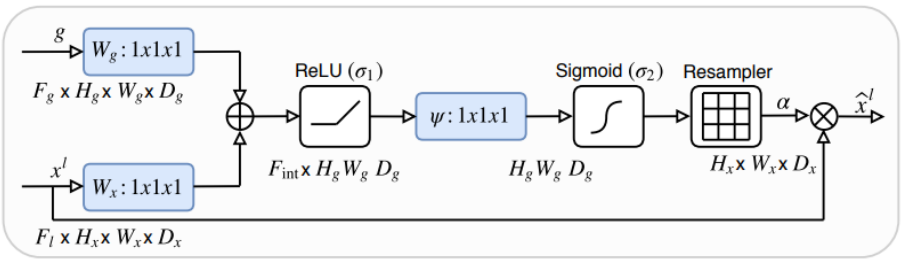


Figure 3.5: Schematic of the AG, x is the feature response propagated through the skip connection and g is the gating signal, in our case those are the features upsampled from coarser scale [36].

Chapter 4

Validation on the DeepLesion dataset

Following all of the modifications made to RetinaNet in chapter 3 we first introduce the DeepLesion [53] dataset in section 4.1. Section 4.2 focuses on the data augmentation done to improve our results and section 4.3 discusses the details of training our model and the improvements made to achieve faster and more stable results. Finally, in section 4.4, we evaluate the model; first, we visually inspect the output of the attention blocks and the final output of the model to ensure that the results are reasonable. Next, we perform an ablation study to measure the effectiveness of our three contributions and compare the results to other literature, which also uses the DeepLesion dataset.

4.1 DeepLesion

DeepLesion is a very recently released dataset and the largest CT scan dataset. Most other datasets contain less than a thousand lesions; this dataset consists of 32,120 CT slices from 10,594 studies of 4,427 unique patients. There are $1\sim3$ lesions in each slice, adding up to 32,735 lesions in total. Most other datasets only focus on collecting only one type of lesion or on only one organ. However, DeepLesion contains lesions from 8 different locations: bone, abdomen, mediastinum (mainly lymph nodes), liver, lung, kidney, soft tissue and pelvis with significantly varying diameters ranging from 0.21 to 342.5mm. Figure 4.1 shows a sample of the lesions.

Most of the current research has focused on developing Computer-aided detection/diagnosis (CADe/CADx) systems that focus on a specific lesion type. For example, the Brain Tumor Segmentation (BraTS) [35] challenge focuses on brain lesion segmentation, and the Lung Nodule Analysis (LUNA) [34] challenge focuses on automatic nodule detection. However, this dataset allows development of CAD systems which focus on multiple lesion types which can be found throughout the body. The size and multiplicity of the dataset will also improve the ability to train deep neural networks which require large amounts of images.

The size of the dataset also makes it easier to train networks without initializing the model with ImageNet [7] or COCO [31] weights. This is particularly helpful for training networks which have more than 3 input channels.

4.1.1 Annotations

The lesions are annotated using response evaluation criteria in solid lesions (RECIST) diameters which consist of two lines: one measuring the longest diameter of the lesion and the second measuring its longest perpendicular diameter in the plane of measurement, however, we found that the two lines are not always perfectly perpendicular. Bounding boxes are also provided, these have been generated from the RECIST diameters by adding/subtracting 5 pixels from maximum/minimum x and y coordinates of the vertices

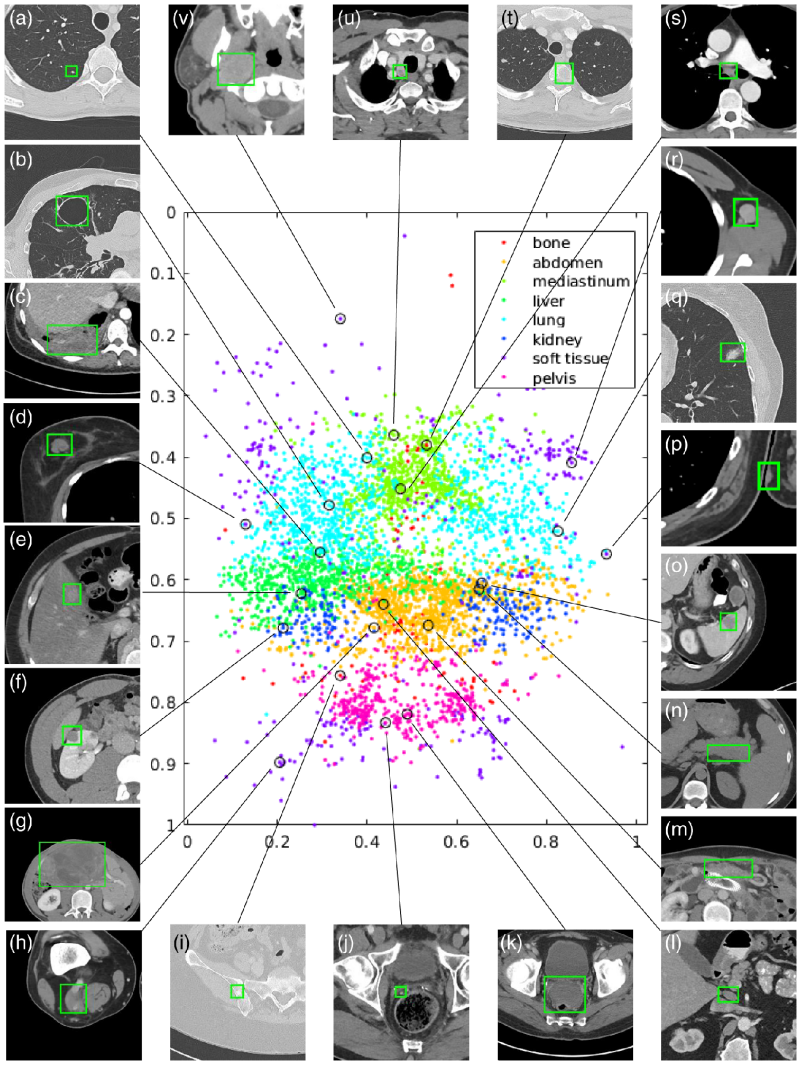


Figure 4.1: Visualization of a subset (15%) of the DeepLesion dataset. The x and y-axes of the scatter map correspond to the x and z-coordinates of the relative body location of each lesion, respectively. Several examples of annotated lesions are shown around the circumference of the diagram [52].

of the two lines. This is usually a reasonable estimate of the bounding box, and only rarely it crops part of the lesion.

For each lesion, there is usually 30mm of extra slices above and below the key slice (slice in which the lesion is labelled) to provide contextual information; however, this is not always the case. Sometimes the key slice does not have a neighbour. The images themselves are single-channel, 16-bit png files. Most commonly, they are 512×512 pixels, but there are a few exceptions.

The images have been split into 70% for training, 15% for validation and 15% for testing by the authors of this dataset, which allows standard comparison of results between different research papers.

There are additional annotations such as the age and gender of the patient, normalized location of the lesion in the body and the lesion type (this is only available for the validation and test data). This information has not been used throughout the project as it is not useful for our task.

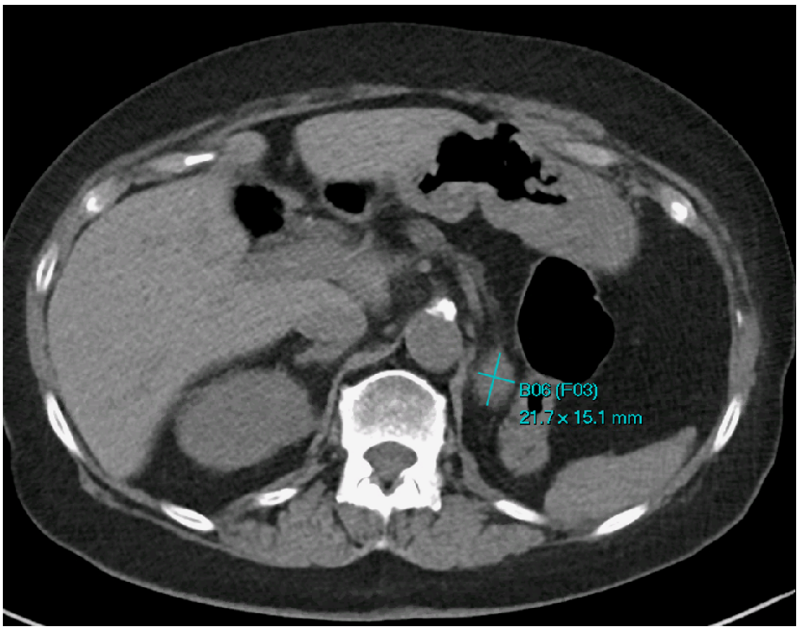


Figure 4.2: Example labeled CT scan [52].

4.1.2 Limitations

The DeepLesion dataset is very useful, but it also comes with a few limitations:

- Only 2D RECIST diameter measurements and bounding-boxes of lesions are available. It has no semantic segmentation masks or 3D bounding-boxes. This makes a segmentation task much harder compared to object detection.

- Lesion type is only specified for the validation and test sets. This makes it very hard to train an object detection network which also classifies the lesion based on its type.

- Not all lesions were annotated in the images, radiologists typically only annotate lesions of interest to facilitate follow-up studies of lesion matching and growth tracking [52]. This means that the object detector might correctly detect some lesions when being trained, but it will get penalized as they aren't annotated and reduce the accuracy of the object detector.

- The dataset contains some noisy bookmarks or bookmarks which measure normal structures such as lymph nodes. The authors note which images are noisy, so it is possible to skip them; however, the list of noisy scans is not exhaustive.

- The spacing between CT slices is not always the same, most frequently the CT slices are 1 or 5 mm apart; however, there are cases of different separation (e.g. 0.625 mm or 2 mm). Interpolation is required to be able to achieve good results when training. One advantage of the difference in spacing is that the trained models will be more robust and should achieve better results when used in practice.

- There aren't always 30 mm of extra slices above and below the key slice, and in some rare cases, the key slice has no extra slices above or below. This makes it harder to use many slices at once with a 2D convolutional network as the number of input channels has to be the same.

4.2 Data Augmentation

Before we can start training the model, we have to preprocess the CT scans so that they can be fed into the model. The scans are made up of axial slices which in the case of DeepLesion are stored as 16-bit single-channel png files where each pixel is an integer in Hounsfield units (HU) [10] (after a fixed offset has been removed). The CT scans first have to be changed to a format which is accepted by RetinaNet.

As most object detectors use traditional 8-bit 3-channel images we have to modify the CT scan to this format. To reduce the number of bits which represent each pixel, we clip the HU window, which is considered. If we want to capture a broad HU range this can be restricted to -1000 to 4000 HU and then normalized to the range of -1 and 1. However, in most cases, even the -1000 to 4000 HU window is too large. In the case of DeepLesion, all of the lesions are within -1024 to 1050 HU, so these values were used to clip the HU window. This can be seen in figure 4.3.

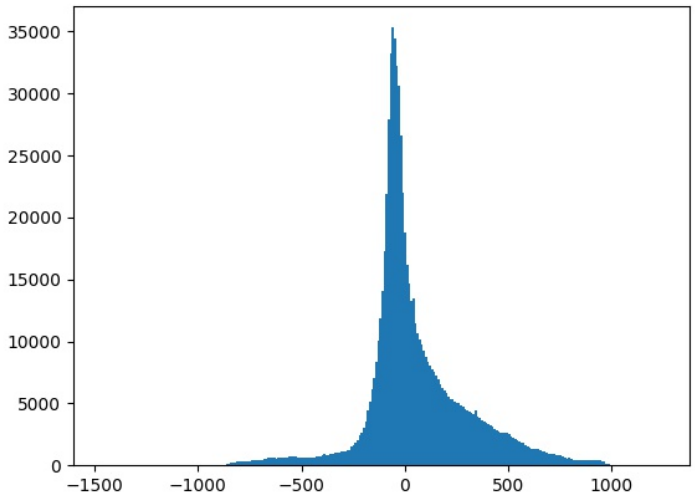


Figure 4.3: Numbers of lesions which are visible at different HU windows. The numbers on the y-axis should be divided by 10 to find the number of lesions which have at least one pixel of a specific HU. This is because each of the histogram buckets covers a range of 10 HU.

All of the images were also resized into 512×512 pixels, resulting in a voxel-spacing between 0.175 and 0.977 mm with a mean of 0.802 mm.

We also experimented with resizing all images so that the voxel-spacing is 0.8 mm. When images are not the same size, padding has to be used during training to ensure that each batch is the same size, although this allows the model to be trained, it might introduce strange artefacts. Another disadvantage of resizing is that it introduces blurring; this is caused by the fact that the newly sampled pixels might be interpolated. When we tested both fixed-sized images and compared them to results achieved when the voxel-size was fixed to 0.8 mm, we found that it was better to use fixed-sized images.

A naive approach to generate 3-channel images is to duplicate the single-channel image two more times. A better approach is to use the neighbouring CT slices, but this is not straightforward as different CT scans are taken with different distance between the slices (most commonly 1 mm and 5 mm). There are two approaches to tackle this problem:

- Take the neighbouring slice without considering the distance between slices. This

is very simple to implement, but the irregularities in the input data will make it harder to train an accurate object detector. When this method was tested, it slightly improved the performance compared to simply duplicating the image; however, the result was not satisfactory.

