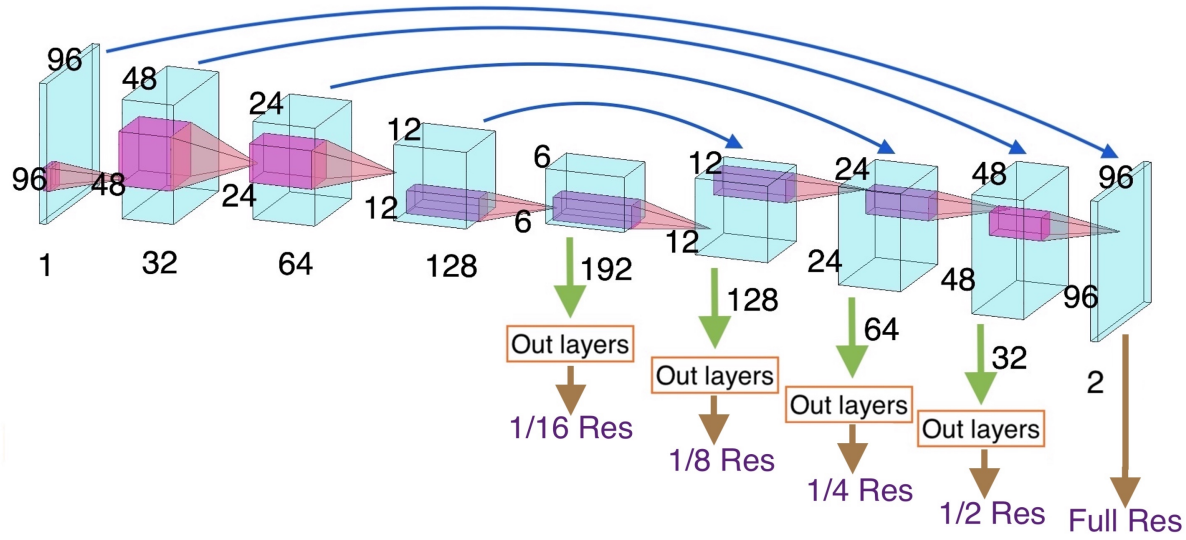
Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project Report 2022



Project Title:	**Image segmentation via multi-resolution convolutional neural networks**
Student:	**Edward Jack Louis Schamp**
CID:	**01514262**
Course:	**EEE4**
Project Supervisor:	**Dr. Wei Dai**
Second Marker:	**Dr. Ayush Bhandari**

Plagiarism Statement

I affirm that I have submitted, or will submit, an electronic copy of my final year project report to the provided EEE link.

I affirm that I have submitted, or will submit, an identical electronic copy of my final year project to the provided Blackboard module for Plagiarism checking.

I affirm that I have provided explicit references for all the material in my Final Report that is not authored by me, but is represented as my own work.

Abstract

Lung tumour segmentation has been greatly improved over recent years through the introduction of several new types of networks, including UNet. Multi-Resolution Convolutional Neural Networks are a proposed architectural improvement to the UNet architecture, which utilise the internal layer values at different stages of the decoder to produce outputs to the network. These additional outputs contribute to the loss which the network calculates at each training step, and allow better loss convergence, leading to a higher Dice metric.

A modified medical version of UNet is tested as a baseline in this report, achieving a Dice metric of 0.4669 after training for 800 epochs on the Medical Segmentation Decathlon Lung Tumours data set. Different Multi-Resolution Convolutional Neural Network approaches are able to significantly improve upon this, achieving Dice scores of greater than 0.5, with the largest Dice score being 0.5405. This is a significant improvement over UNet, and solidifies multi-output networks as a research area which should be investigated and optimised further, such as by applying similar principles to transformer networks.

Acknowledgements

I would like to thank my supervisor, Dr. Wei Dai, for his support by clarifying any issues I had throughout this project. I would also like to thank my peers for their companionship throughout the four years of my degree.

List of Figures

List of Tables

List of Acronyms

CNN - Convolutional Neural Network

MCNN - Multi-Resolution Convolutional Neural Network

LIDC-IDRI - Lung Image Database Consortium image collection

TCIA - The Cancer Imaging Archive

NIFTI - Neuroimaging Informatics Technology Initiative

MONAI - Medical Open Network for Artificial Intelligence

CT - Computerised Tomography

MRI - Magnetic Resonance Imaging

MIT - Massachusetts Institute of Technology

Contents

Chapter 1

Introduction

1.1 Project motivation

Lung cancer has been the most lethal type of cancer in the United Kingdom ever since statistics in this domain were first recorded [27]. This disease accounts for 21% of cancer-related deaths in the U.K.. This statistic is relatively equal in both males and females, with slightly more male deaths, as is usually the case for deaths by cancer. When considering the deadliest cancers, lung cancer is followed by bowel cancer, which accounts for a much lower percentage of total cancer-related deaths (10%). These statistics are visualised in Figure 1.1.

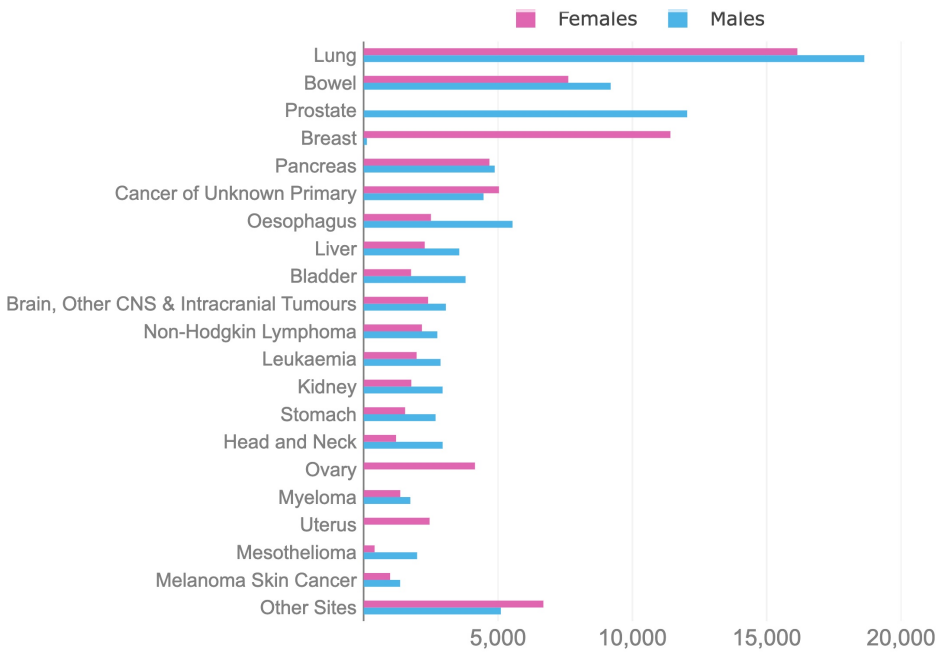


Figure 1.1: A breakdown of U.K. cancer deaths by cancer type and patient gender [27]

From Figure 1.1, it can be observed that lung cancer is more than twice as deadly as any other kind of cancer. Because of this, academic contributions towards reducing the

mortality rate of this disease will potentially be able to prevent millions of deaths, having a significant positive impact on society. Using current methods, experts estimate that the early detection of lung cancer can decrease the death rate of the illness by 14 to 20 percent [3]. The most common method of lung cancer detection is to use CT scanning equipment. This has become the de-facto screening method for lung tumours, and involves the compilation of many horizontal cross-sectional slices of the body over the vertical range which spans the area of the lungs.

The dataset that will be used contains CT scans of the lungs, along with labeled images indicating the areas of the lungs which contain a tumour. The labeled images will be referred to as ground truths or labels in this report, and the lung scans as images. Figure 1.2 gives an introduction into the dataset that will be used for this project, which uses data taken from the LIDC-IDRI dataset, and will be further explained later in the report.

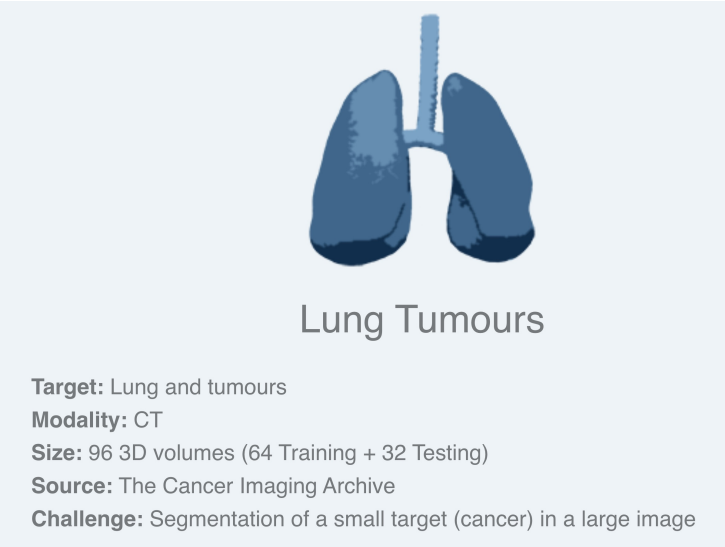


Figure 1.2: Lung tumour segmentation dataset brief description

In CT scans, two-dimensional image 'slices' are taken of horizontal cross-sections of the lungs. Multiple cross-sectional slices are taken, with a vertical separation of a few millimeters between them. All image slices which have been taken can be used to obtain a complete 3D picture of the composition of the lungs. A 'voxel' is the distance between consecutive pixels in the x, y and z-axes, and gives the resolution of a 3D MRI or CT image. A voxel can be thought of as a 3D pixel to aid conceptualisation. The 3D image that is constructed from this provides detailed information into the shape and location of different tissues in the body, including organ tissue, bones, and tumours. Segmentation of tumours from other types of tissue will be focused on in this report.

1.2 Project specification

The aim of this project is to be able to test and provide results for different neural network methods to detect lung tumours. The performance of each implementation will be evaluated after it has been tested. A given implementation will demonstrate an improvement over a previous implementation if it either achieves the same accuracy but in a smaller amount of time, or achieves an improved accuracy. The achievement of either of these metrics, or both, will indicate a useful contribution towards the field of lung tumour detection.

During neural network testing stages, the overall accuracy of a network can be inferred by the value of the loss function which is output at each training epoch. A lower loss function value can usually lead to a higher metric value. However, it is the value of the metric that determines the overall performance of a model, not the loss.

Throughout this project, a class of auto-encoder convolutional neural networks called Multi-Resolution Convolutional Neural Networks (MCNNs) will be used. These are a kind of convolutional neural network, which have previously only been designed to operate on 2-dimensional images. These types of networks consist of different numbers of convolutional 'layers', with each layer taking in a certain input dimension, and producing an output with a certain output dimension. An explanation of convolutional neural networks is given in the Background Section of this report, allowing the reader to understand the adaptations which are needed to be made to a convolutional network in order to use it for lung tumour detection.

1.3 Report structure

This report will assume that the reader has knowledge of basic mathematical principles, and has an idea of how back-propagation and neural network training work. Concepts which are specific to this report, such as the data used to train the network, will be covered in the *Background* Section of the report. Neural network architectures and loss functions that are used in implementations in this report will also be explained in the Background section of the report. If the reader has a good understanding of the UNet architecture, Sections 2.4 to 2.8 may be skipped. Following the convolutional neural network Background sections, the multi-resolution convolutional neural network architecture is explained in detail, along with any important insights that can be used from prior work on MCNNs.

Once the reader has an understanding of the essential concepts used in this report, specific requirements for the project are outlined in the *Requirements Capture* Section. The purpose of the remaining sections of the report is to fulfil the requirements listed here.