- Resample the images at a fixed distance. This generally produces better results. In our experiments, we resample the images to a 2 mm separation. We chose 2 mm because half of the samples in the dataset already have another slice 2 mm away, so fewer images have to be resampled. We also tried using 5 mm as the resampling distance; however, the results were worse, this might be because many lesions are less than 10 mm so they might not be visible across all input channels.

In rare cases where neighbouring slices of the key slice are not provided, we duplicate the key slice to fill the missing input channels.

Data augmentation is used where images are flipped in both horizontal and vertical directions with a 50% chance. We also use affine transformations with rotation/shearing of up to 0.1 radians, and scaling/translation of up to 10% of the image size. We find that this augmentation improves performance and reduces the overfitting of the model.

4.3 Training

Due to the size of the RetinaNet architecture and the size of the DeepLesion dataset, it was not feasible to train locally on a personal laptop. Instead, the hardware provided by the departmental Computing Support Group (CSG). To store data, we used the "bitbucket" network storage device. To train the model we initially used the "graphic" machines which are equipped with a range of NVidia GPUs with the weakest being Nvidia GTX 780 and NVidia GTX Titan X. Most of the time we utilized the machines with NVidia GTX 1080, this was because all of the machines operate on the first come first serve basis and the NVidia GTX Titan X was always utilized. Later into the project, we discovered that the "ray" machines also had NVidia GTX 1080, which were mostly unutilized as not many people knew that they contained GPUs. This greatly improved the rate of experimentation as at some points, we had 4 experiments running concurrently on different machines.

To train the network most of the original parameters were used. We also used Adam [23] optimizer but found that the network converged faster with a learning rate of 10^{-4} instead of 10^{-5}. The batch size was set to 4 as this was the highest we could achieve on the hardware available. Also `ReduceLROnPlateau`[1] was used which reduced the learning rate by a factor of 10 when the mean average precision (mAP) has not improved for 2 consecutive epochs, after this reduction a cooldown period of 3 epochs as used to avoid too many consecutive learning rate reductions. To reduce overfitting, `EarlyStopping`[2] is used if the mAP has not improved for 4 consecutive epochs on the validation set.

Our proposed anchor optimisation described in section 3.3 was used to generate new anchors scales and ratios. We obtain optimal scales of 0.85, 1.08 and 1.36, and ratios of 3.27:1, 1.78:1, 1:1, 1:1.78, 1:3.27, which fits objects of small size and large ratios.

Traditionally RetinaNet is trained by initializing the backbone weights to the weights received by training the backbone on the ImageNet dataset or initializing the whole model to weights received by training RetinaNet on the COCO dataset. However, we found that the final results were very similar when trained both with weights or when the weights were randomly initialized. Both of these datasets always only use 3 channels (RGB) however in medical imaging we are not limited to only 3 channels, so we decided to use random

[1] `https://keras.io/callbacks/#reducelronplateau`
[2] `https://keras.io/callbacks/#earlystopping`

weights as it will make it easier to compare the results when more channels are added. The training usually takes 20 to 30 epochs with around 2 hours per epoch.

4.4 Evaluation

The key metric when evaluating the project is sensitivity of the object detector on the test set of the DeepLesion dataset. The dataset has been split by the authors into 70% for training, 15% for validation and 15% for testing. A lesion is considered detected if the intersection over union (IoU) is more than 0.5. This allows our results to be directly compared with other literature, which uses the DeepLesion dataset. The sensitivity is evaluated at 0.5, 1, 2, 4, 8 and 16 average false positives (FPs) per image.

Another important metric is the sensitivity for different lesion sizes; the sizes are split into three categories based on the length of the shorter RECIST diameter. The first category contains lesions smaller than 10mm, the second contains lesions between 10mm and 30mm and the third contains lesions larger than 30mm.

A secondary metric which we also report is the average inference time; this is not directly comparable to other research due to the hardware differences however it can be used to indicate the added complexity of our modifications to RetinaNet.

4.4.1 Current state-of-the-art results

Given that the DeepLesion dataset had only been published in July 2018 there are only 3 relevant publications which can be directly compared to this work. Alongside releasing the DeepLesion dataset, Yan et al. [52] also demonstrated how it can be used for lesion detection and provided baseline results. They used Faster R-CNN and adopted VGG-16 as the backbone of the network. Tang et al. [48] demonstrated results using Mask R-CNN, used pseudo mask construction and hard negative example mining to propose a new universal lesion detection method which they named ULDor. Currently, the best results have been achieved by Yan et al. [51] when they adopted the R-FCN network and incorporated 3D context into the design.

4.4.2 Visual evaluation of the attention mechanism

When Oktay et al. [36] used an attention gate (AG) module for suppressing irrelevant regions during segmentation of the pancreas, they were able to achieve excellent visual results. We want to see if the same is the case when the AG module is applied to our modified RetinaNet which does both segmentation and object detection and when trained on a more varied dataset which contains 8 different lesion types across a broad HU range.

We inspect the output of the ψ convolution after the sigmoid activation function has been applied, as seen in figure 3.5. These values are then transformed between 0 and 1 and overlaid on the original image, red indicates probability closer to 1 and blue indicates probability close to 0. If the attention map is smaller than the input image it is resized without using interpolation. There is a green box in each image which displays the ground truth annotation for reference.

In total we use 5 AG modules, first is connected in the skip connection between the input image and the segmentation output, the others are connected in the skip connections between C1-C4 and P1-P4, this can be seen in figure 3.4. In figure 4.4 we display 3 CT samples and for each sample 5 attention maps.

We find the results to be very intriguing as each attention map is very different and seems to be focusing on different structures with a different purpose. Looking at the two finest attention maps (left), we can see that they highlight regions with high lesion probability, the second column, however, is much more discriminate compared to the first

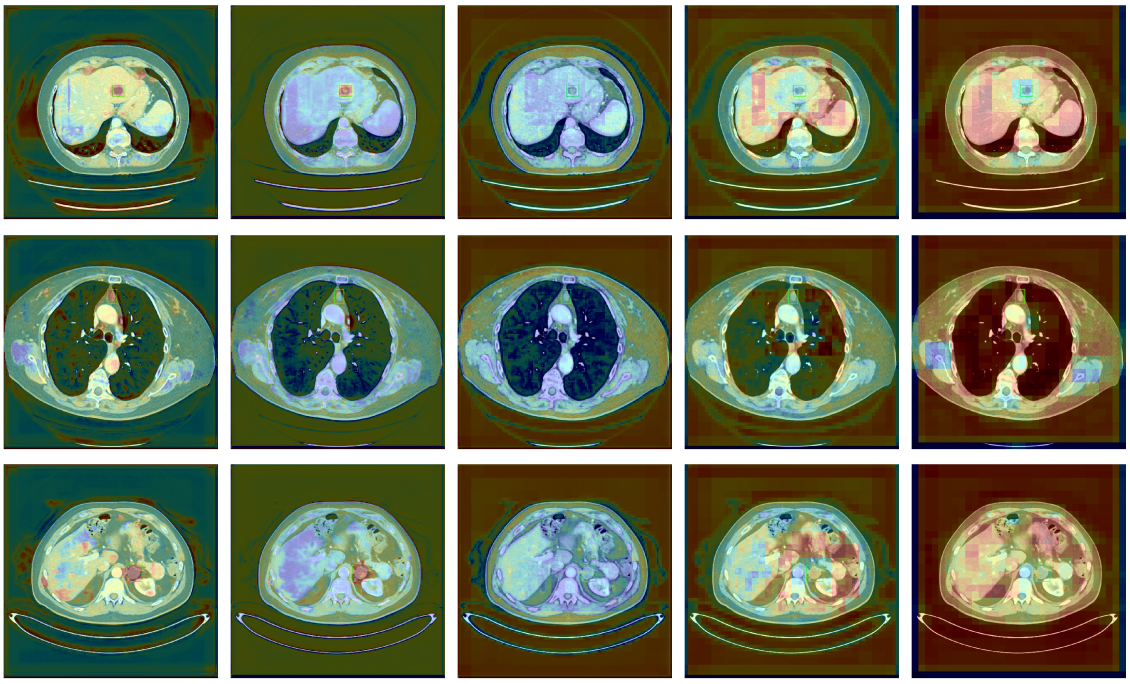


Figure 4.4: Example attention map overlaid on the input image, red indicates probabilities closer to 1 and blue indicates probabilities close to 0. The images are ordered from finest to coarsest on each row. Left-most attention map is between the input image and the segmentation where P1 is used as the gating signal. The right-most attention map is between C4 and P4 where P5 is used as the gating signal.

one. We expect that this is because the second AG module uses the output from the P2 block as the gating signal, this is also the part of the network where the classification and regression sub-networks are connected and produce predictions for small lesions. However, when we look at the third column, we can see that most of the probabilities are low, which implies that the information from this skip connection is not as useful. The last two columns have most of the area's red which means that most of the information from the skip connections is passed through, one observation which we can make is that the probabilities are lower around bones, this is especially visible in the second row. The same behaviour could be seen in most of the other images which are not displayed here.

Although it is not evident at first glance what the AG modules have learned, they seem to be producing relevant results and improve the specificity of the model overall. They are not clearly highlighting specific organs as demonstrated by Oktay et al. however, this task involves a large variety of organs and structures, which makes the task harder.

4.4.3 Visual results of lesion detection

In figure 4.5 we can see visual examples of detecting lesions on test images, a more extensive set of images which covers all 8 types of lesions which are contained in the DeepLesion dataset can be seen in Appendix A in figure A.2. The probability threshold is set to 0.3 yielding 0.5 false positives (FPs) per image on average. The figure displays lesions of various sizes, appearance and type. The overlap between the ground truth and the predicted bounding box is very accurate and in some cases, the ground truth box is not even visible because of perfect overlap. The right-hand column shows heatmaps which have been generated by dense mask supervision, which shows that the additional output is not only useful as an additional training signal, but the output is also very accurate. It is also visible that our approach to generating dense labels from RECIST diameters yields good

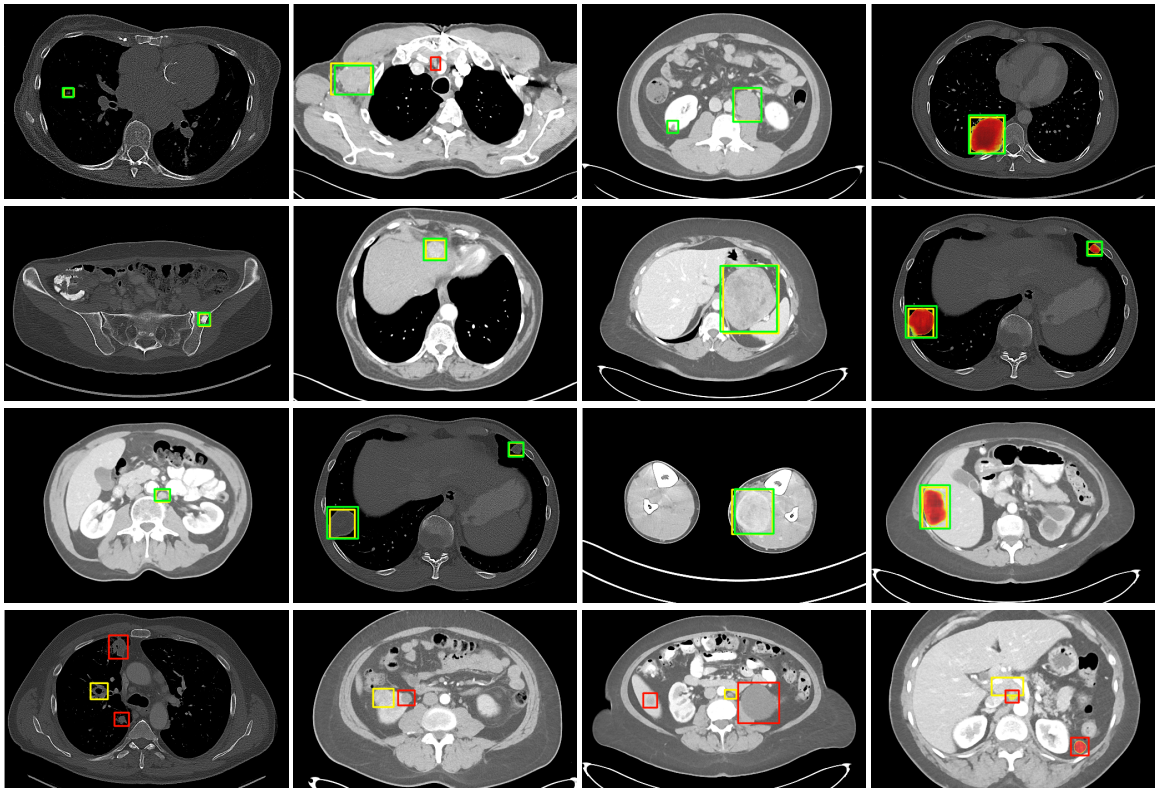


Figure 4.5: Visual results for lesion detection at 0.5 FP rate using our improved RetinaNet. The first three columns show different sizes from small to large. The right column shows heatmaps from the segmentation layer overlaid on detections. Yellow boxes are ground truth, green are true positives, red are false positives. Last row shows intriguing failure cases with possibly incorrect ground truth.

results which are sufficient for training and it is not necessary to generate them manually. In the last row, we can see several results where the network fails to detect the lesion correctly. The detections look plausible and might be negative because some of the lesions were not labelled as they weren't relevant to the clinicians who annotated the dataset.

4.4.4 Ablation study

In Table 4.1, we present the lesion detection sensitivities with different FPs per image. As a baseline we quote the results provided by Yan et al. using Faster R-CNN (reported in [51] as they do not report results for 4 FPs in the original paper) and a result provided by Tang et al. [48] using Mask R-CNN. The current best results achieved by Yan et al. [51] using a modified Faster R-CNN called 3DCE and results achieved by Tang et al. [48] using a modified Mask R-CNN called ULDor are also reported. In an ablation study, the results mentioned above are first compared to the original RetinaNet without any modifications and then our improvements are incrementally added with the final improvement showing the results for our final model.

The results show that the default anchor configuration performs poorly, even worse than Faster R-CNN and Mask R-CNN, which indicates that out-of-the-box approaches are sub-optimal for medical imaging. This is because datasets such as COCO, which are used to design better object detection models, generally contain larger objects of many classes and medical imaging datasets contain smaller objects which are usually classified into fewer classes.

After the automatic anchor optimization was used to optimize the scales and ratios

Table 4.1: Detection performance of different methods and our ablation study at multiple false positive (FPs) rates.

Methods	0.5	1	2	4	8	16	runtime
Faster R-CNN [40]	56.90	67.26	75.57	81.62	85.83	88.74	32 ms
Mask R-CNN [16]	39.82	52.66	65.58	77.73	85.54	91.80	-
ULDor (Tang et al. [48])	52.86	64.80	74.84	84.38	87.17	91.80	-
3DCE (Yan et al. [51])	62.48	73.37	80.70	85.65	89.09	91.06	114 ms
original RetinaNet [30]	45.80	54.17	62.50	69.80	75.34	79.48	28 ms
+ anchor optimization	64.82	74.98	82.29	87.87	92.20	94.90	31 ms
+ dense supervision	70.24	78.28	85.10	90.39	93.81	96.01	39 ms
+ attention gate	**72.15**	**80.07**	**86.40**	**90.77**	**94.09**	**96.32**	41 ms

Table 4.2: Detection performance of the modified anchor configuration compared to the original configuration.

Methods		0.5	1	2	4	8	16
Original config.	<10mm	13.99	20.49	27.98	35.77	43.19	49.44
	10-30mm	58.32	68.26	77.28	85.25	90.22	93.47
	>30mm	75.03	81.17	89.30	92.71	95.81	98.14
Modified config.	<10mm	61.90	71.90	78.77	84.85	89.50	92.70
	10-30mm	65.60	75.61	83.24	88.93	92.83	95.71
	>30mm	69.77	80.62	87.90	91.16	94.41	96.90

of anchor boxes, RetinaNet already outperforms previous state-of-art models. This is a relatively inexpensive step as it does not require any modification to the model or the input data; neither does it require extensive hyperparameter optimization and a lot of hours of training. This whole step can be done automatically, which makes it an ideal first step before investing a lot of time and resources into modifying the network architecture. The sensitivity at 0.5 FP is 2.34% higher than 3DCE and this improvement can be seen across all FP rates. The increase in runtime can be attributed to the fact that the model produces more positive predictions which require non-maximum suppression to be applied to more objects; however, the number of weights has not changed at all.

The optimised anchor configuration results in an average IoU of 0.7046 compared to the original 0.5808 IoU. Another key metric is the number of lesions which are not matched by any bounding box as unmatched lesions effectively reduce the training set size; the new configuration successfully matches all of the lesions in the training set, the original configuration fails to match 5573 samples which is 26.10% of the training set.

In table 4.2, we can see a more in-depth evaluation of the sensitivity achieved on the DeepLesion dataset at different FP rates when both original and modified anchor configuration is used. From this, we can see that the model which was trained with the original configuration struggles to find small ($<$ 10mm) lesions reliably or even mid-size (10-30mm) lesions. This is greatly improved by the modified configuration; however, it is still harder to find smaller and mid-sized lesions compared to the large lesions. The original configuration is better at finding lesions which are larger than 30 mm; we suspect this is because the smaller lesions were not matched and were ignored during training, so the training set was skewed to larger lesions which wasn't the case with the original configuration.

Adding dense supervision with segmentation masks generated from RECIST diame-

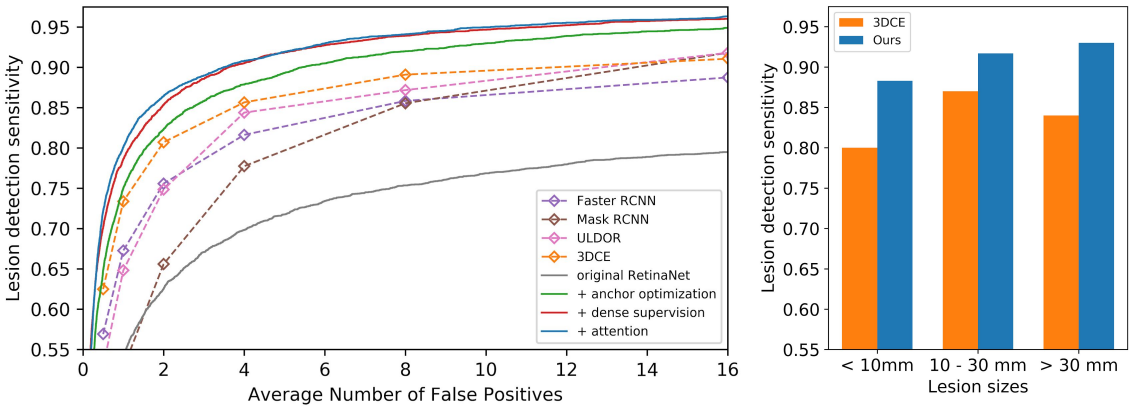


Figure 4.6: **Left**: FROC curves for our improved RetinaNet variants and baselines on DeepLesion dataset. **Right**: Per lesion size results at 4 FPs compared to 3DCE [51].

ters significantly boosts detection sensitivity across all FP rates, with 5.42% improvement at 0.5 FP. The pixel-wise supervision adds an important training signal, providing more precise localization information in addition to bounding-boxes. This step adds additional complexity as it requires multiple convolutions and upsampling blocks to be added, which increases the runtime by 25% however, it still runs faster than 3DCE. Using the consistency between bounding-box regression and dense classification also helps to reduce false positives, although this hasn't been added as a separate entry in the table the main improvement was seen at 0.5 and 1 FP per image. Finally, adding an attention mechanism further improves the performance; however, this is not as significant at higher FP rates. Overall our combined architecture achieves a sensitivity of 72.15% at 0.5 FPs and 90.77% at 4 FPs, with an improvement of almost 10% at 0.5 FP over the best-reported results.

In figure 4.6, we display the free-response receiver operating characteristics (FROC) curves of different methods. For Faster R-CNN, Mask R-CNN, ULDor and 3DCE we use 6 points to plot the lines as we could only get data for 0.5, 1, 2, 4, 8 and 16 FPs which is the same as displayed in table 4.1. On the right in figure 4.6, we can see the performance of our model compared to 3DCE for different lesion sizes. The lesions are divided into three groups based on the length of the shorter RECIST diameter. A significant difference between our model and 3DCE can be seen for lesions smaller than 10 mm, in this case, our sensitivity is 88.35% compared to 80% for 3DCE however model still outperforms 3DCE at different sizes as well. We achieve sensitivity of 91.73% compared to 87% achieved by 3DCE with lesions between 10mm and 30mm, and for lesions larger than 30 mm the sensitivity of our model is 93.02% compared to 84% of 3DCE. Although 3DCE performs well compared to other approaches, it seems that the additional 3D context is not as helpful for small lesions compared to medium and larger ones. The feature pyramid successfully retains responses from small lesions and together with dense supervision it can achieve very good results mainly on small lesions, this is critical as it is easier to miss a smaller lesion in a large scan when it is analyzed by clinicians.

Each lesion in the test and validation datasets also is assigned one of the 8 lesion types. In table 4.3, we can see the sensitivity at 4 FPs per image on the test set of DeepLesion. We report the accuracy of Faster R-CNN and 3DCE (reported in [51]) as well as our results with anchor optimization, dense supervision and attention gate improvements.

From the results, we can see that our model achieves better results for almost all lesion types except for the soft tissue type, which contains lesions in muscles, skin, and fat. This might be because there aren't a lot of samples of soft tissue lesions in the dataset and they are more spread across the body compared to kidney or bone lesions. Compared to the results achieved by Faster R-CNN and 3DCE we achieve a lower spread of sensitivities

Table 4.3: Detection performance of different lesion types at 4 false positives (FP) per image. The last row contains the number of samples of each lesion type in the test set. The lesion types have been sorted based on the size of the test. The abbreviations of lesion types stand for abdomen (AB), lung (LU), mediastinum (ME), liver (LV), pelvis (PV), soft tissue (ST), kidney (KD), and bone (BN), respectively. The mediastinum type mainly consists of lymph nodes in the chest. Abdomen lesions are miscellaneous lesions that are not in the liver or kidneys. The soft tissue lesion type contains lesions in muscles, skin, and fat [51].

	Lesion type							
	AB	LU	ME	LV	PV	ST	KD	BN
Faster R-CNN	75	88	84	80	76	76	72	55
3DCE	80	91	88	84	81	**82**	76	63
Ours	**89**	**93**	**93**	**94**	**90**	80	**87**	**83**
Number of samples	1171	1100	864	700	409	327	233	108

among the different lesion types, the highest sensitivity is 94% for liver lesion type and the lowest is 80% for soft tissue lesion type which results in a spread of 14%. 3DCE achieves 91% for lung lesion type and 63% for bone resulting in 28% spread. We can see the greatest improvement with bone lesions where the sensitivity improved by 20% from 63% achieved by 3DCE to 83% achieved by our model.

Chapter 5

Validation on the whole-body MRI dataset

We were able to achieve a sensitivity of 90.77% at 4 false positives per image on the DeepLesion dataset which significantly outperforms all other research on the same dataset. In this chapter we will demonstrate that our model can be applied to another medical dataset without any modifications to the architecture and is truly a universal lesion detector.

To do this we will use a whole-body Magnetic Resonance Imaging (MRI) dataset and attempt to correctly detect primary colorectal and lung lesions. Most of this chapter focuses on describing the work required to preprocess and augment the data, how we overcame several difficulties such as finding lesions in a 3D volume with a network which uses 2D slices. Finally we will evaluate our performance on the dataset and discuss which changes were the most beneficial.

5.1 Whole-body MRI dataset

The second dataset used is an in-house whole-body MRI dataset which contains Apparent diffusion coefficient (ADC), Diffusion-weighted (DW) and T2-Weighted (T2W) imaging protocols as well as semantic annotation of a primary colorectal and lung lesion in each scan. The dataset is much smaller than the DeepLesion dataset with only 213 MRI scans, however we found 3 of these scans to contain artefacts preventing us from using them so only 210 scans will be used.

Figure 5.1 displays 9 slices (3 slices per imaging protocol) from 3 different MRI scans. The figure is a good example of the modality and richness of the data but also it demonstrates that the challenges which we will face such as noise, missing slices and misalignment.

5.1.1 Annotations

The dataset was provided with semantic segmentation's of lesions. In each scan only the primary lesion should be annotated however in some cases the annotations were noisy which resulted in multiple disjoint partial segmentation of the lesion or it contains multiple smaller volumes which were probably introduced accidentally during the annotation process. To remedy these issues we applied 1 iteration of binary dilation to connect the semantic segmentation's which are disconnected but are close to each other. Afterwards 1 iteration of binary erosion was applied to counteract the dilation while keeping the volume connected. Finally all holes in the mask were filled in, the assumption was that if a voxel is entirely surrounded by a lesion then it should be considered to be a part of the lesion. If the segmentation volume still contains multiple volumes the largest one is retained and the rest are removed.

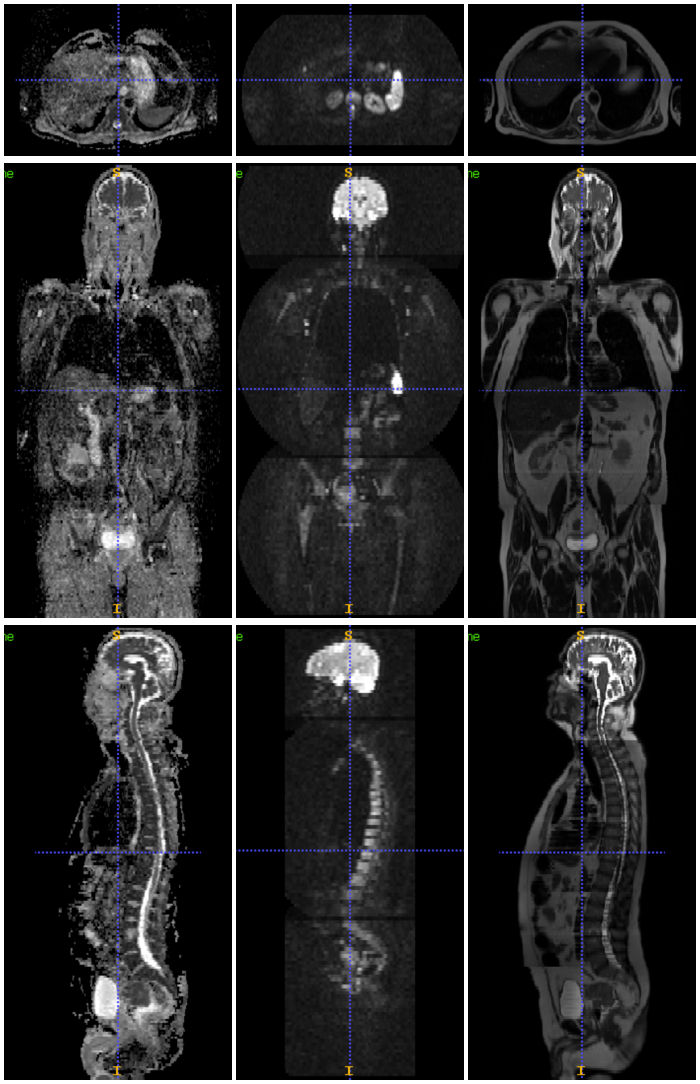


Figure 5.1: Example of an ADC, DW and T2W scan (from left to right). For each of the imaging protocols three different slices are shown. They are taken along the axial, coronal and sagittal plane (from top to bottom).