The *Analysis and Design* Section outlines how choices were made from the conception of an idea to its implementation, explaining the advantages and disadvantages of different potential methods. This section lists the main difficulties that were anticipated for this project, and how these were tackled by sensible design choices. The final design choices were made after many initial design attempts which did not produce the intended results, and some of these initial designs are explained in this section too.

The *Implementation* Section describes all important design features in the final implementation of the project, along with explanations for each decision. This is followed by the *Testing* Section, which explains the process of determining which design variations performed better. Using the knowledge gained from testing, the final networks which provide the final results for this report are given in the *Final Results* Section.

The *Evaluation* Section explains the meaningfulness of the results obtained, and is followed by the *Conclusions and Further Work* Section, which suggests elements which were not explored in this work, which may be interesting to explore in the future.

Chapter 2

Background

2.1 The Evolution of Digital Image processing

Digital image processing describes the act of altering a digital image in some way to produce a desired outcome. The first notable applications of digital image processing were arguably performed in the 1960s in the United States at Bell Labs, MIT and some other notable institutions [21]. Ever since the inception of digital image processing, one of the main focuses of its applications has been on medical images. This is due to the fact that medical scans such as CT scans or X-rays must be analysed upon inspection by healthcare professionals, introducing human error into the process. Image processing algorithms which can identify some elements of patient scans allow the degree to which human error affects diagnoses to be reduced. Furthermore, the medical industry is one of few industrial with a direct impact on people's survival, making it a vital part of society.

2.2 Medical Imaging Datasets

2.2.1 Data usage regulations

Many medical imaging datasets exist online, and many of these are open source. One such dataset is from the Cancer Imaging Archives, and is named the Lung Image Database Consortium image collection (LIDC-IDRI) dataset. The official Cancer Imaging Archive (TCIA) website states "TCIA encourages the community to publish your analyses of our datasets" [11]. LIDC-IDRI is available for download under the Creative Commons Attribution 3.0 Unported License [13]. There are data usage policies and restrictions to using this dataset [12], which include:

1. Not using the data to find patient identities
2. Acknowledging the dataset in all publications of work

3. Reviewing their policies before hosting your own online copy of the dataset

To address the third bullet point, the data will be hosted in a private Google Drive folder, and will not be made public. The first and second points will be strictly adhered to throughout this project, with the dataset being acknowledged here, within the final report for this project.

2.2.2 Hardware Considerations

The LIDC-IDRI dataset, in its original form, includes 1010 CT scans. While it is possible to train networks on this data, it would require a powerful and expensive GPU, along with needing a training time of approximately a week or more. The viability of using multiple custom RTX6000 GPUs was looked into, and the cost of this was estimated to be approximately £3000, which would only be worth it if the project goal was to produce an optimally trained model which would be deployed in hospitals. However, since at the start of this project, the networks which will be investigated had not been tested for the purpose of lung tumour segmentation, this project is clearly more experimental in nature. Having said this, using the results of this project, the reader is free to train the models described on the full dataset for optimal performance.

2.2.3 The Medical Segmentation Decathlon Lung Tumour dataset

A much more suitable portion of the LIDC-IDRI data which is used in competitions globally is called the *Medical Segmentation Decathlon Lung Tumour dataset*. This dataset was mentioned in the introduction to this report, and contains 64 training CT scans, along with 32 testing CT scans. The CT scans are provided in the NIFTI file type, which has the file extension '.nii.gz'. This file type was determined to be the de-facto file type for MRI and CT scans, to ensure uniformity across all different scanner manufacturers. This makes it easier to impose minimum industry standards for the quality and information contained within these types of medical images, and also improves the experience of data processing for researchers wishing to process the data.

2.2.4 The NIFTI file format

All NIFTI files contain a header, which contains approximately 50 pieces of data, and a size of 348 bytes. Most of this data does not need to be covered in this report, but the interested reader can go to the *Brainder* website [24] for a detailed description of all of the information contained within the header. The main items of interest that can be accessed through the header are listed below:

1. Image dimensions
2. Data type (e.g. signed int)
3. Orientation information
4. Image Offset

The orientation information refers to whether all slices were taken from the same angle, or if there are some slices taken from a different angle to aid visualisation of specific areas of interest. Often, this is expressed as a simple transformation such as a degree of rotation, however this is also expressed in an Affine matrix, which contains information on the transform which can be applied to the image to bring it to a standard alignment (which may differ between different scans). The image offset is the number of pixels that may exist within an image before the scan starts. It can be thought of as a border to the image, padding it slightly, and sometimes added so that the image dimensions are equal in the x and y axes. Images can be cropped to no longer contain the offset if this is desired. However, even if cropping is possible, it is often undesirable. An example of this is when implementing a Multi-Resolution Convolutional Neural Network (MCNN). Here, image dimensions which are a power of 2 are required, in order to successively reduce the resolution of an image by factors of 2.

The image dimensions allow the input dimensions for a neural network to be specified. This may be changed through data preprocessing, as is done throughout this report. However, knowledge of the original input spacial dimensions remains important. The data type allows for the selection of an appropriate learning rate, and output activation function, as well as being useful in choosing a loss function.

The label data contains zeros where there is no tumour, and ones where a tumour exists. Hence, the network output should be able to output values ranging from 0 to 1. Possible final layer activation functions include *Sigmoid* and *Softmax* to accomplish this.

2.2.5 Data dimensions

Lung CT scan data is 3-dimensional data, meaning that each CT scan consists of multiple images with x and y axis dimensions. The images are of resolution (512,512) and the number of slices ranges from 112 to 687 in the training data. Since the z-dimension is not fixed, this poses a challenge for neural network implementations. Neural networks must operate on a fixed input size, since the size of the data through all of the layers dictates the number of weights that the network has between each of the layers. This is often solved by batching the data, and will need to be experimented with throughout this project.

When propagating through a neural network with 2-dimensional input data, the representations will be 3-dimensional. The extra dimension is added for the network channels, which will be explained in the background section named Convolution. Since the input image is grey-scale, only a singular channel is required, and the input image dimensions could be (512, 512, 1) where a grey-scale image with resolution (512,512) is the input to the network.

Likewise, for 3 dimensional CT scan inputs, the neural network data representation will be 4-dimensional. Input image dimensions could be (351, 512, 512, 1) where 351 grey-scale images with resolutions (512,512) each are the input to the network.

2.3 Lung tumour stages and sizes

The average volume of the lungs in a human is approximately 6 liters. This corresponds to 6000 cm^3, and makes it the largest bodily organ by volume except for the skin. Because CT scans take slices over a volume, the large volume that needs to be covered when detecting tumours in the lungs may pose challenges for tumour detection and segmentation. This is because many slices will need to be taken, but the tumour will only exist in a few slices.

Lung tumours which are less than 4cm in diameter (or 4*4*4 = 64 cm^3 in volume) are classified as class I tumours. These tumours occupy less than 1% of the total volume of the lungs. Stage II tumours are from 4 to 5cm in diameter, occupying from 1 to 2% of the total lung volume. Stage III, IV and V tumours are 5cm or larger, and have spread to other organs in the body. At stage III and beyond, it is recommended to get a brain CT scan, along with several other organ scans, to ensure that the cancer has not spread to these locations.

In the Medical Decathlon Lung Tumour Dataset, there are between 120 and 180 slices taken for each pair of lungs which are scanned, and on average 5 to 10 slices contain a tumour. For a CT scan with 150 slices, and 8 of them containing a tumour, this means that 95% of slices will have a mask which is entirely filled with zeros. This means

that the probability distributions between different classes (tumour vs. no tumour) is very unbalanced, and must be taken into account when considering an appropriate loss function.

The 5-year survival rate for lung cancer which is in stages I or II is 64%, whereas the survival rate at stage III is 37%, and stage IV is a mere 8%. From this, it becomes evident that it is most beneficial to concentrate on datasets which include tumours which are 5cm or less in diameter, such as the dataset which will be used in this project.

2.4 Classification, Detection and Segmentation

The most basic application of computer vision is object detection. After this comes object classification, and an even more complex application than this is image segmentation. Image segmentation aims to detect the outline of the object which is to be detected. A perfect image segmentation system would be able to define which pixels are part of the object or objects which are desired to be detected, and get this definition correct all of the time.

In practice, most image segmentation problems are almost impossible to solve perfectly. However, computer vision is approaching ever higher accuracies for many applications, especially through the use of deep learning. Another advantage of digital image processing is that computers can perform numerical calculations much quicker than humans, and as such can execute multiple different operations on images in a fraction of the time that a human could perform them in. Therefore, a perfect digital image processing system would have an accuracy that is far superior to any expert in the medical field.

The differences between image classification, object detection and image segmentation are visually displayed in Figure 2.1:

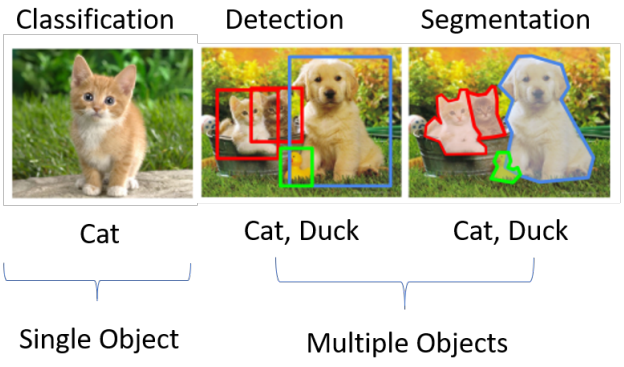


Figure 2.1: The differences between classification, detection, and segmentation

It can be observed from Figure 2.1 that the information about an image which is provided in the network output increases as one moves from classification to detection, and finally

to segmentation. The output of a classification problem is in fact very simple to express, with the number of output states needed being one greater than the number of classes (to also represent that no class has been detected). Image segmentation, on the other hand, must encode the outline of the object, pixel by pixel, and provide this as the output. This means that the accuracy of image segmentation techniques is most commonly measured using a mathematically formulated metric such the Dice index, which will be explained in Section 2.5.2.

At this point it should be noted that there are two main types of image segmentation: Semantic segmentation and instance segmentation. The difference between these two types of segmentation is that instance segmentation differentiates between multiple occurrences of the same type of object, giving each one a unique label. Semantic segmentation groups all occurrences of the same type of object into a single occurrence. In the field of tumour detection, semantic segmentation is most commonly used, since the goal is to detect all tumours, and not to only detect certain instances of tumours. This concept is illustrated in Figure 2.2.

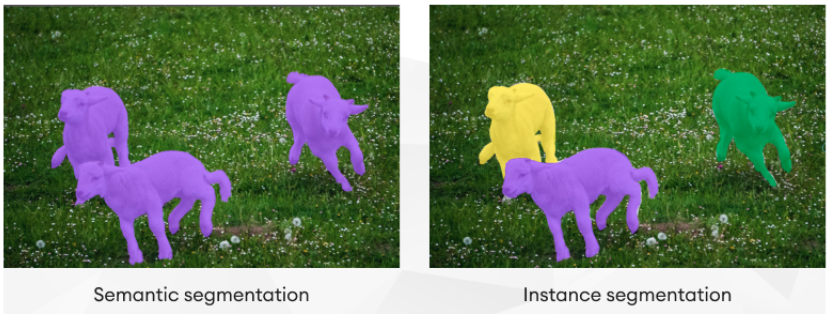


Figure 2.2: Semantic segmentation vs instance segmentation

2.5 Loss functions

2.5.1 Binary Cross-Entropy Loss

All datasets that can be used to train a supervised neural network consist of labeled data. This means that both the input images and output labels must be used in conjunction with one another to evaluate the performance of a network. The output of a neural network for any given input is compared to the desired output (the label) through means of a loss function. Loss functions are almost always constructed so that the aim is to achieve the minimum possible value, and this will be the case for all loss functions used in this report. The two loss functions that will be applied to segmentation are Binary Cross-Entropy and Dice Loss.