5.1.2 Limitations

Comparatively to the DeepLesion dataset the whole-body MRI dataset is much smaller. It contains only 210 usable scans whereas the DeepLesion dataset consisted of 32,120 CT slices from 10,594 studies of 4,427 unique patients. Although the 210 scans can generate more 2D samples which contain a lesions, the slices are highly correlated so they add less value than having more scans from unique patients. This means that it will be harder to get good results and more care is required during training to stop the model from overfitting.

A major drawback of the dataset is that it has been collected from multiple sources on different MRI systems which have different field strengths and coil characteristics. Another drawback compared to CT scans is that image intensities in MRI do not have a fixed interpretation so if the same patient is scanned using a different protocol the intensities might be different. Although this might not be a problem if a human reader is inspecting the scan it will pose a problem for our model if correct preprocessing is not used. An example of this can be seen in figure 5.2.

The last limitation which reduces the dataset size usable for training is the occurrence of artefacts and noise in the scan. An example of few artefacts can be seen in figure 5.3.

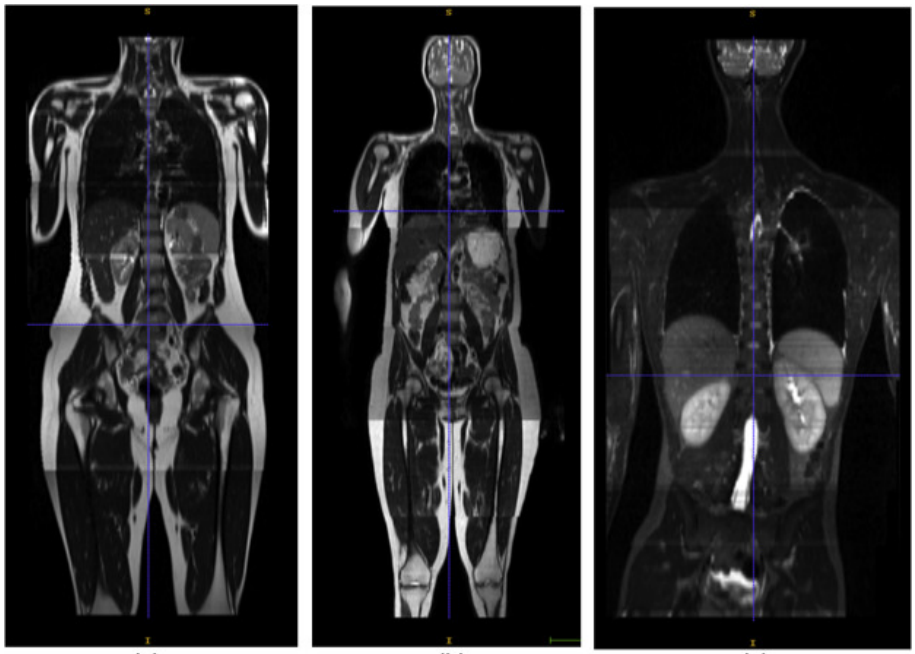


Figure 5.2: Multiple T2W scans taken with different protocols [25]

5.2 Data Augmentation

As the dataset is made up of 3 different scans, namely ADC, DW and T2W, it is first required to prepossess them individually before they are merged and fed into the model.

First all of the scans are resized so that they have the same size and spacing. In our case the spacing is (1.5, 1.5, 5.5) and the size is 320x320x232. Although fully convolutional networks don't require all input images to be the same size, if multiple images are fed in a single batch then the input size has to be the same for the batch so it was easier to resize the entire scan instead of ensuring a whole batch has the same size at runtime.

To achieve the best results it is important to rescale the voxel intensities such that most of them are between -1 and 1. We experiment with two different approaches to do this.

This first approach uses clipping of intensities which removes all outliars. The ADC values are clipped between $[0, 3800]$, DW values are clipped between $[0, 1050]$ and T2W are clipped between $[0, 8200]$. After the values are clipped they are normalized between -1 and 1. This approach worked well on the DeepLesion dataset because the values ended up centered around 0 however in this case the distribution of intensities is positively skewed and most of the intensities after normalization end up centered around -1.

To tackle the skewness of the data we use a StandardScaler implemented in `sklearn` [38]. This rescales all of the values so that the mean is 0 and the standard deviation is 1. Given that a large part of the scan is background all zero intensities are ignored when calculating how the values should be transformed. This approach produces much better results than using clipping.

During experimentation we noticed that the ADC and T2W scans are not perfectly aligned so we experiment with registration to transform the T2W scan. Initially the centres of mass for both scans are aligned, then Mutual Information is used for Rigid Transformation and consecutively for Affine Transformation. Both of these transformations are done at 3 different resolution settings. The lowest resolution is 4 times smaller than the original image and Gaussian blurring with sigma of 3 is applied, at this level the algorithm was limited to 10000 iterations. The next resolution is 2 times smaller than the original one

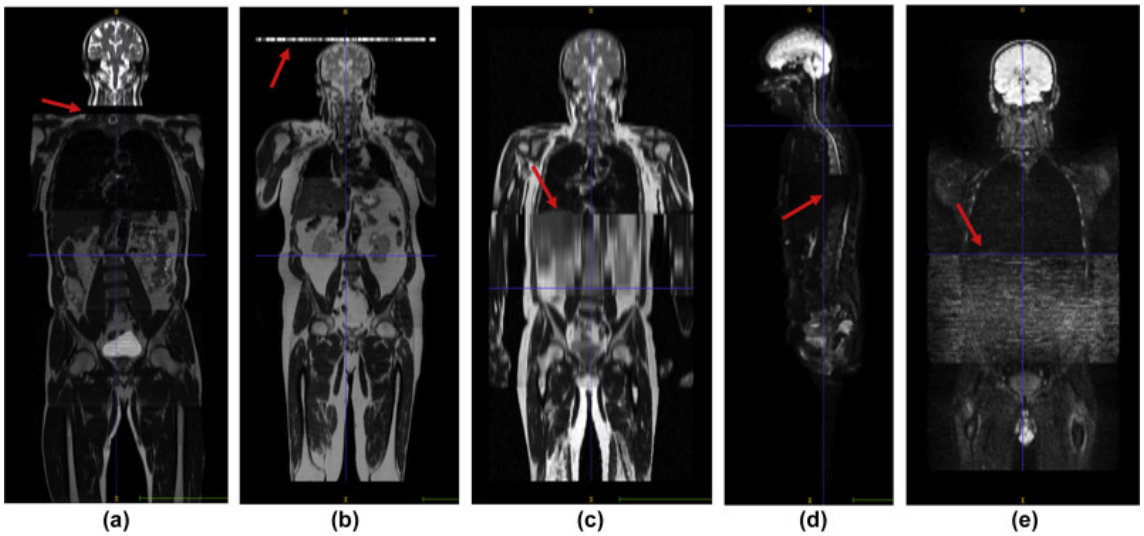


Figure 5.3: Example of the data quality challenges (artefacts) we encountered in the dataset such as missing slices, interference and motion artefacts [25]

with sigma of 1 and 1000 iterations and then the last round of optimization is applied at the original resolution without any blurring for 100 iterations. To reduce the time taken for registration the Mutual Information is calculated by only sampling 10% of the pixels and 32 bins are used for computing the histogram intensity. During evaluation we found that in some cases the scans were better aligned but in others it resulted in a lot of unnatural elastic deformation and did not make a meaningful impact on the results. We did not use registration in the final model.

We have also experimented with several forms of denoising in an attempt to improve the results however we were not able to do so. The following methods were attempted:

- Gaussian denoising was applied on the ADC scans with multiple different sigma values ranging from 1 to 5. The best results were for sigma of 1 however the accuracy was still lower than without denoising.
- Noise2noise [26] was used because it has been previously successfully applied to other MRI data. The results when using this were better than when Gaussian denoising was used however there was no improvement. However for this experiment an unofficial Keras implementation was used instead of the official one with a slightly different model so that could have been one of the causes.
- Non-local means denoising implemented by the dipy [11] library was used with both Rician and Gaussian noise. It provides a function to estimate the amount of noise from local patches which avoids having to use hard-coded values. This was tested on both ADC and DW scans.

To further reduce overfitting we apply elastic deformation to the entire scan. This allows us to augment the data in all 3 axes. The deformation grid has three points with a sigma of 25. For each scan which is in the training set we generate 4 additional elastically deformed scans.

Our model requires both semantic segmentation and bounding boxes for training, generating bounding boxes is however easier than generating semantic segmentation masks from bounding boxes as we have done previously. The bounding box is generated separately for each axial slice of the MRI by finding the minimum and maximum x and y values for the segmentation in that slice and adding or subtracting 5 pixels. This is done so that

the smallest bounding boxes are not ignored during training. The expansion of bounding boxes might introduce a slight amount of noise during training however it ensures that every slice which has a partial semantic segmentation will also have an associated bounding box which is big enough so that it isn't ignored by the classification and regression sub-models. Another advantage of slightly increasing the bounding box size is that more bounding boxes are detected by the coarser levels in the feature pyramid which should theoretically improve the results as there are more convolution layers between the input and output.

5.3 Training Data Generation

Our model only uses 2D convolutions and produces 2D bounding boxes and heatmap. This makes it impossible to feed in all of the slices at the same time into the model. Instead we only input 9 axial slices at a time for each of the 3 scans, this results in a total of 27 channels. To reduce the number of false positive predictions we also randomly sample 8 negative slices (slices which don't contain a lesion) from each scan so that there is a roughly equal number of slices with and without a lesion. For a slice to be selected it must contain at least one pixel with intensity greater than 0 in the ADC scan. It also cannot be within 5 slices of another slice which contains the lesion. This is done to avoid selecting slices which might still contain a small part of the lesion even though they have not be labelled.

We split the dataset into training, validation and test such that there are 150 (71.4%) scans in the training set, 30 (14.3%) scans in the validation set and 30 (14.3%) scans in the test set. The dataset contains 136 scans with colorectal lesions and 74 lung lesions. To ensure that the test and validation sets are roughly representative of the dataset there are 10 lung lesions and 20 colorectal lesions in both the validation and test sets.

5.4 Training

One of our goals when training on the whole-body MRI dataset was to make minimal modifications to the model design to demonstrate that the same model can be applied to other datasets so no modification was made to the architecture itself. The only modification was to the configuration parameters which define the dimensions of the anchors and the batch size. The same scales were used for this dataset as for DeepLesion however instead of using ratios of 3.27:1, 1.78:1, 1:1, 1:1.78, 1:3.27 we used ratios of 2:1, 1.5:1, 1:1, 1:1.5, 1:2.

We were able to increase the batch size from 4 to 8 samples given that the input size to the 2D convolution is 320×320 instead of 512×512 (number of channels is omitted as increasing the number of input channels in a 2D convolution does not increase the number of weights in further convolutions).

The model was trained using the same method as when trained on DeepLesion dataset however we found that training from randomly initialized weights was very unstable. Instead we used the weights trained on the DeepLesion dataset.

5.5 Stabilizing loss when negative samples are used

When we used negative samples during training we noticed that the loss was very unstable, this however was not the case when only positive samples were used. The problem got worse as the ratio between positive and negative samples increased. Initially the thought behind this was that the network is trained to produce a lot of positive predictions and is penalized more when there are no positive samples in the training batch. However upon

further inspection of the implementation of Focal Loss which was used during training we noticed that the loss is divided by the number of positive targets in the batch. In the classification sub-network the number of positive targets is the same as the number of anchors which have more than 0.5 IoU with the ground truth which is usually at least 5 per sample. In the semantic segmentation output there is usually at least 100 positive targets per sample. However if all samples in a batch are negative we would divide by 1.

Given that the batch size used was 8 and there was the same number of positive and negative samples in the dataset there was a $0.5^8 = 0.0039$ (0.39%) probability that a batch was completely negative. In this case the loss was not divided and could be 10 to a 100 times higher than usual.

To remove these loss spikes we first tried to use the mean loss instead of dividing by the number of positive targets. This greatly reduced the loss as most samples are classified correctly as negative and removed spikes in the loss. The drawback of this approach is that it puts more weight on larger lesions as the output of the semantic classification has more positive pixels and the loss is higher in this case. This is contradictory to our goal as larger lesions are easier to classify. Another issue with this approach was that RetinaNet also uses L1 loss for the output of the regression subnetwork, because the Focal Loss got reduced, the overall weight of the L1 loss was higher which indicated to the network that it is more important to have a good overlap between the predicted bounding box and the ground truth but it is not as important to correctly classify the lesion. Again this is contradictory to our goal as detecting the lesion is more important.