Cross-entropy loss functions work by measuring the similarity between two given probability distributions. The labeled data can be referred to as a 'mask', which is a zero-filled

tensor or array which consists of ones in the desired locations. In the case of lung tumours, the ones will be located in the positions where a tumour exists, and zeros will be located in all other locations.

The distributions of the output masks must be taken into account to determine the suitability of a loss function for any given task. It has been explained that the lungs are a large organ compared to the size of typical lung tumours. This means that the probability distributions of the tumour and non-tumour regions are very unbalanced. This is important to take into account when considering the suitability of loss functions, since it means that the loss function may be minimised most efficiently by fitting to non-tumour regions only. Because of this, the model may learn to output all zeros, indicating that a tumour does not exist in any of the image slices.

2.5.2 Dice Loss

The Dice loss is another loss function that will be used, which corrects for some of the shortfalls of the cross-entropy loss function. It uses similar principles to the Intersection over Union (IoU) loss function, and places a high reliance on the intersection between the predicted output region and the ground truth. By doing this, the loss function converges to a state where the intersection between the prediction and ground truth overlap more and more.

The Dice index is defined as [2]:

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|}$$

Where $|X|$ is the cardinality of X, or the number of elements in the set X. This metric measures the overlap between two sets, and scales it by the combined total number of elements in each of the sets. In other words, $|X \cap Y|$ is the intersection of the sets, and $|X| + |Y|$ is the union. So the Dice metric is twice the intersection over the union.

The Dice metric ranges in value from 0 to 1, with a higher metric meaning a better prediction. A perfect prediction would be one in which the predicted set and ground truth are identical, making two times the intersection equal to the union of the sets.

For example if X and Y contain 9 elements each, and 5 of these elements are present in both sets, the Dice index becomes:

$$DSC = \frac{2|5|}{|9| + |9|} = \frac{10}{19} \approx 0.53$$

Because loss functions are minimised, the Dice metric cannot be used as a loss function. Instead, the dice loss is defined as:

$$\text{Dice loss} = 1 - \text{Dice metric}$$

Image segmentation on medical datasets when applied to tumour detection currently achieve Dice indices from 0.4 to 0.98 depending on the dataset. This is due to the large range of tasks that are possible in medical segmentation Some datasets require the segmentation of an organ, and some require the segmentation of small tumours. Organ location varies minimally from person to person, and the same organ looks very similar from person to person too. Hence, a dataset to identify the Spleen in a person can achieve a very high Dice metric (approx. 0.9). Conversely, when identifying lung tumours, most networks will achieve Dice metrics in the range of 0.3-0.6, with only the best transformer networks able to outperform that, depending on the difficulty of the dataset.

2.6 2-Dimensional Convolutional Neural Networks

The first convolutional neural network (CNN) that attracted attention in the world of AI was LeNet5 by Yann LeCun in 1998 [4]. From this, and including some developments on it, the standard convolutional neural network architecture has been optimised to include the same standard stages. The network includes initial convolutional layers which each consist of different filters, to extract different features from the image. These filters each correspond to an output feature map, or an output channel. Convolutional layers usually slightly reduce the height and width of the data, and increase the number of channels. Following a convolutional layer, CNNs often reduce the dimension of representation of the image with a pooling layer. The final layers of the network are fully connected layers, which can be represented by convolutional layers with a filter dimension of 1x1. The elements of a CNN which were just mentioned will now be explained in a way that is most relevant to MCNNs.

2.6.1 Convolutional layers - Kernels and strides

The networks which will be used throughout this report make frequent use of convolutional layers. For MCNNs, it is important to analyse the factors which alter the spacial dimensions of the output. Two of these important factors are convolutional kernels and strides.

A convolution, in machine learning, involves the application of a filter to the data in a layer of the network. The mathematical definition of convolution is the application of a function to another function to provide an output which is the modified original function. In machine learning, the application refers to the multiplication of each element of a filter with a corresponding section of the input which is the same size as the filter, and the summation of all of the multiplied elements. The filter here is referred to as a kernel, and

the process of multiplication and summation is given in Figure 2.3. This depicts a two dimensional convolution, which is used in neural networks that process 2D images.

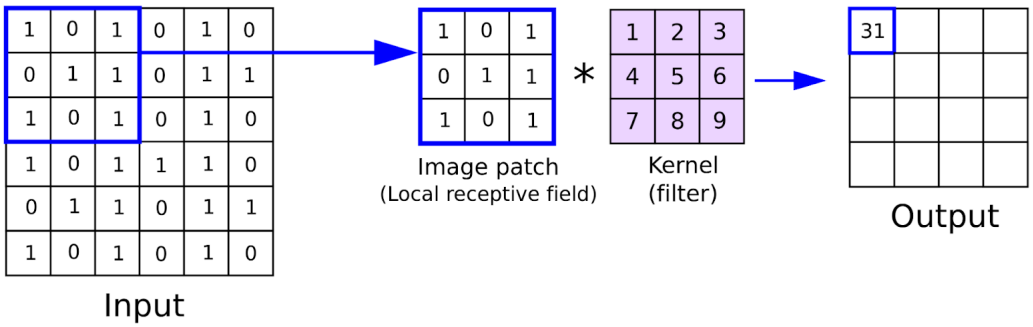


Figure 2.3: Visual depiction of convolution [32]

From Figure 2.3, it is evident that with kernel dimensions of (3, 3), the output image dimensions are equivalent to the input image dimensions, but reduced by 2 pixels in the x and y dimensions. This is because of the kernel size. The equation for the output size is a function of many factors, among which is the kernel size. The equation for output kernel size is as follows:

$$\text{output size} = \frac{\text{input volume - kernel size + 2*padding}}{\text{stride}} + 1 \tag{2.1}$$

Padding can be visualised as the addition of zeros around the complete length of the border of a convolutional layer output. The padding value gives the number of layers of zeros that are added. In the example in Figure 2.3, a padding value equal to 1 would make the output image dimensions equal to the input image dimensions.

The final parameter which affects the size of an output of a convolutional layer is the stride. If the kernel size is (1,1), the output size is the input size divided by the stride. Larger kernel sizes will reduce the output size by a greater amount as the kernel size increases. The exact relationship given in Equation 2.1.

2.6.2 Channels

Different kernel weights can be applied to the same image to give a different convolution output. This is very often done in neural networks since each kernel can help to extract information relating to a different aspect of the input. An example for lung tumour detection is that one kernel could help to identify the corner regions of the tumour, and another could extract information regarding the straighter sections of the tumour outline. The output given by each different kernel is called a *channel*, and these channels are stacked to form the last dimension of data which is propagated through a neural network, as was introduced in Section 2.2.5.

2.6.3 Transposed convolutional layers

Up-sampling can be performed using a variety of standard techniques in signal processing, but the most effective to learn a non-linear up-sampling function is learned up-sampling. Learned up-sampling uses learned kernel weights and biases to dictate a non-linear up-sampling function, in a transposed convolutional layer.

Transposed convolutional layers can be used to reverse the change in dimension that occurs as a result of convolutional layers. This layer type usually outputs an image which is of a larger dimension than the input image. However, these layers are not the inverse of the convolutional layer, and actually work via a different series of calculation steps. The only manner in which they are the inverse of a convolution is that they provide the inverse spacial dimensions. The steps involved in the transposed convolutional layer are as follows:

1. create a sparse matrix with zero rows and columns in between each of the rows and columns in the original matrix.
2. pad the image.
3. perform convolution on the generated image with a stride of 1.

If the aim of a transposed convolutional layer is to invert the spacial reduction performed by a convolutional layer, the number of zero rows and columns inserted in step 1 is equal to the stride of the convolutional layer minus one. The padding in step 2 should be equal to (k-p-1) where k is the convolution kernel size, and p is the convolution padding. The convolution performed in step 3 should have a kernel size equal to the original convolution. This process allows for the kernel weights to be trained, and for the up-sampling to be learned.

2.6.4 Pooling layers

The purpose of pooling layers is to reduce the dimension of the input data by grouping nearby data entries and performing a function on them. In image processing, pooling takes adjacent pixels. Common types of pooling include max pooling, min pooling and average pooling. These are self explanatory, taking the maximum, minimum and mean average of the pixels considered. Pooling can be used to decrease the number of weights used in a network, or just to reduce the number of layers required to perform a specific degree of down-sampling. In convolutional neural networks, convolutional layers are followed by activation layers, and then the activation output is pooled.

The most common pooling is max pooling in convolutional neural networks, since this maintains the contrast of the image. Areas which are background are often filled with zeros, and will remain this way with max pooling, whereas areas of interest keep the most

prominent (highest valued) pixel of interest. Despite the advantages of max pooling in maintaining image contrast, it is widely accepted that to reduce the quality of an image, average pooling also has advantages. This is especially true for low image resolutions, since when different important image features belong to neighbouring pixels, max pooling will completely disregard all pixels except for one. Average pooling, on the other hand, will maintain elements from all grouped pixels. A visual of max pooling is displayed in Figure 2.4, which reduces the input data resolution from 4x4 to 2x2.

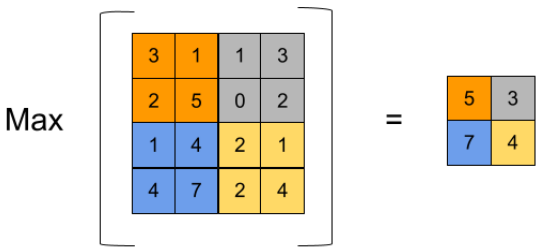


Figure 2.4: Visual depiction of max pooling

2.7 The Auto-encoder Architecture

As mentioned earlier, convolutional and pooling layers have the effect of down-sampling the input, since they reduce its spacial dimensions. As a result of this, the output of multiple convolutional and pooling layers must be up-sampled to reconstruct the original data dimensions. The most effective form of up-sampling has been stated in Section 2.6.3 to be learned up-sampling by using transpose convolutions. Learned up-sampling uses a neural network to perform the up-sampling process, allowing the learned weights and biases to dictate a non-linear up-sampling function.

The convolutional neural network can now be split into two general sections: the down-sampling section and the up-sampling section. In between these sections, the data representation will be at its lowest dimension. The section of the network which performs the down-sampling encodes the data to reach its low-dimension representation, and is called the encoder. Similarly, the up-sampling section of the network decodes this representation, and is called the decoder.

2.8 UNet

UNet was developed as an improvement to the basic convolutional network architecture explained in Section 2.7. One if its primary uses is applications in semantic segmentation, making it extremely relevant to this project. It is made up of a learned encoder, followed by a learned decoder. The main architectural improvement that UNet brings to encoder-decoder networks is that of skip, or shortcut, connections [6]. These connections allow information to pass directly from encoder to decoder layers that are of the same dimension, without needing to go through all of the encoder layers first. This prevents the loss of some more complex or higher frequency information, whilst also keeping the ability of the encoder-decoder architecture to detect high-level features very well. This has the overall effect of allowing the network to consider finer information while keeping the overall computation cost low. An image of UNet is displayed in Figure 2.5.

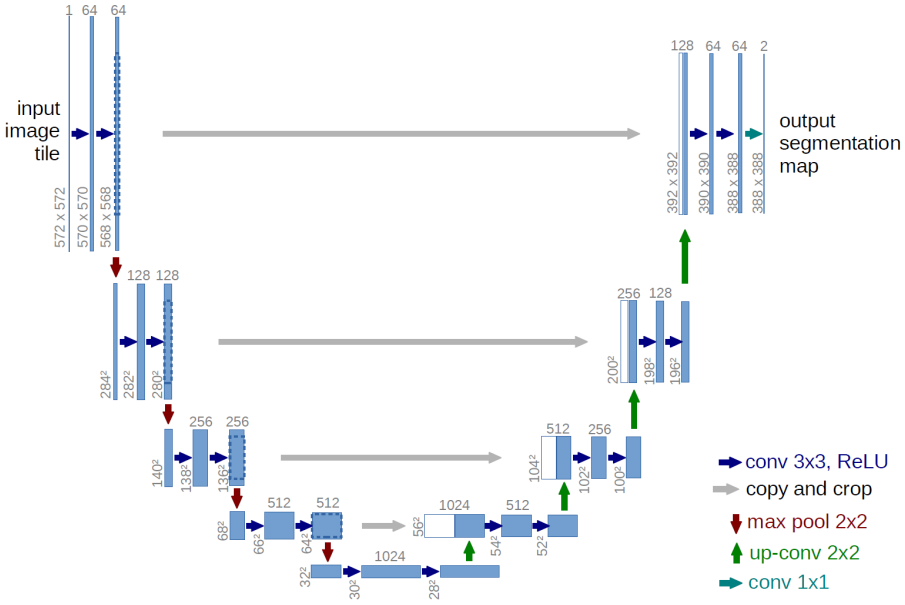


Figure 2.5: UNet [7]

UNet has been extensively developed upon for image segmentation purposes, and a list of research references, along with a summary of their main contributions, is given in Table 2.1.

Architecture name	Main contribution
TDC-UNet [14]	Triple UNet with dilated convolution
3D Md-UNet [15]	Multi-dataset collaboration
TMD-UNet [16]	Triple-UNet with multi-Scale input features and dense skip connection
MS-UNet [17]	A multi-scale UNet with feature recalibration

Table 2.1: Developments on UNet for image segmentation

The references listed in Table 2.1 have all been tested for medical image segmentation, each providing better results than UNet alone by introducing a new technique into the network. These four papers were all released in 2021 and 2022, and each demonstrate an improved performance on UNet. Techniques which are used in these networks include dilated convolution or multi-dataset collaboration. These techniques are examples of how UNet can be optimised for image segmentation. The direction that this project will take is to focus on another development on top of UNet: The Multi-resolution Convolutional Neural Network. This type of network is the main focus of this report, and will now be explained in detail.

2.9 Multi-resolution Convolutional Neural Networks (MCNNs)

We have seen that UNet gives a good basis for an encoder-decoder convolutional architecture to apply to image segmentation problems. Because of this, it will sometimes be referred to as a "backbone" upon which modifications can be built. UNet encodes the original resolution image into an intermediate representation which is of a lower resolution, then decodes this to an output with a resolution which matches the input resolution. This, by itself, already utilises the concept of combining information extracted from different resolutions of images. The main two reasons for this are:

1. UNet allows low frequency (low resolution) components in the data and high frequency components to propagate through the network, by means of lower resolution representations in the encoder and decoder layers, and skip connections.

2. Decreasing the dimensions of each layer causes the network to require many less weights than if all layers were kept at the original resolution. This allows the network to train faster.

3. Propagating low frequency components through the network reduces over-fitting.

MCNN networks have been shown to give a better convergence characteristic when compared to UNet for certain image processing tasks, such as image de-noising. Due to the promising nature of MCNNs, this type of network will tested to see if it can achieve faster convergence than its UNet backbone for the purpose of Lung Tumour semantic segmentation.

2.9.1 Promising aspects of MCNN Networks

The paper titled "Multi-resolution CNNs for inverse problems" [8] presents data to show that MCNNs have yielded high accuracy when applied to image-to-image translation tasks such as: 1. super-resolution, 2. image generation and 3. image denoising. This suggests that the network type is capable of learning complex image processing operations and could be applicable to image segmentation. In UNet, the low frequency components are recovered first through the encoder layers, and the high frequency components are then recovered slowly through the decoder layers and skip connections. The suggested improvement to UNet is to connect each layer of the decoder to some convolutional layers, reshaping the layers a bit to give an output for each layer of the decoder. The earlier layers in the decoder will give more coarse outputs which contain lower frequency components, and vice versa. This has the effect of providing an output at all possible "frequency bands" that the network encodes, covering low frequencies as well as high frequencies at the output stage.

The results of this are as follows:

1. The multi-resolution reconstruction allows the convergence to avoid local minima. Due to this, converge to a lower, global loss function minimum may be possible.

2. The convergence is faster with the multi-resolution network for the tasks that have already been demonstrated, with the loss being less than or equal to the UNet loss.

These results are shown in Figure 2.6.

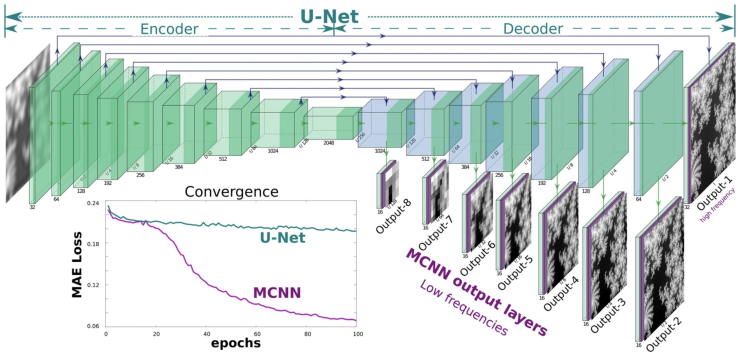


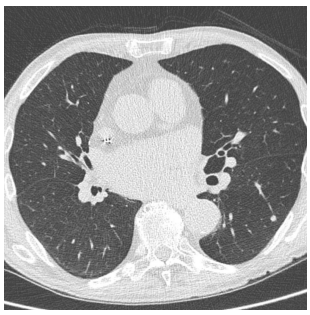
Figure 2.6: The multi-resolutional convolutional neural network architecture and loss characteristic

The main principle that the MCNN works on is that the model has multiple outputs, allowing for the calculation of multiple losses, all of which can contribute to the convergence to a loss function minimum. This should have some benefits in all image-to-image applications, so should be applicable to lung tumour segmentation.

2.9.2 Possible drawbacks of MCNN Networks

The convergence characteristic for MCNNs in Figure 2.6 is for predicting phases from defocused images. This application, like the other applications mentioned, is likely to benefit from low resolution output components, since a defocused image is in some ways similar to a low resolution image. Segmentation tasks, on the other hand, may not benefit from the use of lower network resolutions. This prompts the need for discussion of the potential downfalls of MCNNs for semantic segmentation.

When examined more closely, it becomes apparent that all three applications in the MCNN paper are for denoising images which have a significant degree of noise added to them. Because CT scans are typically not very noisy, the use of a multi-resolution CNN may lead to little or no improvement on this type of data. The noise level of CT scans compared to the images used in the multi-resolutional CNN paper is visually shown in Figure 2.7.



(a) Lung CT scan

(b) Diffuse reflected images

Figure 2.7: Demonstration of the very high degree of noise in diffuse reflected images (boxed in red) compared to the sharpness of a lung CT scan. This shows a likely disadvantage to using multi-resolution CNNs for lung CT scans

2.9.3 Evaluation of advantages and disadvantages of MCNNs

It has been stated that MCNNs can be applied to all image-to-image conversion problems, and help to propagate information about lower resolution components through the network. However, the loss function convergence characteristics that have been experimentally produced by using this type of network have all been applications involving images which have a degree of noise added to them. This is not the case for lung tumour segmentation. Therefore, results which give an improvement as large as shown in Figure 2.6 are not expected in this project. What can be expected is that convergence could be superior for some MCNN model implementations, due to the fact that the multiple outputs of the model correspond to the calculation of multiple loss values, which will aid the network in converging to a loss function minimum.

2.10 2D vs. 3D Networks

In Section 2.2.5 the different data dimensions for 2D and 3D inputs into a neural network were introduced. This is important, since the effectiveness of MCNN networks can be compared to the effectiveness of the equivalent depth UNet structure by using both 2D and 3D network inputs. Normally, lung tumour segmentation would only be performed on 3D input data with 3D neural networks. Because of this, the first part of this project implementation will involve constructing a 3D MCNN from a 3D UNet backbone.

2.11 3D Convolutions

3-dimensional convolutions contain kernel weights in the same way that the 2D convolutions in Section 2.6.1 do. However, these kernels are three dimensional, with an example possible size of (3,3,3), leading to $3^3 = 27$ total filter weights. The biases should also be added to this weights total, bringing the total weights to $27 + 3^2 = 36$.

A visual display of a 3D convolution is shown in Figure 2.8 [28]. From this Figure, it is evident that 3D convolutions allows information along the z-axis of the input data to be extracted by different kernels, theoretically increasing the performance capabilities of the network.

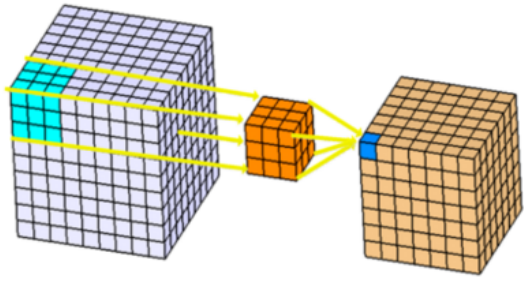


Figure 2.8: A visual of the dimension changes involved in 3D convolution

When applying 3D convolutions, just like 2D convolutions, the network must take in a fixed input size. This is necessary since the network weight matrices are defined by the shape of each of the network layers, since they must weight each connection from a given layer to each connection in the next layer. Therefore, when the raw data which will be used as input to a 3D network has a variable size, a fixed-size volume which is part of the input is usually chosen as the input size to the network. This input is called a *patch*, and the patch size must be carefully selected depending on the intended network application.

Chapter 3

Requirements capture

Now that the reader has a good technical understanding of the key building blocks which form both CNNs and MCNNs, a more detailed list of the exact aims of this project are formulated to aid the clarity of the report. In the sections which follow, the goal is to work towards achieving all of the requirements specified. The requirements are as follows:

1. Demonstrate that MCNN model losses can converge faster than their UNet equivalent network for binary image segmentation tasks.
2. Explain the effects of changing the following MCNN parameters when performing lung tumour segmentation, displaying tests when necessary:
   (a) The pooling type used to generate low resolution labels.
   (b) The 'patch' size (network input size), and how to select a patch.
   (c) The ratio of positive patches (which contain a tumour) to negative patches (which do not contain a tumour) used during training.
   (d) The number of resolutions used for the final network loss.
   (e) Weighting different resolutions differently for the final loss.
   (f) Using these different losses at different points during training.
3. Obtain final results which beat the equivalent UNet network, using the experiments in 2.(a) to 2.(f).
4. Explain the insights gained from the testing process for potential future work.

The initial demonstration of the convergence of MCNNs given by point 1 should be performed using the fastest possible implementation, since if convergence cannot be achieved using MCNNs, there is little point in testing MCNNs on lung tumour detection tasks with more complex implementations.

Following this initial implementation, the implementation should be expanded, with the option to control and alter all parameters which will be investigated. Design choices and testing will then be performed on MCNNs with varying parameters, listed in 2.(a) to 2.(f) above. The results from these tests will be the main contributions that this report provides to this research field.

Finally, results will be presented which combine the knowledge from all of the tests which were previously conducted, and the main insights will be summarised. Suggestions for potential future testing will be included at the end of this report.