To overcome the problems with the method above we instead keep a running average of the number of positive ground truth outputs in a batch and in the case that the whole batch has only negative samples we divide by the loss by the average instead of dividing it by 1. This way the ratio between L1 and Focal Loss remains the same and we take into account the size of the lesion. There is still an issue that we don't take into account how many positive and negative samples there are in a batch, we only address the case with 0 positive samples however there might be 1 positive and 7 negative samples. Although in this case the loss might still be divided by a low number introducing a small spike, during training we found that our solution above greatly improved the stability and there was no need for further optimization.

5.6 Merging 2D predictions

To train and validate the model we feed in 2D slices of the MRI scans and the model outputs 2D masks and bounding boxes. Our goal is to produce 3D segmentation so we need to devise a suitable method of combining our results into a single prediction.

The initial approach was to merge the bounding boxes on adjacent slices if they had an IoU more than 0.5 into a single 3D prediction. However this approach has several shortcomings:

- It is possible that two bounding boxes on the same slice have an overlap of more than 0.5 with another bounding box on the adjacent slice. If we add both bounding boxes into our prediction we effectively create a fork in the lesion which most probably doesn't represent its structure, we could either merge the 2 boxes or we can pick the box with the highest probability. This seems like a reasonable approach however greedily picking the best anchor can lead to a sub-optimal final result.
- Using 0.5 as an IoU between slices is very arbitrary and introduces another hyper-parameter which needs to be optimized.
- It is not clear how to calculate the final prediction score for the lesion based on the scores from the individual bounding boxes.

Instead of relying on the bounding box we used the mask output and concatenated the outputs for all slices into a single volume. Then to generate a binary mask we normalize the prediction volume so that all values are between 0 and 1 and then round the values to give us either 0 or 1. There are usually some very small predictions which are too small to be lesions so all predictions which are only on a single slices are ignored. Bounding boxes are then used to define the score of a prediction, the bounding box with the highest score and with IoU of 0.5 with the prediction is used.

5.7 Evaluation

In this section we will evaluate how differences in preprocessing, training and evaluation affect the sensitivity of our model. We cannot compare it to other research as the whole-body MRI dataset is completely new and no one else has trained models on it for the same task and to the same extent as we have. The task of colorectal and lung lesion detection in whole body MRI using machine learning has not been extensively studied so it is not possible to compare to other models even if they have been trained on other datasets.

5.7.1 Visual results

In figure 5.4 we can see a visual example of a DW scan and a ground truth segmentation of the lesion. In figure 5.5 we can see the segmentation mask generated by our model. The prediction shows two volumes which might be colorectal lesions, there are also two lung lesions predicted (not shown in the image). There are also several very small predicted volumes next to the main predictions. As these are separate they might increase the number of false positive predictions in our evaluation even though they are very close to each other. To improve the results there needs to be another post-processing step which removes small noisy predictions however it is not possible to just do this based on the size of the prediction as lung lesions are also very small.

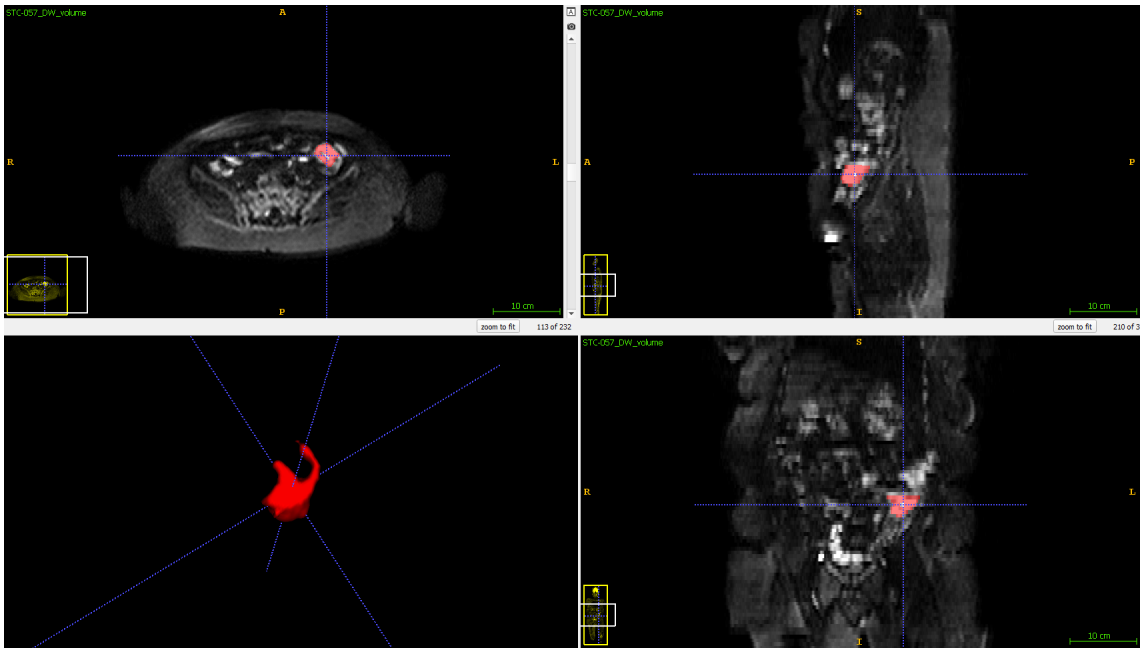


Figure 5.4: The ground truth segmentation overlaid on a DW scan. Bottom left view shows a 3D render of the lesion and the other show 2D slices of the MRI.

Instead of displaying the binary mask segmentation it might be more useful to display a heatmap. An example can be seen in figure 5.6. This focuses the clinicians focus at the

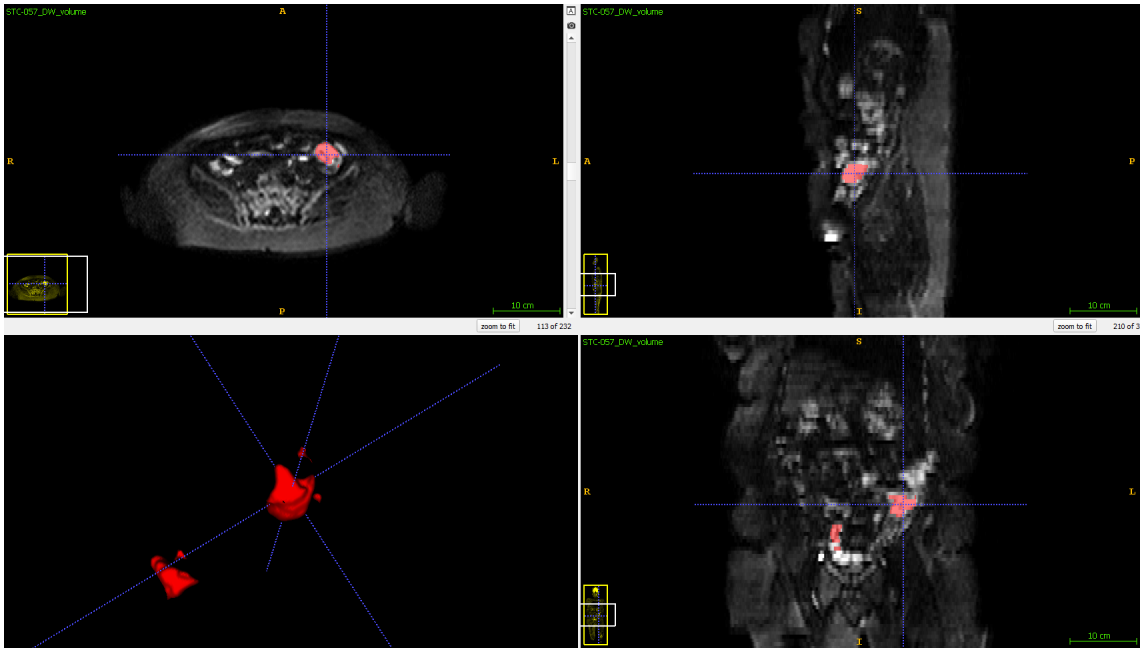


Figure 5.5: The predicted segmentation overlaid on a DW scan. Bottom left view shows a 3D render of the lesion and the other show 2D slices of the MRI.

relevant parts and also includes the relative importance instead of a binary classification. We can see that the main lesion has areas with red (higher probability) compared to the secondary prediction.

More examples of predictions and ground truth can be also found in Appendix B in figures B.1, B.2 and B.3.

5.7.2 Transfer Learning

It is possible to train the models with initializing it with weights from a previous model or to use random weights. It can sometimes be beneficial to use previously trained weights as many of the deeper layers might not change much and the weights might already contain useful features. We will use the weights trained on the DeepLesion dataset, to make the weights fit better for our usecase we trained the model on the DeepLesion dataset with the same anchor configuration as we use for this dataset. However it is not possible to reuse all of the weights as for the DeepLesion dataset we only used 3 or 9 input channels and currently we are using a total of 27 channels, 9 for each of the three MRI types. This means that the first convolution has to be randomly initialized however all of the other weights can be reused. In figure 5.7 we can see the mAP when we use random and pre-trained weights.

From the figures we can see that we are able to achieve a higher mAP when we use weights from a model trained on the DeepLesion dataset. Another benefit is that it takes fewer iterations to train the model, it took 19 iterations to train without pretrained weights and only 12 iterations with pretrained weights resulting in a 37% reduction in training time. This demonstrates that even though the CT and MRI scans are completely different the information learned by the model is still useful for other tasks.

5.7.3 Ablation study

First it is important to establish the evaluation metric used. For 2D images it is common to use IoU of 0.5 when bounding boxes are compared. For us this decision isn't as straight

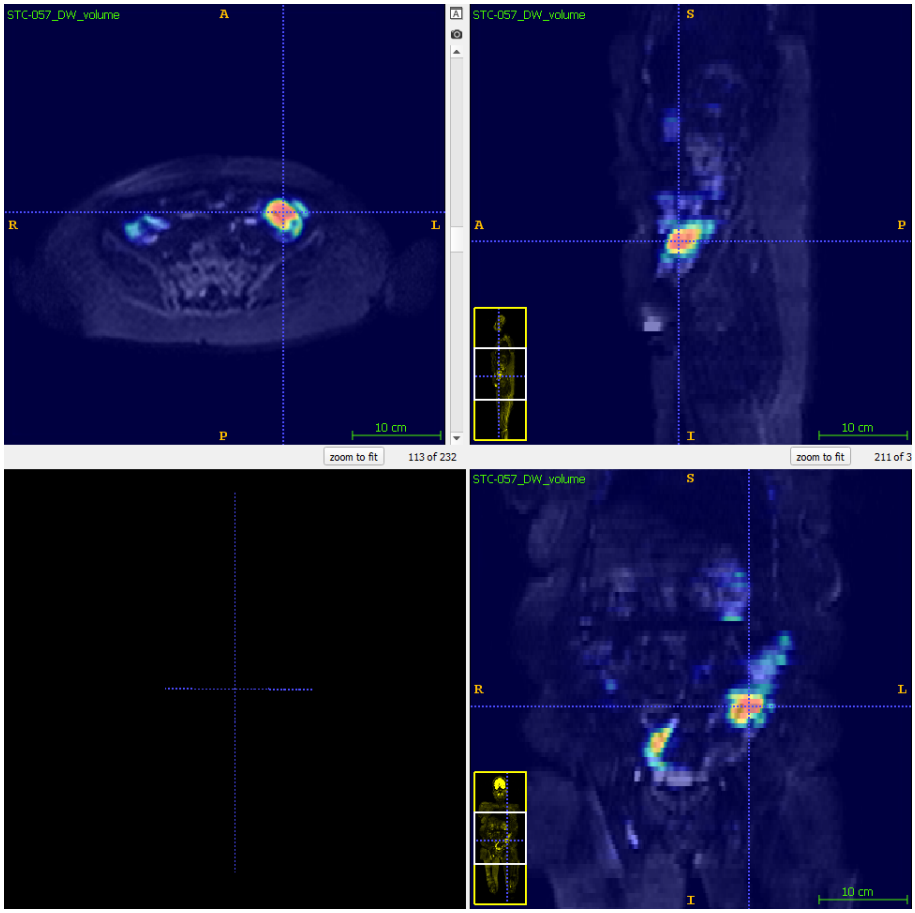


Figure 5.6: The predicted heatmap overlaid on a DW scan. Areas with red are areas with a high probability of a lesion and blue areas have low probability.

Table 5.1: Detection performance of different scan preprocessing methods.

Methods	0.5	1	2	4
Value clipping	46	63	70	70
`StandardScaler`	66	70	73	73

forward because we generate both segmentation masks and bounding boxes. In addition to this we find it more important to get an approximate location with high confidence instead of getting a perfect segmentation. We will consider a lesion as detected if it has a DICE score of 0.2 with our prediction.

The first modification which we evaluate is how we preprocess the intensities in the scans. Initially we used value clipping and then rescaled the clipped values between -1 and 1. The second method uses `StandardScaler`[1] implemented in `sklearn`. It rescales the intensities so that they have a mean of 0 and standard deviation of 1. Both of these have been explained in further detail in section 5.2. The results of this experiment can be seen in table 5.1. From the results it is visible that it is much better to use the `StandardScaler` as it achieves better results, mainly with a low number of FPs.