Chapter 4

Analysis and design

4.1 High-level Design Overview

When designing any neural network using Python neural network frameworks, there is a standard series of main steps that must be implemented and integrated with each other to make up the final design. These design steps are given in Figure 4.1.

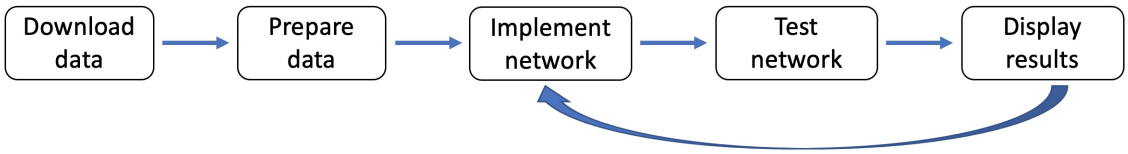


Figure 4.1: A diagram of the iterative design process for the project

For each model that was implemented, after results were obtained and analysed, the model was adjusted to test more promising potential improvements. This resulted in an iterative design process, as pictured in Figure 4.1. The high-level details behind how each step of the process was implemented are covered in this section. Any initial design choices which needed to be changed are also included, to show the thought that went into the final design.

4.2 Initial Design Pipeline

The following steps explain how the design pipeline was initially implemented:

1. Code Environment Choice - A Google Colab notebook was used, with a Colab Pro + subscription.
2. Data Downloading - The Medical Segmentation Decathlon Lung Tumour Dataset was downloaded to a personal Google Drive, and the Google remote machine was provided with access to the drive.
3. Data Processing - spacial dimensions halved from (512, 512) to (256, 256) to enable the images to be stored in the remote machine's RAM. Data was scaled after. Data was collected into patches with dimension (64,256,256) by iterating through the first data dimension.
4. Network Implementation - Keras implementation of a deep UNet model with 9 different auto-encoder layers. This network is given in Figure 4.2.
5. Network Testing - The MCNN network in Figure 4.2 was tested by using all 9 resolution outputs, and compared to the UNet backbone network. Pixel-wise binary cross entropy loss was used.
6. Network Results - The convergence characteristic of the MCNN and UNet architectures is given in Figure 4.3.

As listed in point 5, a diagram of the spacial dimensions throughout the MCNN network is given in Figure 4.2:

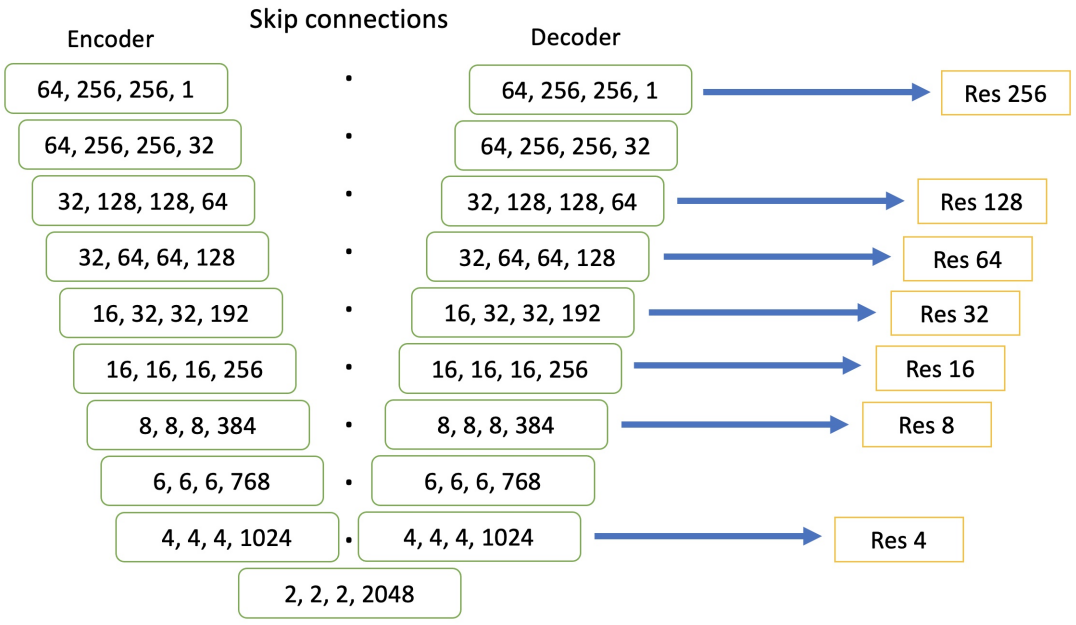
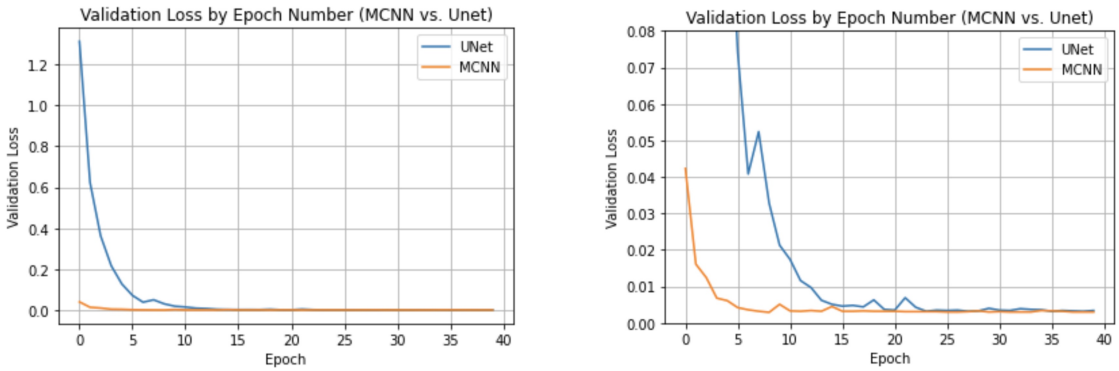


Figure 4.2: A diagram of all intermediate dimensions of a 9 layer MCNN

To reduce the spacial dimensions of the input by a factor of two, a stride of 2 is used in the horizontal and vertical dimensions of each image. A kernel size of (3,3,3) is used for each convolution. The outputs are named "Res X" for different x and y resolutions X. A useful outcome from this initial test was that the MCNN network converged to a low loss significantly faster than its UNet equivalent network. This is seen in the results in Figure 4.3.



(a) (b)

Figure 4.3: 9 layer MCNN and UNet network binary cross-entropy losses compared

The advantages of the Google Colab Pro + subscription are that background execution is allowed, so the browser tab does not require constant supervision when networks are running. Additionally, up to three high RAM (51GB total RAM) sessions can be executed simultaneously, which greatly speeds up the testing.

The MCNN structure that was implemented altered the original image resolution from (512, 512) to (256, 256) using mean average pooling. Using the same method, data labels for the network were constructed which ranged in resolution from (256, 256) down to (2, 2). This was then compared to a max pooling implementation, and it was found that max pooling yielded better results.

The reasoning behind not stopping at a relatively low resolution, for example (4, 4), is that the UNet structure used involves residual, or skip connections. These allow useful information to propagate across layers with the same spacial dimensions. This raises the question of whether additional encoder and decoder layers may increase the potential of the network to over-fit. However, the lower the spacial dimensions are of the image, the lower the frequency of the information they encapsulate. Since over-fitting is a property which almost exclusively applies to the network's ability to fit to a more specific and volatile function than the target function of the task required, it is an outcome which cannot realistically be obtained through the addition of lower resolution encodings. This is demonstrated by this network not over-fitting, since the validation loss never noticeably increases throughout training.

To summarise, the contributions that this implementation has given to this research project are that:

1. MCNNs can speed up training convergence when applied to binary image segmentation problems.
2. Max pooling should be used to scale down the resolution of the label data, since it comprises of zeros and ones, not intermediate decimal values.
3. All possible output resolutions can be used to give improved results for MCNN image segmentation tasks, and can result in faster loss convergence under the correct circumstances.

4.3 Difficulties faced with the Initial Design

The initial design described above demonstrates that MCNNs can be an effective extension to UNet models for the purpose of binary semantic segmentation, since the loss convergence characteristic of the MCNN model is faster than the UNet model. However, the following criteria from Section 3 were either implemented inefficiently or were not able to be implemented with this design:

1. The input image size is reduced to (256,256), meaning that state-of-the-art results are unobtainable.
2. Data is processed as NumPy arrays, which is slow and inefficient.
3. The method for selection of network input patches will result in negative patches approximately 95% of the time, causing the network to mainly fit to the non-tumour regions.
4. Dice loss implementations are inefficient, since one-hot conversion is implemented in NumPy.
5. The multiple network outputs cannot easily be weighted in the training process by using Keras.
6. The different resolution network outputs cannot be easily saved after running a training process. Only the total loss can be stored in a CSV file.

While point 3 is implementable, and points 5 and 6 may be possible to implement after further examination of the TensorFlow back-end source code, the other issues cannot be addressed without changing the project implementation. Because of this, an implementation was constructed from scratch which made use of the MONAI library, built on the PyTorch framework [29]. This implementation makes up the final design pipeline, and will now be explained at a high level. The finer details of the implementation will be

explained in Section 5.

4.4 Final Design Pipeline

The final design made use of the "Medical Open Network for Artificial Intelligence" library, abbreviated as MONAI. The in-built functions that are provided by MONAI allow design choices to be made without the need to code efficient machine learning functions from scratch.

The following steps explain the details of how the final design pipeline was implemented:

1. Code Environment Choice - A Google Colab notebook was used, with a Colab Pro + subscription.
2. Data Downloading - The Medical Segmentation Decathlon Lung Tumour Dataset was downloaded to a personal Google Drive, and the Google remote machine was provided with access to the drive.
3. Data Processing - MONAI integrated transforms were used to process the data, at its original resolution. Data was collected into patches with dimension (96, 96, 96).
4. Network Implementation - MONAI network layers were customised to build a Medical UNet model with 5 auto-encoder layers in total.
5. Network Testing - The MCNN network was tested by using all 5 resolution outputs, and compared to the equivalent UNet backbone network. Dice loss was used, and the Dice metric was evaluated.
6. Network Results - The convergence characteristic of the MCNN and UNet architectures was straightforward to record from the training process code.

A key design decision which was made for the final implementation was the decision to re-code the entire data-loading process and model from scratch in PyTorch. This allowed for a much more customisable training process, more efficient data loading, all built upon MONAI [29], which is a dedicated medical imaging library. The full implementation will now be explained in the next section of the report.

Chapter 5

Implementation

5.1 Data Downloading

The Medical Segmentation Decathlon Lung Tumour Dataset is located on the Medical Segmentation Decathlon website [33], and is open source. It utilises CT scans from the Cancer Imaging Archive, specifically from the LIDC-IDRI dataset. The dataset is formatted into a .tar file, which was downloaded to my local Google Drive, and extracted to give the raw NIFTI files. The total size of the dataset was 8.53GB. In order to use the dataset, the Google remote machine needed to be given access to the contents of the Google Drive. This was done by mounting the drive in the Colab notebook, and manually giving permission to the remote machine by logging in to the correct Google account via a pop-up window. After this, the directory containing the data files can be navigated to, and the files can be operated on.

5.2 Data Processing

The MONAI library includes a set of transforms which can be used during data preprocessing. These needed to be carefully selected by considering their appropriateness for lung tumour segmentation clarity. The final transforms chosen are:

1. `Orientationd` - used to align all data along the same axis based on the affine matrix provided in the NIFTI file, explained in Section 2.2.4.
2. `CropForegroundd` - used to eliminate the border pixels which are zero-valued, reducing images and labels to only include the area with CT results.
3. `RandCropByPosNegLabeld` - used to crop volumes from the 3D CT scan images. This was used so that a positive label would be at the centre of a cropped sample 50% of the time, and a negative label would be at the centre the other 50% of the

time. The dimension of the cropped data was set based on the height and width of the original images. The closest that a positive label pixel is to the edge of an image is 58 pixels away. A final cropped shape of (96,96,96) was decided to be suitable since any positive label at the centre of this would be maximally 48 pixels away from the edge of the image, and the crop would not extend past the border of the image, making it valid. Spacial dimensions with a value of 96 each means that the resolution can be halved 5 times in total before reaching a size of (3,3,3). This leaves many lower resolutions available to experiment with when applying an MCNN to the data.

Aligning the data, cropping it to exclude null data, and selecting data patches to include equal numbers of positive and negative samples, make up the data preprocessing for this implementation. Additional processing such as scaling the values of the images was deemed unnecessary, since it yielded no improvement in results.

5.3 Data Loading

The data is then cached as a `monai.data.CacheDataset` [25] to enable approximately 10 times faster data loading during training. The training data transforms specified above are applied to the data when constructing the `CacheDataset`. This dataset is then loaded into a `DataLoader` object under a MONAI framework which is based upon a standard PyTorch data loader. After the data is in a `DataLoader` object, it can be used within a standardised PyTorch training process by using the `batch_data` function to load a batch of image and label data.

5.4 Model Construction

5.4.1 UNet Backbone

The UNet backbone was built using the MONAI `Convolution` and `TransposeConvolution` layers, along with PyTorch tensor concatenation, i.e. `torch.cat([tensor_a, tensor_b])` for residual connections. Convolutional layer kernel sizes were varied from (7,7,7) down to (3,3,3), and a stride of (2,2,2) was used in all layers with `padding = 'same'` to halve the data dimension. The UNet residual layers make use of an improvement which was first demonstrated on segmentation of the left ventricle of the heart [27], which uses a convolutional layer after each residual layer concatenation. `tensor_c = torch.cat([tensor_a, tensor_b])` results in `tensor_c` having double the number of channels as `tensor_a` and `tensor_b`. To halve the channels again, a MONAI convolutional layer is defined as follows:

```
self.conv_params = Convolution(data_dimension, in_channels, out_channels, strides,
kernel_size, activation, norm_type, dropout, bias, conv_only, is_transposed)
```

Each of the parameters within the convolutional layer are set to the following values for a convolution which follows a residual connection:

```
self.res_conv = Convolution(3, 2*channels, channels, strides=1, kernel_size=3,
act=Act.PRELU, norm=Norm.BATCH, dropout=0.0, bias=True, conv_only=False,
is_transposed=False)
```

`TransposeConvolution` layers are defined by setting `is_transposed=True`. A dropout rate of 0.0, and the PRELU activation function, are used for all layers. Setting `conv_only=False` makes the layer a convolution, batch normalisation, and activation layer.

After additional channel dimension testing, the final version of UNet used was determined. A high-level overview of this architecture is given in Figure 5.1. In this diagram, the data z-dimension is omitted, since data with dimension (x dim, y dim, z dim, channels) is not visualisable. However, the z-dimension is equal to the x and y dimensions at all points in the network, so can be inferred.

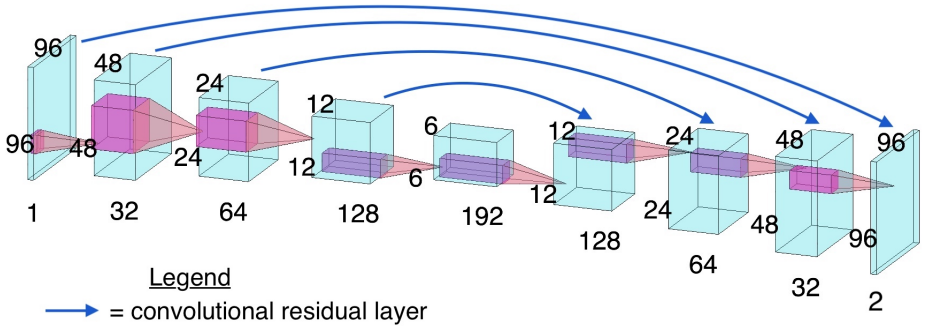


Figure 5.1: Final UNet design, with the z-dimension omitted for visualisation

As can be seen from Figure 5.1, the UNet contains 5 different resolutions at which outputs can be extracted. The MCNN structure which utilises all possible output resolutions is given in Figure 5.2.

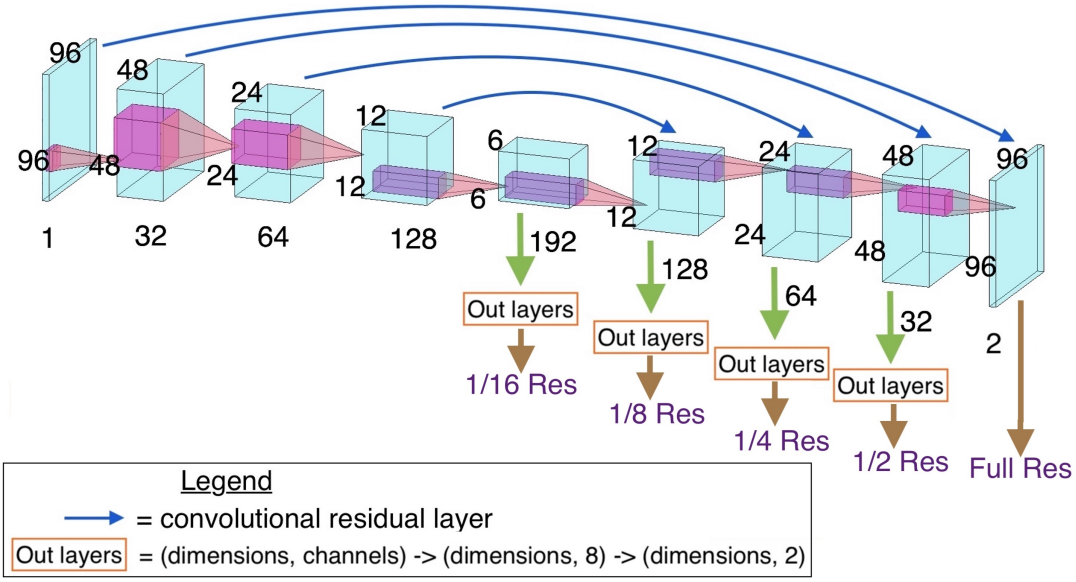


Figure 5.2: Final MCNN design, with the z-dimension omitted for visualisation

It should be observed that the network has 2 output channels, instead of 1 channel as is usually the case for greyscale image outputs. This is because the Dice metric uses one-hot encoded vectors. A binary class label vector of size (3) is converted to a vector of size (3,2) when one-hot encoded, as shown in Figure 5.3. Likewise, labels of size (96, 96, 96) are converted to size (96, 96, 96, 2).

Original One-hot encoding

[1,0,0] -> [[0,1],[1,0],[1,0]]

Figure 5.3: Binary one-hot encoding. Tensor dimensions go from (3) to (3,2).

The activation function used for the final layer of the network is the Softmax activation function, which forces the sum of all channel outputs to equal one. This is appropriate since the sum of all elements of a one-hot encoded vector is equal to one. The Softmax output represents the predicted probability of the label belonging to a certain class (either background or tumour in this case). The ADAM optimiser was chosen for this network, with a learning rate of 10^{-4}.

The names of the different outputs of the MCNN network in Figure 5.2 are given in Table 5.1, along with their dimensions.

Output name	Output dimensions (x dim, y dim, z dim, channels)
Full Res Output	(96, 96, 96, 2)
1/2 Res Output	(48, 48, 48, 2)
1/4 Res Output	(24, 24, 24, 2)
1/8 Res Output	(12, 12, 12, 2)
1/16 Res Output	(6, 6, 6, 2)

Table 5.1: Output names used in this report, along with their dimensions

Each output in Table 5.1 is used along with the label of the same dimension, to calculate a loss function value. The Dice loss is used here, and is explained in Section 2.5.2. The total network loss must be some combination of the five available losses which have been calculated in Table 5.1. The network architecture is now at a stage where the MCNN can be tested with different combinations of the available output losses. To complete the implementation, the model training code must include the calculation of these losses by using the different network outputs and labels.

5.5 Model Training

Since the MONAI `dataloader` object does not provide functionality to load in multiple output labels for a singular input image, the label resolutions needed to be lowered during each training step. This implementation is necessary since the area that makes up each image patch is randomly selected, and the number of different possible patches is too high to store different resolution labels for every possible patch. For each image, there are approximately 1000 possible positive label patches (assuming a tumour size of 10x10x10 pixels), and 64,000,000 possible negative patches (assuming 400x400x400 pixels of valid negative labels in a CT scan with dimensions (512,512,400)). The storage of tens of millions of labels for each input is simply impossible. The code below shows how the different labels were calculated during each training step:

```
for batch_data in train_loader:
        step += 1
        inputs, labels = (
            batch_data["image"].to(device),
            batch_data["label"].to(device),
        )

        labels1 = make_block_reduce(labels.cpu().detach().numpy(), dim =
        ↪  (2,2,2,2), mode=np.max)
        labels2 = make_block_reduce(labels1, dim = (2,2,2,2), mode=np.max)
        labels3 = make_block_reduce(labels2, dim = (2,2,2,2), mode=np.max)
```

```
labels4 = make_block_reduce(labels3, dim = (2,2,2,2), mode=np.max)

optimizer.zero_grad()
outputs, mcnn4, mcnn3, mcnn2, mcnn1 = model(inputs)

loss_orig = loss_function(outputs, labels)
loss1 = loss_function(mcnn1, torch.from_numpy(labels1).to(device))
loss2 = loss_function(mcnn2, torch.from_numpy(labels2).to(device))
loss3 = loss_function(mcnn3, torch.from_numpy(labels3).to(device))
loss4 = loss_function(mcnn4, torch.from_numpy(labels4).to(device))
loss = loss_orig + loss1 + loss2 + loss3 + loss4

loss.backward()
```

Here, the `make_block_reduce` function uses the SciKitImage function `block_reduce`, and first appears in the MCNN paper by F. Wang et al. [8]. It is adapted to use max pooling is this implementation.

Aside from this addition to the PyTorch training process, the rest of the training is slightly adapted from a standard PyTorch training process, of which many examples are given in PyTorch documentation [30]. Now that the training process is correctly coded, the desired outputs need to be recorded to evaluate model results.

5.6 Evaluating Model Results

In order to be able to evaluate the results of a model at any time, the model must be saved after training. This is performed using `torch.save_model(model_name, directory)`. Additionally, the losses and Dice metrics of the models needed to be stored, and saved to an output file for later analysis. The method of data storage was in a Pandas `DataFrame`, which is saved to a CSV file using the command `dataframe.to_csv`. The training data was split so that the last 10 CT scans were used as a validation set, which weren't loaded into the training `dataloader`, but used to calculate the Dice metric. This train/validation split is the standard procedure followed to evaluate a model before submission during competitions in the Medical Segmentation Decathlon.

The inference method chosen was sliding window inference, using `monai.inferers.sliding_window_inference` [31]. The spatial window size for inference was chosen to be (160,160,160), and the inference batch size was chosen to be 4.

The model size of a trained MCNN is approximately 1.1GB, so a Google Drive storage upgrade needed to be purchased to give 100GB total storage, in order to store all of the models trained throughout this project. This way, any model can be loaded and evaluated

after training if necessary for results analysis. Additionally, this makes all trained models for this project available upon request.