To make the dataset more diverse we experiment with elastic deformation applied to the original MRI scans before we create 2D slices. The aim of this is to reduces overfitting and

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

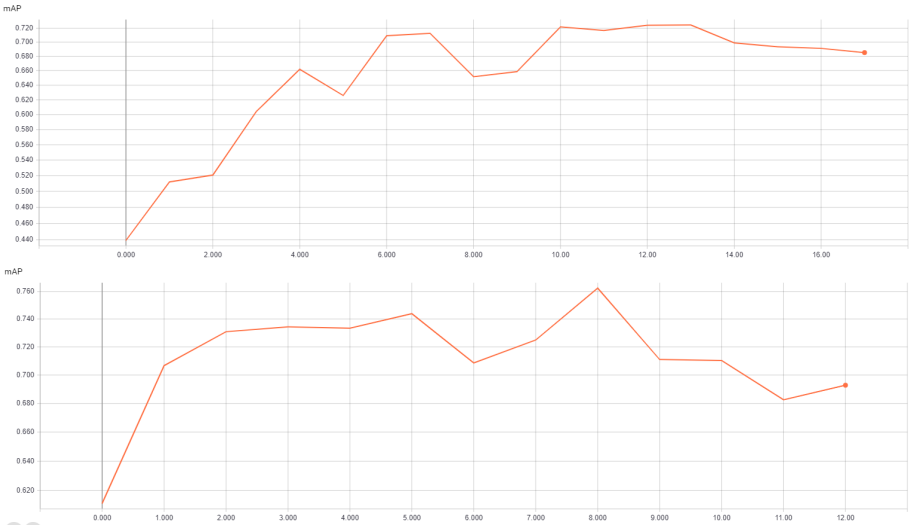


Figure 5.7: Plot of the mean Average Precision (mAP) achieved during training. The top image shows the mAP during training when no weights are used. The bottom image shows the mAP during training when pre-trained weights are used.

Table 5.2: Detection performance of elastic deformation with different values of sigma and different transformation of individual training samples.

Methods	0.5	1	2	4
Sigma of 10 + No 2D transformation	43	56	66	73
Sigma of 10 + 2D transformation	66	70	80	80
Sigma of 25 + No 2D transformation	60	73	76	80
Sigma of 25 + 2D transformation	66	70	83	83

to add 3D transformation in addition to the 2D transformations which are done during training. Ronneberger et al. [41] used elastic deformation with sigma of 10 to improve the results when they introduced the U-Net model. We experimented with both sigma of 10 and 25. After we generate the 2D slices we also investigate whether further 2D augmentation such as rotation and translation (the augmentation is the same as described in section 4.2) is still useful. The results of this experiment can be seen in table 5.2.

It is very hard to directly judge how the change in sigma affects the performance however we can see that using 2D transformation improves the results and that the results with elastic deformation are better than the results in table 5.1.

In table 5.3 we can see the detection performance for each of the lesion types in the dataset. It is however hard to judge the performance as there are only 10 lung lesions and 20 colorectal lesions in the test set. It would be beneficial if cross-validation was used however this would greatly increase the amount of time required to train and validate our results.

In practice is more important to find the approximate location of the lesion instead of a perfect overlap between the ground truth and the semantic segmentation. We experiment with another evaluation metric which instead measures the distance between the centroid of the prediction and the ground truth. If the centroid is within the ground truth then the distance is considered as 0, otherwise its the distance to the closest voxel which is part of the lesion. The sensitivity (TPR), precision (PPV) and F1 scores can be seen in figure 5.8. From the figure we can see that even at 2 predictions per image on average we can find more than 80% of the lesions within 4cm from our prediction and with 4 predictions

Table 5.3: Detection performance split by lesion type.

lesion type	0.5	1	2	4
Colorectal	60	60	80	80
Lung	80	90	90	90

per image we can find all of the lesions.

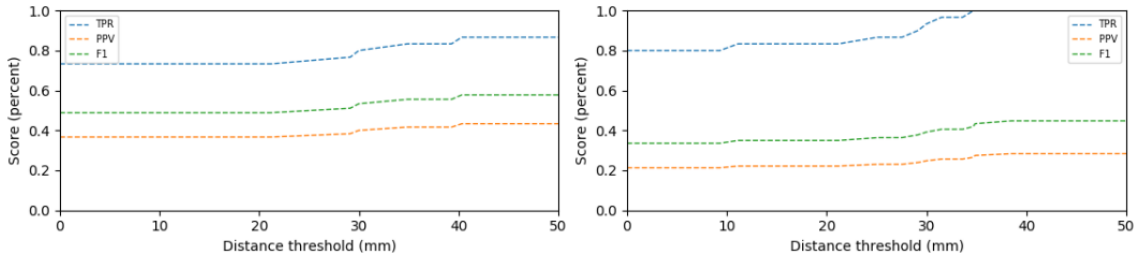


Figure 5.8: Sensitivity (TPR), Precision (PPV) and F1 score of our model for different ground truth-detection distances. In the left image there are 2 predictions per image on average, in the right image there are 4 predictions per image on average.

Chapter 6

Spot-the-Lesion Game

To demonstrate our results and introduce people to medical imaging, we have built a small browser game where the user has to locate a lesion in a CT scan. This tool is not aimed at doctors as it is relatively simplistic, it only allows the users to click on a single image without viewing neighbouring slices, seeing the CT scan along all axes and modifying the HU window. The website has been designed for both PC and larger touch screens with responsive design and simple controls.

In the game, the user is shown a slice from a CT scan which contains only one lesion. The user has 10 seconds to find and click on the lesion. This task is difficult so to help the user a circle will appear after 5 seconds indicating a smaller area where the lesion can be found. After the player runs out of time, the ground truth, as well as the prediction from our model, appears. To make the game fair, we only use the most probable prediction from our model, even if there might be multiple. In figure 6.1 we can see the game screen which the user can see. In figure 6.2 we can see the red circle which narrows down the search area, the ground truth and the prediction from our model.

We use our model as an opponent. Initially, the aim was to deploy the model and send requests to it live. However, continuously running it a server with a GPU would be costly and just running inference on a CPU would be slow as our model is very large. Instead, we attempted to deploy it on a serverless platform called Algorithmia[1] which is built purposefully for deploying machine learning models and allows us to run them on a GPU. During testing we saw that it takes between 60 and 120 seconds to complete the requests and even subsequent requests are equally slow so it seems that the container which runs the model does not stay warm between runs so it would not be possible to warm it up when a user visits the website.

To avoid having to wait so long for a prediction, we pre-compute the results for all samples and then request the result with the image which will be shown to the user. To keep the game exciting and to avoid repetition of the same images we use all of the test slices which contain a single lesion, this gives us a total of 4817 samples.

The CT images which are shown to the user are chosen randomly; however, during user testing we found this to be a disadvantage because some of the samples were noisy or the lesion was too hard to find. To avoid this, we give the user the ability to provide a random seed so that a predictable sequence of samples is generated. This change makes it easier to demonstrate the website and allows multiple people to compete with each other while being shown the same examples.

[1] https://algorithmia.com/

Figure 6.1: The main view of the website. There are instructions on how to play the game, on the right you can see how well you do compare to our trained model and in the center, you can see the current CT slice.

Figure 6.2: The view of the website after time runs out. The red circle indicates a smaller area where the lesion can be found. The yellow rectangle is the ground truth and the green rectangle is the prediction from our model.

Chapter 7

Conclusion and Future Work

We have successfully redesigned RetinaNet to be used for medical imaging and achieved state-of-the-art results on two datasets showing that our approach is generalisable.

This was done by first designing an approach which can be used to automatically optimise RetinaNet anchor configuration without having to use many resources on training the network. This greatly improves iteration speed and reduces the number of variables which have to be manually optimised. To make better use of the annotated data, we generated dense masks from weak RECIST labels which are routinely produced in clinical practice. This allows full use of the available data and improves the accuracy without having a large impact on training or inference times. Further, we have also integrated an attention mechanism into the network, which has improved the sensitivity mainly at low number of false positives.

To validate our results we have used two datasets. The first one being the DeepLesion dataset, where we achieve a sensitivity of 90.77% at 4 false positives per image, significantly outperforming the best reported methods by over 5%. We also used a much smaller whole-body MRI dataset to demonstrate that the model can be applied to 3D datasets with different modalities and still achieve excellent results.

Overall, the results show that we have indeed built a universal lesion detector. Although the results significantly outperform the best reported methods, there is still a lot of area for further research before this work can be used in the industry.

7.1 Future Work

Although we have been able to achieve excellent results and evaluated our model on two different datasets, there is still a lot of room for future work. The primary focus would be on improving the datasets which are used for training. The second opportunity for future work is to improve the model design itself, because to achieve state-of-the-art results, medical imaging requires modification to existing models, as we have shown. The final area for extension is to do a more in-depth evaluation which would involve clinicians, in order to achieve a better understanding of how well our approach works when used in industry.

7.1.1 Dataset cleaning and augmentation

When we used the DeepLesion dataset, we saw that it contained noisy CT scans. Some of these were labelled as noisy and could be ignored during training, validation and testing. However, due to the complexity and expertise required when labelling the training data, there may still be incorrectly labelled scans. When we used the whole-body MRI dataset, we could see random noise in the ground truth labels or two annotations which were disjoint

and made it seem as though the scan contained two lesions instead of a single larger lesion. If noisy data is used during training it will add noise to the results and reduce the accuracy of the final model. The scans themselves can be very noisy, which makes it harder to achieve good results. This is why we briefly experimented with denoising to improve the results. It would be beneficial to do more in-depth study and investigate the best approaches to improve the performance of our model.

It is also crucial to use data augmentation to achieve good results when the datasets are very small. Throughout this research we used the standard augmentation with default parameters, which is also used on 2D images and only briefly experimented with 3D elastic deformation. To improve, all of the augmentation should be extended to the 3D space so that rotation and shearing is applied over all three axes; the parameters used should be optimised for our use case and new methods of data augmentation should be used.

7.1.2 Model Design

Although the original RetinaNet model has been optimised to achieve better results on medical images, most of the optimisation has been done while testing it on the DeepLesion dataset. The dataset is made up of CT scans which are 3D; however, in each case, we were only interested in running the predictions on a single slice, which reduced the need for extensively using 3D context. It might be interesting to study how the current model can be augmented to contain more 3D context; one example is to only use 3D convolutions throughout the network. Another approach would be to use a Convolutional LSTM at the output of the network so that the outputs from neighbouring slices are used. Xiaobo et al. [27] proposed Recurrent RetinaNet and has shown how it can be used in videos to keep more consistency in predictions between different frames. A similar approach could be applied to CT and MRI scans to ensure that a lesion is not predicted only in a single slice. A different strategy could be to then use a secondary 3D network which looks at a smaller patch but uses 3D convolutions to decide if the patch contains a lesion or not.

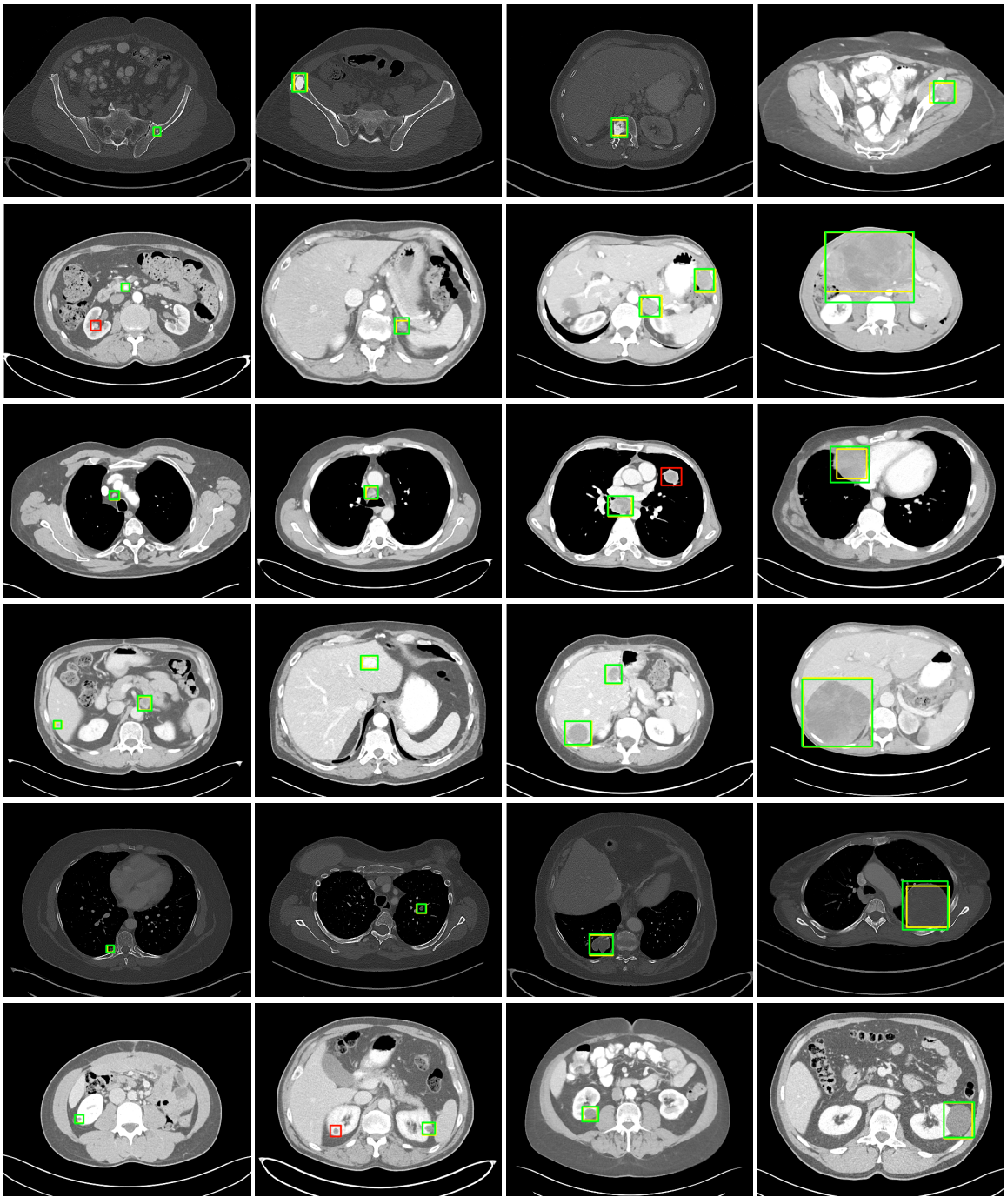
We were able to achieve very good sensitivity at the cost of having more false positives and lower precision. Most of the research focuses on achieving very high sensitivity; however, future work should focus on improving the sensitivity with only 0.5 or 1 FP per image.