Chapter 6

Testing

6.1 Loss function testing

6.1.1 Different output resolution numbers

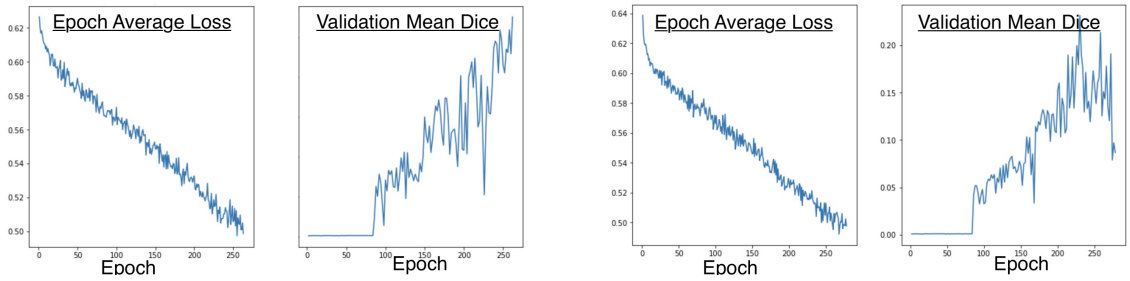
The final MCNN model implementation was given in Figure 5.2, and has 5 different loss outputs. Not all of these losses need to be used during testing, so it is logical to start by testing which losses are beneficial to use. The first test that needed to be performed was to evaluate the difference which is observed when the lowest resolution losses are used, compared to when the lowest resolution losses are not used.

The two loss function formulations that were decided on were trained for 264 epochs each, and are given by:

$$\begin{aligned}\text{Five Res Loss} = \text{Full Res Loss} + 1/2\ \text{Res Loss} + 1/4\ \text{Res Loss} \\ + 1/8\ \text{Res Loss} + /16\ \text{Res Loss}\end{aligned}$$

$$\text{Three Res Loss} = \text{Full Res Loss} + 1/2\ \text{Res Loss} + 1/4\ \text{Res Loss}$$

The best Dice metric reached by the *Three Res Loss* model during training was 0.2752, and the best Dice metric reached by the *Five Res Loss* model during training was 0.2398. However, since neither model was trained to completion, and Dice metric curves are prone to spiking at slightly different epoch numbers, a comparison of Dice metrics is not an appropriate evaluation method at this stage. To demonstrate the volatility of the dice metric by epoch number, the loss and metric evaluation results are given in Figure 6.1.



(a) Three Res Loss model (b) Five Res Loss model

Figure 6.1: Full resolution loss curve, and Dice metric by epoch

In order to evaluate the performance of both models more holistically, a visual examination of the tumour segmentation predictions by each model is performed. Results of this were very interesting, and not as one might expect. These results are shown in Figure 6.2.

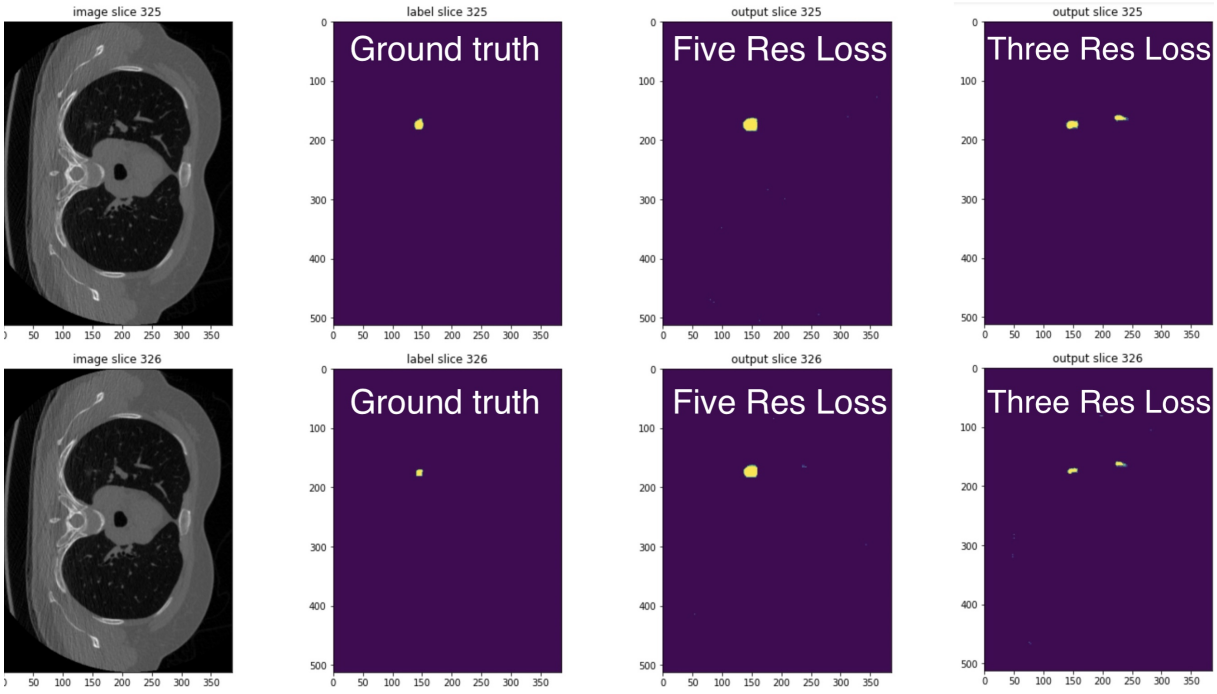


Figure 6.2: Visual segmentation comparison of different loss formulations

From Figure 6.2, it can be observed that:

1. Although the Five Res Loss model has a lower Dice metric, it is able to identify the tumour position more accurately than Three Res Loss.
2. The Three Res Loss model seems to more accurately identify the tumour size, so may have greater precision than the Five Res Loss model.

While these results are interesting, they may just be an anomaly for this specific training cycle, and need to be confirmed with further testing.

6.1.2 Weighting lower resolutions higher

From the model comparison in Section 6.1.1, it is possible that the use of lower output resolutions could allow the location of the lung tumour to be detected more accurately. Another way to see this is that if this problem became a classification problem, the Five Res Loss model would outperform the Three Res Loss model.

Up until now, all output losses have been weighted equally, and summed together. However, this does not need to be the case, and it would be logical that weighting lower resolution losses even higher may make the results from Section 6.1.1 even more pronounced.

6.1.3 Three proposed losses

The ability to use different weightings for the different output resolutions results in an infinite number of possible combinations of the different network output losses. When evaluating the efficacy of weighting the lowest resolution loss the highest, it makes sense to assign incrementally higher weightings to each lower resolution loss. Since there are five losses in total, this results in the weightings (0.2, 0.4, 0.6, 0.8, 1.0) for the 5 losses. This decision was also made because experimentation in the original MCNN paper [8] demonstrated that leaving a component of full resolution loss enables the convergence to maintain proximity to a full resolution optimum. The goal of using lower resolution losses is to aid convergence to a full resolution optimum, hence this prior work is useful guidance here.

If lower resolution loss performance is to be compared with higher resolution loss performance, it is necessary to construct multiple loss functions, and compare their results. It is appropriate for the higher resolution loss function to use the inverse weightings that the lower resolution loss function makes use of, and a medium resolution loss to act as a control, with the sum of all weights being identical for all losses so that the learning speed is not artificially increased.

The equations for the three losses which are obtained as a result of the earlier logic are:

$$\begin{aligned}\text{High Res Loss} = \text{Full Res Loss} + 0.8{*}1/2\ \text{Res Loss} + 0.6{*}1/4\ \text{Res Loss} \\ + 0.4{*}1/8\ \text{Res Loss} + 0.2{*}1/16\ \text{Res Loss}\end{aligned}$$

$$\begin{aligned}\text{Medium Res Loss} = 0.6{*}\text{Full Res Loss} + 0.6{*}1/2\ \text{Res Loss} + 0.6{*}1/4\ \text{Res Loss} \\ + 0.6{*}1/8\ \text{Res Loss} + 0.6{*}1/16\ \text{Res Loss}\end{aligned}$$

$$\begin{aligned}\text{Low Res Loss} = 0.2{*}\text{Full Res Loss} + 0.4{*}1/2\ \text{Res Loss} + 0.6{*}1/4\ \text{Res Loss} \\ + 0.8{*}1/8\ \text{Res Loss} + 1/16\ \text{Res Loss}\end{aligned}$$

6.2 False Positives and "Focus"

The three loss functions defined in Section 6.1.3 were each used in the MCNN model defined in Section 5.4, and trained for 150 epochs each. This epoch number was chosen to be displayed in this section because this resulted in the best visual clarity of results. The network which uses High Res Loss is called *HighMCNN*, the network which uses Medium Res Loss is called *MedMCNN*, and the network which uses Low Res Loss is called *LowMCNN*.

Visual outputs of the tumour segmentation are now examined for these three networks. This will be done by first examining how the models perform on the "first" slices of a tumour, which are defined here as the two lowest index number slices within a CT scan which contain a visible positive label. The output for each of the three networks is given in Figure 6.3.

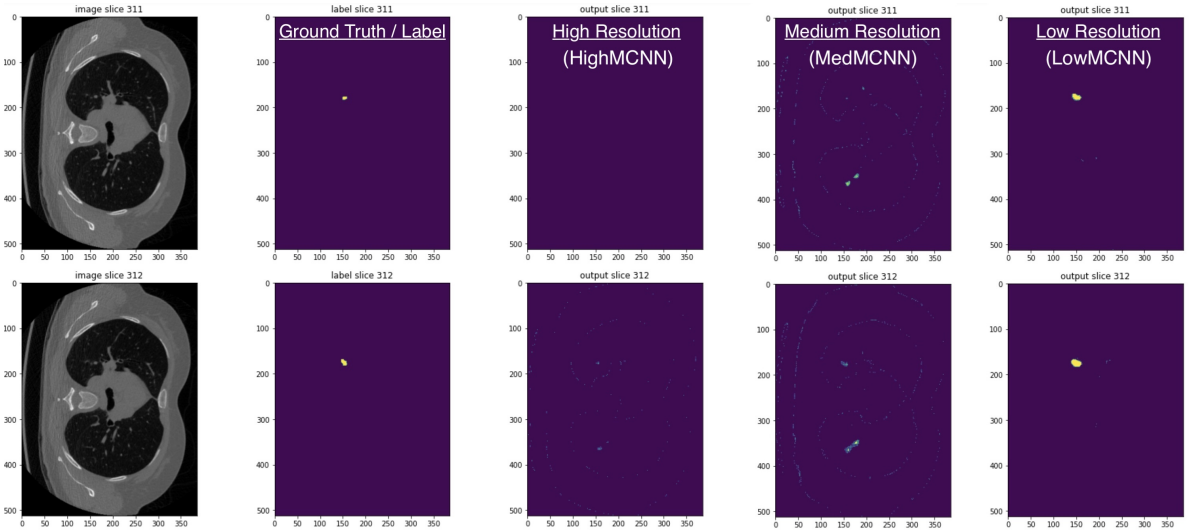


Figure 6.3: Performance of different resolution loss MCNNs for detecting the start of a tumour, after 150 epochs

Once Figure 6.3 is closely examined, a trend becomes evident. The MCNN with the loss which is more heavily weighted towards lower output resolutions predicts the tumour segments in the correct location, whereas MedMCNN can only guess the second slice with a very small predicted tumour size. HighMCNN performs the worst, with almost no detection of either of the first two tumour slices. Additionally, when the HighMCNN and MedMCNN resolts are examined very closely, the outline of the lungs can be seen in their predictions. To make this more clear, a zoomed version of the bottom left corned of the three network outputs and the ground truth is given in Figure 6.4.