7.1.3 Extensive evaluation

We have extensively evaluated the model on both the DeepLesion and whole-body MRI dataset, compared it to other research, evaluated the visual results and performed an ablation study. This gave us confidence that the model can achieve state-of-art-results on the test set, but we were not able to compare our results to the results which would be achieved by clinicians on the same dataset. Although the model might not yet be better than a doctor, it might improve the performance of doctors when they get access to the output of our model. We have shown that the model achieves very good sensitivity, which means that only a few lesions are missed. Paired with the doctors' ability to identify false positives the final performance may be better compared to human-only performance. Studying this in a clinical trial would be very beneficial as there are not many studies which compare computer-aided detection to human-only performance on a publicly available dataset.

Appendix A

Visual results for CT lesion detection



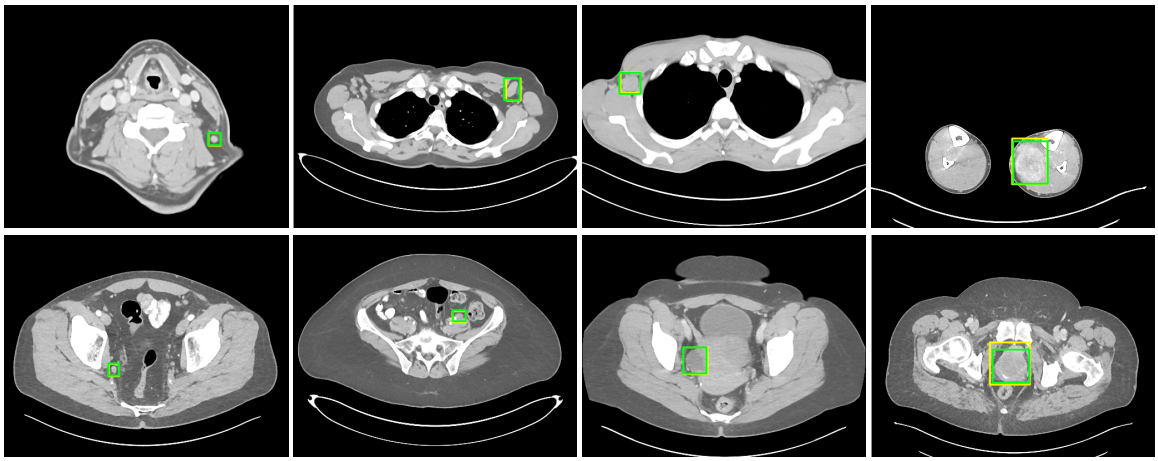


Figure A.2: Visual results for lesion detection at 0.5 FP rate using our improved RetinaNet. The rows correspond to bone, abdomen, mediastinum, liver, lung, kidney, soft tissue, and pelvis lesions, respectively. Each row contains examples of lesions of different sizes ordered from smallest to largest. Yellow boxes are ground truth, green are true positives, red are false positives.

Appendix B

Visual results for MRI lesion detection

Figures B.1, B.2 and B.3 show the ground truth, a correct and very accurate prediction and then one false positive prediction of a lung tumour however the model is not constrained to only the lungs and the false positive prediction is a colorectal prediction.

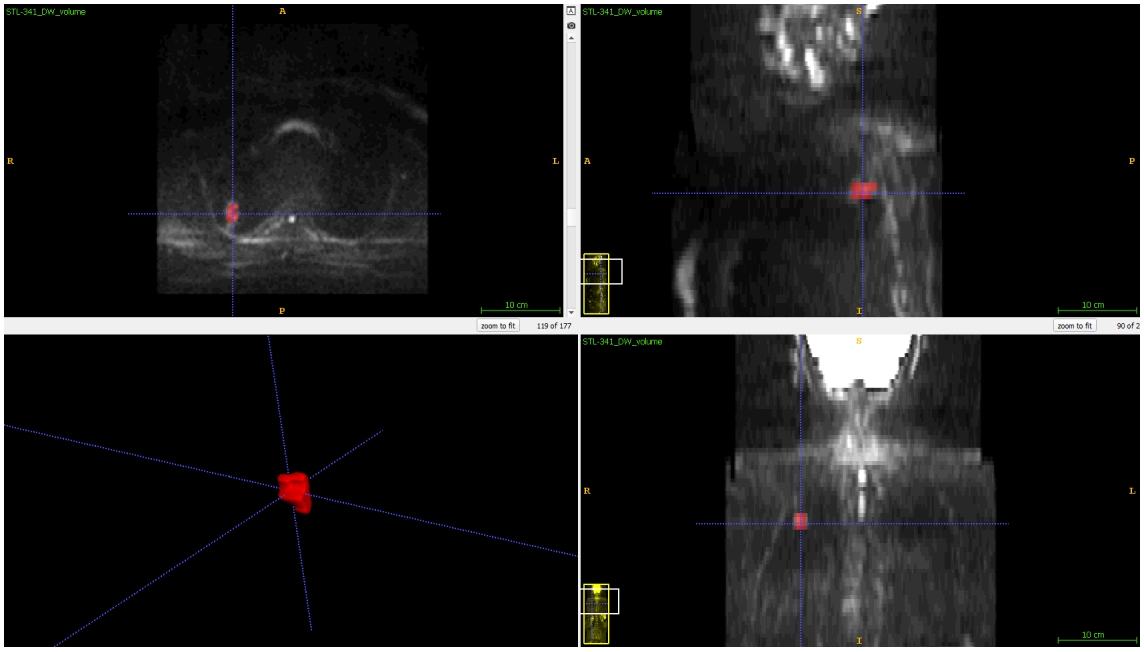


Figure B.1: The ground truth segmentation overlaid on a DW scan. Bottom left view shows a 3D render of the tumour and the other show 2D slices of the MRI.

Figures B.4 and B.5 show the ground truth and a correct prediction of a colorectal tumour. We can see that although there are bits of the tumour not covered in our prediction compared to the ground truth the centroid of the prediction is correct.

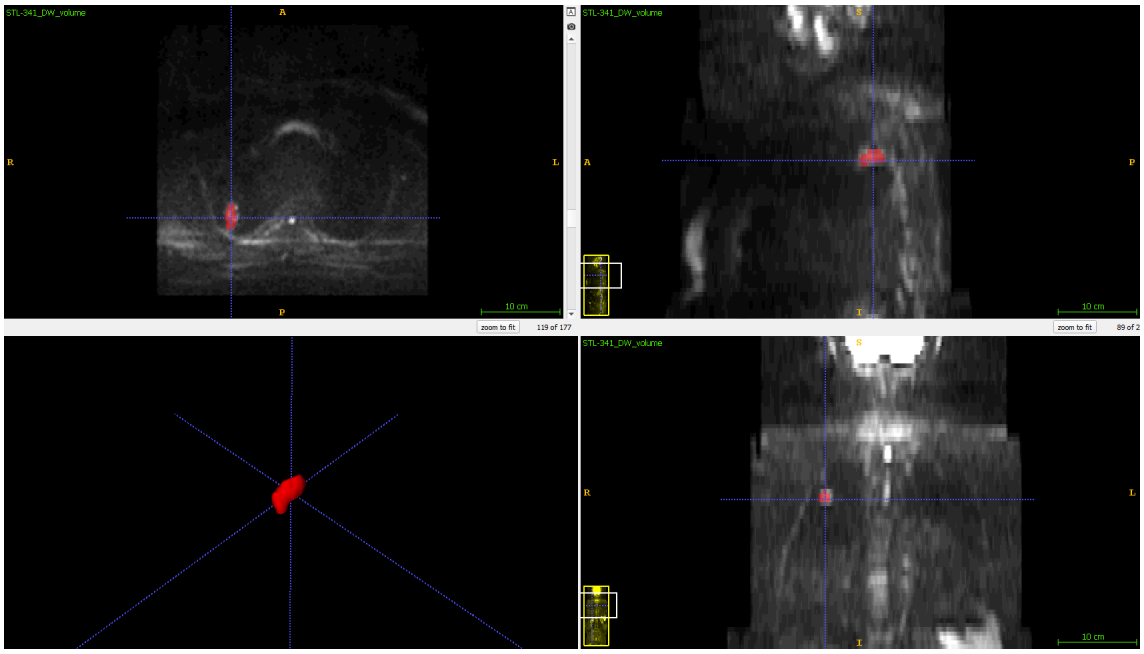


Figure B.2: Very accurate predicted segmentation overlaid on a DW scan. Bottom left view shows a 3D render of the tumour and the other show 2D slices of the MRI.

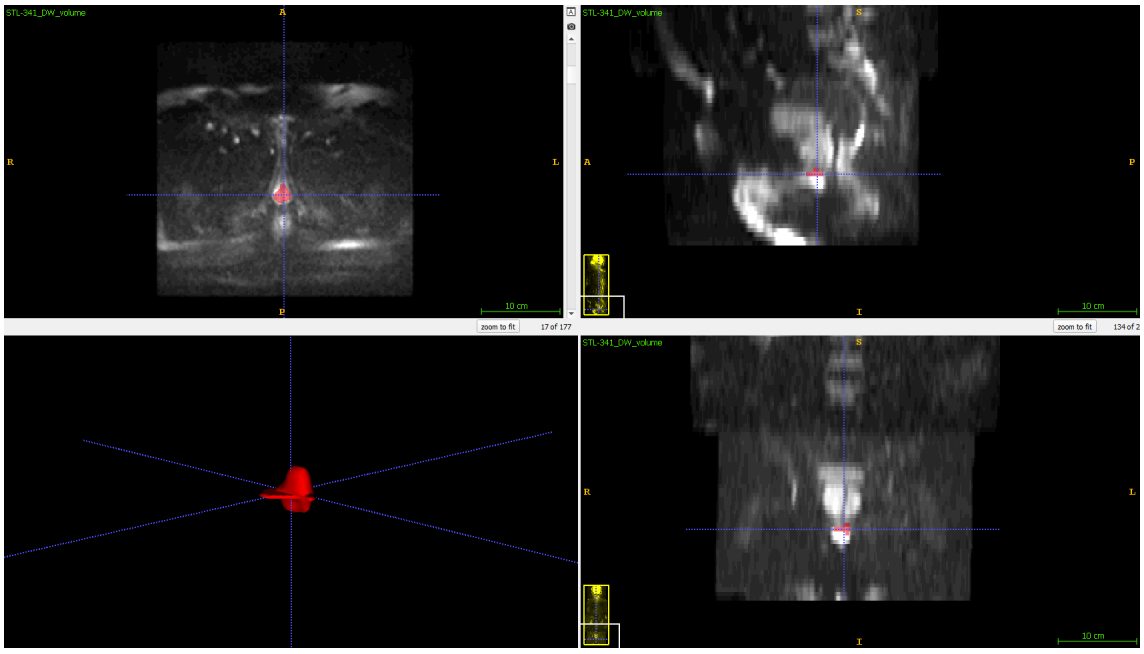


Figure B.3: A falsely predicted segmentation overlaid on a DW scan. The shape of the prediction is very strange as one of the slices has much larger surface area than the others, these large discrepancies could help a second network to further remove false positives. Bottom left view shows a 3D render of the tumour and the other show 2D slices of the MRI.

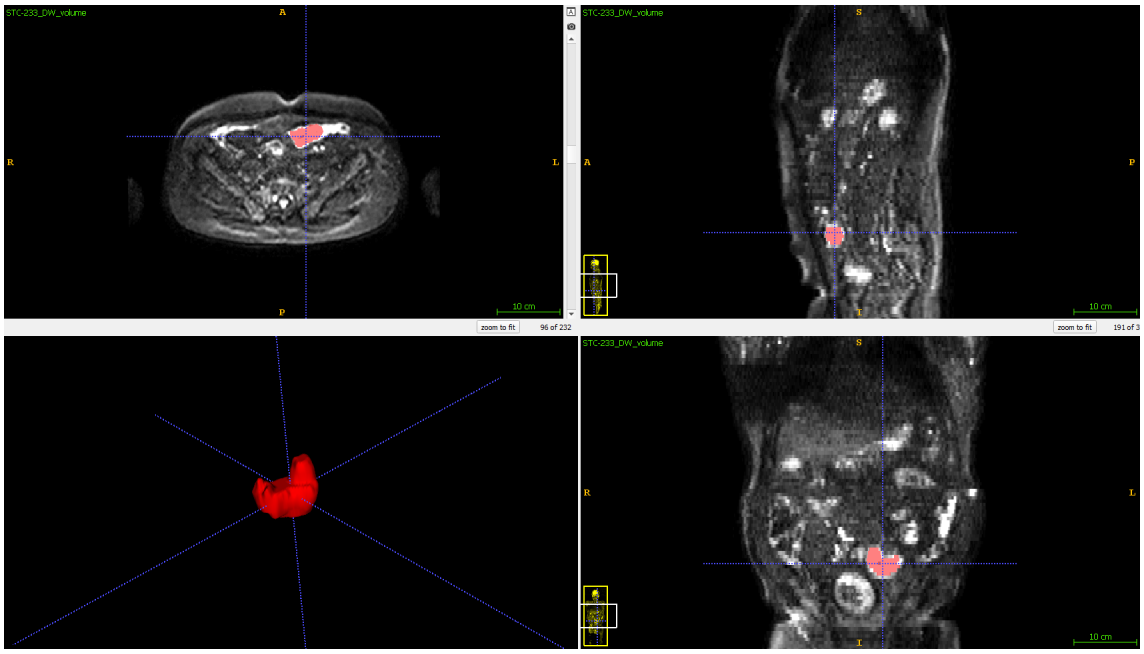


Figure B.4: The grorund truth segmentation overlaid on a DW scan. Bottom left view shows a 3D render of the tumour and the other show 2D slices of the MRI.

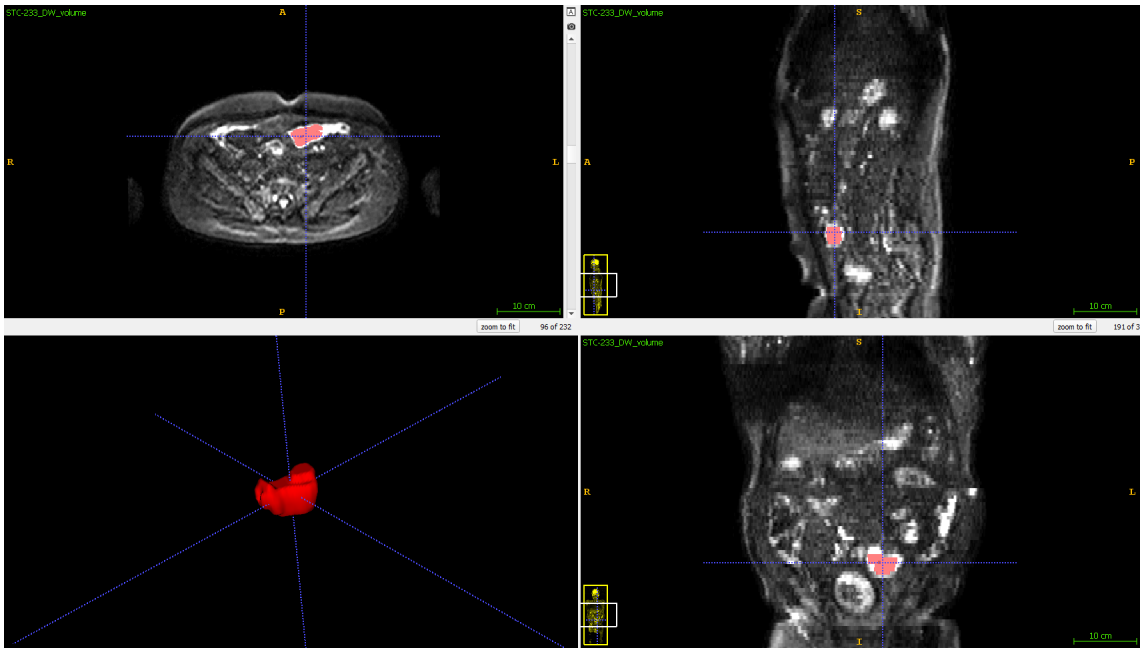


Figure B.5: The predicted segmentation overlaid on a DW scan. Bottom left view shows a 3D render of the tumour and the other show 2D slices of the MRI.

Bibliography

[1] M. Bennett, "Avoidable mortality in the UK: 2016," Apr 2018. [Online]. Available: https://www.ons.gov.uk/peoplepopulationandcommunity/healthandsocialcare/causesofdeath/bulletins/avoidablemortalityinenglandandwales/2016

[2] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[3] J. Cai, Y. Tang, L. Lu, L. Yang, R. M. Summers, and et al., "Accurate weakly-supervised deep lesion segmentation using large-scale clinical annotations: Slice-propagated 3d mask generation from 2d recist," in *MICCAI*, 2018, pp. 396–404.

[4] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[5] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: object detection via region-based fully convolutional networks," *CoRR*, vol. abs/1605.06409, 2016. [Online]. Available: http://arxiv.org/abs/1605.06409

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, June 2005, pp. 886–893 vol. 1.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[8] E. A. Eisenhauer, P. Therasse, J. Bogaerts, L. H. Schwartz, D. Sargent, R. Ford, J. Dancey, S. Arbuck, S. Gwyther, M. Mooney *et al.*, "New response evaluation criteria in solid tumours: revised RECIST guideline (version 1.1)," *European journal of cancer*, vol. 45, no. 2, pp. 228–247, 2009.

[9] W. Foundation, "Object detection," Nov 2018. [Online]. Available: https://en.wikipedia.org/wiki/Object_detection

[10] ——, "Hounsfield scale," Jan 2019. [Online]. Available: https://en.wikipedia.org/wiki/Hounsfield_scale

[11] E. Garyfallidis, M. Brett, B. Amirbekian, A. Rokem, S. Van Der Walt, M. Descoteaux, and I. Nimmo-Smith, "Dipy, a library for the analysis of diffusion MRI data," *Frontiers in neuroinformatics*, vol. 8, p. 8, 2014.

[12] S. Gaur, N. Lay, S. A. Harmon, S. Doddakashi, S. Mehralivand, B. Argun, T. Barrett, S. Bednarova, R. Girometti, E. Karaarslan *et al.*, "Can computer-aided diagnosis assist in the identification of prostate cancer on prostate MRI? a multi-center, multi-reader investigation," *Oncotarget*, vol. 9, no. 73, p. 33804, 2018.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-Based Convolutional Networks for Accurate Object Detection and Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142–158, Jan 2016.

[14] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015. [Online]. Available: http://arxiv.org/abs/1504.08083

[15] GOV.UK, "Chapter 2: trends in mortality," Sept 2018. [Online]. Available: https://www.gov.uk/government/publications/health-profile-for-england-2018/chapter-2-trends-in-mortality

[16] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: http://arxiv.org/abs/1703.06870

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[18] R. Hossain, C. C. Wu, P. M. de Groot, B. W. Carter, M. D. Gilman, and G. F. Abbott, "Missed Lung Cancer," *Radiologic Clinics of North America*, vol. 56, no. 3, pp. 365 – 375, 2018, imaging of Lung Cancer: Update on Staging and Therapy. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0033838918300046

[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[20] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993

[21] P. F. Jaeger, S. A. A. Kohl, S. Bickelhaupt, F. Isensee, T. A. Kuder, H. Schlemmer, and K. H. Maier-Hein, "Retina U-Net: Embarrassingly Simple Exploitation of Segmentation Supervision for Medical Object Detection," *CoRR*, vol. abs/1811.08661, 2018. [Online]. Available: http://arxiv.org/abs/1811.08661

[22] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online]. Available: http://www.scipy.org/

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

[24] H. Kundel, C. Nodine, and D. Carmody, "Visual scanning, pattern recognition and decision-making in pulmonary nodule detection," *Investigative radiology*, vol. 13, no. 3, p. 175—181, 1978. [Online]. Available: http://europepmc.org/abstract/MED/711391

[25] I. Lavdas, B. Glocker, D. Rueckert, S. Taylor, E. Aboagye, and A. Rockall, "Machine learning in whole-body MRI: experiences and challenges from an applied study using multicentre data," *Clinical Radiology*, vol. 74, no. 5, pp. 346 – 356, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0009926019300741

[26] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2Noise: Learning Image Restoration without Clean Data," *CoRR*, vol. abs/1803.04189, 2018. [Online]. Available: http://arxiv.org/abs/1803.04189

[27] X. Li, H. Zhao, and L. Zhang, "Recurrent retinanet: A video object detection model based on focal loss," in *Neural Information Processing*, L. Cheng, A. C. S. Leung, and S. Ozawa, Eds. Cham: Springer International Publishing, 2018, pp. 499–508.

[28] Y. Li, "Detecting Lesion Bounding Ellipses With Gaussian Proposal Networks," *CoRR*, vol. abs/1902.09658, 2019. [Online]. Available: http://arxiv.org/abs/1902.09658

[29] M. Liang, W. Tang, D. M. Xu, A. C. Jirapatnakul, A. P. Reeves, C. I. Henschke, and D. Yankelevitz, "Low-dose CT screening for lung cancer: computer-aided detection of missed lung cancers," *Radiology*, vol. 281, no. 1, pp. 279–288, 2016.

[30] Lin, Goyal, Girshick, Ross, and Piotr, "Focal Loss for Dense Object Detection," Feb 2018. [Online]. Available: https://arxiv.org/abs/1708.02002

[31] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[32] Liu, Wei, Anguelov, Dragomir, Erhan, Dumitru, Scott, Cheng-Yang, Berg, A. C., and et al., "SSD: Single Shot MultiBox Detector," Dec 2016. [Online]. Available: https://arxiv.org/abs/1512.02325

[33] I. L. Lowekamp BC, Chen DT and B. D. (2013), "SimpleITK," 2013. [Online]. Available: http://www.simpleitk.org/

[34] LUNA16, "LUNA16 - Lung Nodule Analysis 2016." [Online]. Available: https://luna16.grand-challenge.org/

[35] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest *et al.*, "The multimodal brain tumor image segmentation benchmark (BRATS)," *IEEE transactions on medical imaging*, vol. 34, no. 10, pp. 1993–2024, 2014.

[36] O. Oktay, J. Schlemper, L. L. Folgoc, M. C. H. Lee, M. P. Heinrich, K. Misawa, K. Mori, S. G. McDonagh, N. Y. Hammerla, B. Kainz, B. Glocker, and D. Rueckert, "Attention U-Net: Learning Where to Look for the Pancreas," *CoRR*, vol. abs/1804.03999, 2018. [Online]. Available: http://arxiv.org/abs/1804.03999

[37] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS-W*, 2017.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[39] Redmon, Joseph, Divvala, Santosh, Girshick, Ross, Farhadi, and Ali, "You Only Look Once: Unified, Real-Time Object Detection," May 2016. [Online]. Available: https://arxiv.org/abs/1506.02640

[40] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[41] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[42] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," in *ACM transactions on graphics*, 2004, pp. 309–314.

[43] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations*, 2015.

[44] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[45] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," *CoRR*, vol. abs/1602.07261, 2016. [Online]. Available: http://arxiv.org/abs/1602.07261

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[47] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *CoRR*, vol. abs/1512.00567, 2015. [Online]. Available: http://arxiv.org/abs/1512.00567

[48] Y. Tang, K. Yan, Y. Tang, J. Liu, J. Xiao, and R. M. Summers, "ULDor: A universal lesion detector for ct scans with pseudo masks and hard negative example mining," *arXiv preprint arXiv:1901.06359*, 2019.

[49] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, "Selective Search for Object Recognition," *International Journal of Computer Vision*, 2013. [Online]. Available: http://www.huppelen.nl/publications/selectiveSearchDraft.pdf

[50] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, Dec 2001, pp. I–I.

[51] K. Yan, M. Bagheri, and R. M. Summers, "3D Context Enhanced Region-Based Convolutional Neural Network for End-to-End Lesion Detection," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, A. F. Frangi, J. A. Schnabel, C. Davatzikos, C. Alberola-López, and G. Fichtinger, Eds. Cham: Springer International Publishing, 2018, pp. 511–519.

[52] K. Yan, X. Wang, L. Lu, and R. M. Summers, "DeepLesion: automated mining of large-scale lesion annotations and universal lesion detection with deep learning," *Journal of Medical Imaging*, vol. 5, pp. 5 – 5 – 11, 2018. [Online]. Available: https://doi.org/10.1117/1.JMI.5.3.036501

[53] K. Yan, X. Wang, L. Lu, L. Zhang, A. P. Harrison, M. Bagheri, and R. M. Summers, "Deep lesion graphs in the wild: relationship learning and organization of significant radiology image findings in a diverse large-scale lesion database," in *CVPR*, 2018.

[54] M. Zlocha, Q. Dou, and B. Glocker, "Improving RetinaNet for CT Lesion Detection with Dense Masks from Weak RECIST Labels," *arXiv preprint arXiv:1906.02283*, 2019.