The persence of the lung outline in HighMCNN and LowMCNN's predictions is, of course, an incorrect output, and shows that visually LowMCNN has much superior detection of the starting slices of tumours when trained for 150 epochs. A disadvantage of LowM-

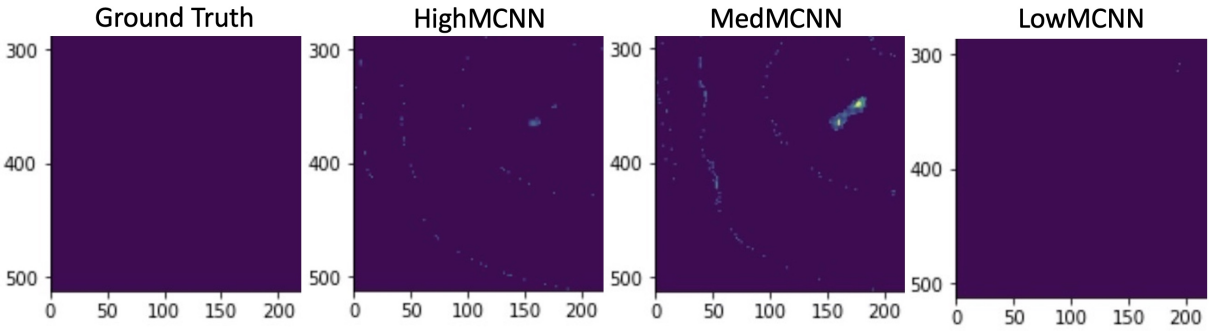


Figure 6.4: Lower left corner of labels and predictions in Figure 6.3, zoomed in

CNN's predictions are that each tumour slice is predicted to be larger than it actually is. This overestimation results in it having a Dice metric which is only slightly higher than the other networks, at 0.14 compared to 0.11 for HighMCNN and 0.08 for MedMCNN. However, it must be remembered that Dice metrics at early training stages are not indicative of later training stage convergence.

These results show that LowMCNN has a low probability of predicting a tumour to be in the wrong location compared to HighMCNN and MedMCNN, since both HighMCNN and MedMCNN contain tumour location predictions which lie in the lower half of the image. Therefore, if this was a classification problem, LowMCNN would most quickly converge to a state where there are no / very few false positives. Intuitively, this makes sense, since by max pooling the output labels, a lower resolution image is generated where a greater proportion of the total pixel number contain a tumour. This has the effect of partially correcting for the large class imbalance between tumour and non-tumour regions, causing the network to prioritise tumour segmentation rather than gradually segmenting out the background, as is done in HighMCNN and MedMCNN. This effect will be called "Focus" in this report, since the network focuses on segmenting the foreground, rather than segmenting out the background.

6.3 The addition of fine-tuning

It has been seen in Figure 6.3 that the predicted tumour size using the LowMCNN network is significantly larger than the actual tumour size. This is an effect of using lower resolution contributions to the network loss function, since HighMCNN and MedMCNN both predicted a smaller tumour area. Because of this, it becomes evident that a network may benefit from some elements of low resolution loss function training, and some elements of high resolution loss function training.

Network training which mainly uses losses calculated from high resolution outputs improve the precision of the prediction. This intuitively makes sense since higher resolution data representations contain much more detailed information, theoretically allowing a pixel-precise border of the tumour region to be learned. Every time a label is pooled, information from that label is lost, meaning that a pixel-precise border of the tumour is impossible to learn.

Due to the fact that LowMCNN outputs an imprecise but accurate segmentation, this can be recognised as a scenario that is perfect for the application of fine-tuning. Fine-tuning is a technique which, when applied to neural networks, involves the further training of a model in order to make final small improvements to the network convergence. To preform fine-tuning, an element of the network training must be optimised for convergence to a loss function minimum. Here, the weightings of the losses which contribute to the loss function itself will be changed, in order to perform fine-tuning.

6.4 FocusNets

In order to further investigate the results obtained from HighMCNN, MedMCNN and LowMCNN, the same loss functions will be used as are used in these three networks to continue testing. However, now the networks will be trained to completion, or at least near enough to completion to evaluate their Dice metrics.

An approach which includes initial network "Focus", followed by network fine-tuning will be used. To achieve this, networks will change loss function from Low Res Loss to Medium Res Loss and High Res Loss after training for a given number of epochs. For ease of reference, these trained networks will be referred to as a class of networks called FocusNets. The precise definition of each network, and results of these networks, will be included in the Results section of the report.

Chapter 7

Final Results

7.1 FocusNets Definition

It has been determined that a combination of a loss function weighted more towards lower resolutions, and a loss function weighted more towards higher resolutions, could be used to achieve results which could resemble the initial training, and fine-tuning stages of loss function convergence. This will be implemented by changing the loss function which a network uses after a given number of epochs, and continuing training. This class of trained networks is named FocusNets. The number of epochs of each loss function that each FocusNet will train for is given in Figure 7.1.

	Number of training epochs			
Network Name	Full Res Loss	Low Res Loss	Medium Res Loss	High Res Loss
Medical UNet	800	0	0	0
FocusNet100	0	100	100	600
FocusNet200	0	200	200	400
FocusNet300	0	300	300	200
FocusNet400	0	400	400	0

Figure 7.1: The number of epochs of each loss function that each network will train for

7.2 Dice Metric Analysis

7.2.1 FocusNets Dice Graphs

The dice metric was evaluated during training for each epoch, for each FocusNet model. The Dice metric graphs for FocusNet100, FocusNet200, FocusNet300 and FocusNet400 are given in Figure 7.2.

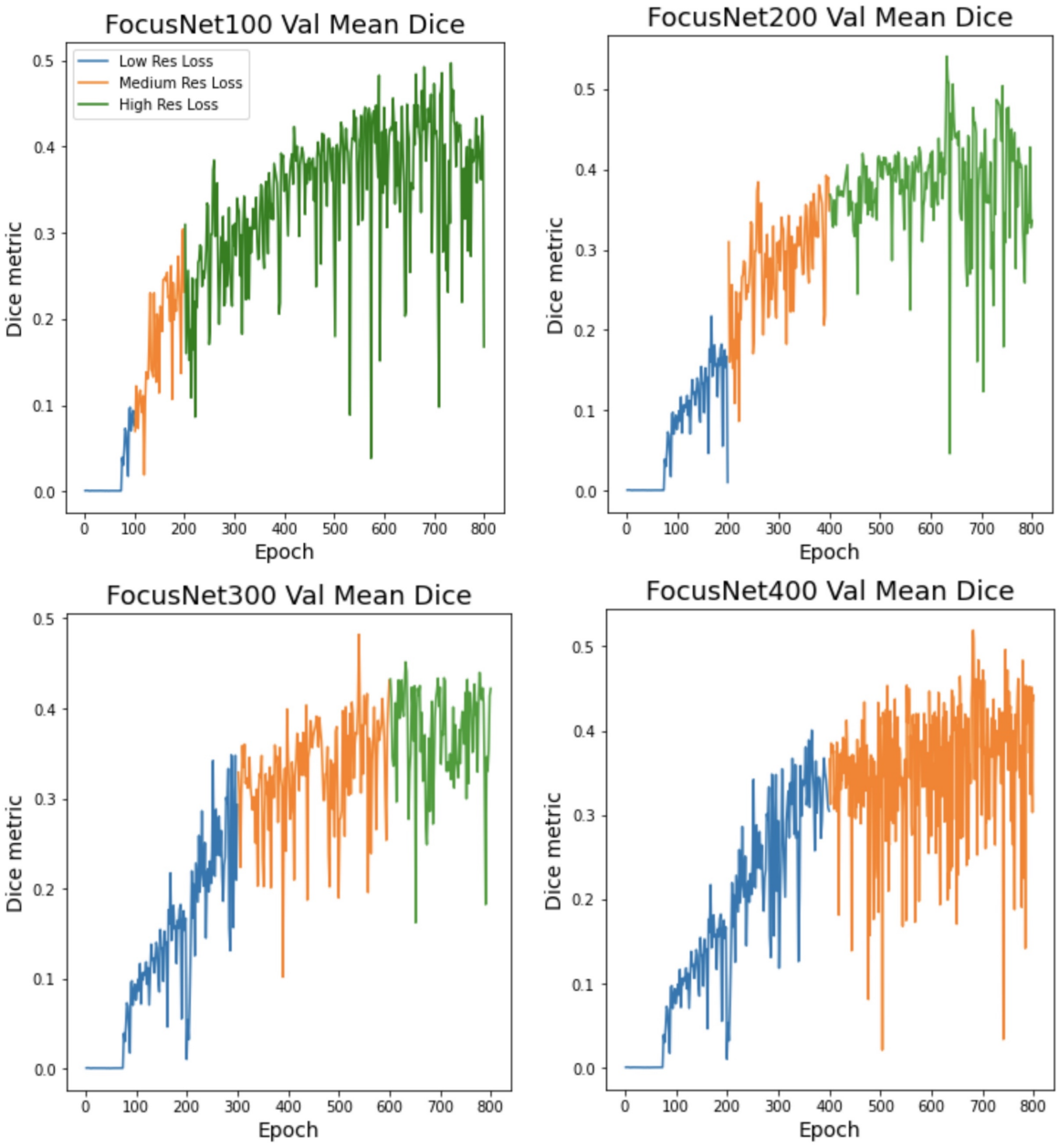


Figure 7.2: Dice metric over 800 training epochs for FocusNets

7.2.2 Analysis and Explanation of Results

FocusNet400 trains using Low Res Loss for the first 400 epochs of training. At epoch 400, FocusNet400 is able to achieve a maximum Dice metric of 0.4020. This is greater than the highest Dice metric achieved by all of the other networks by epoch 400, and demonstrates the fast early convergence of Low Res Loss segmentation to the ground truth labels.

When observing the Dice metric plots for FocusNet300 and FocusNet400, it is possible to see that the gradient of increase of average Dice value drops when switching from Low Res Loss (shown in blue) to Medium Res Loss (shown in orange). From this, one could question why the network is not trained for 800 epochs using only Low Res Loss. This was tested, and the Dice metric obtained was lower than all of the FocusNets.

The reason why it is not promising to use Low Res Loss for all 800 epochs can be deduced when comparing the Medium Res Loss curve (orange section) in all four FocusNets. It can be observed that for FocusNet100, the Medium Res Loss section of the Dice metric plot increases very rapidly. In fact, the Medium Res Loss section in FocusNet100 (epochs 100-200) increases significantly faster than the same epochs range trained using Low Res Loss in FocusNet200. However, in FocusNet400, the Medium Res Loss portion of the plot (epochs 400-800) appears to have a maximum value which is capped, and is not likely to spike high like it does in FocusNet300 at around epoch 550, or like the High Res Loss does in FocusNet200 at around epoch 650. The use of Low Res Loss and Med Res Loss early on during training in FocusNet100 and FocusNet200 appears to allow the High Res Loss (green) portion of the plot to spike better and result in a superior maximum Dice metric. This is likely due to the fact that the loss function is not stuck in local minima which correspond to Low Res Loss minima, which results from over-training with Low Res Loss. Over-training with Low Res Loss hinders performance since the aim of the segmentation task is to achieve the maximum possible Dice metric on the original resolution images.

To confirm that the High Res Loss section of the Dice metric plot is unlikely to spike after a large number epochs of training at Low and Medium Res Losses, the FocusNet300 plot only needs to be observed. Throughout the 200 epochs of High Res Loss training, the maximum Dice metric achieved with Medium Res Loss is not improved upon, since Low Res Loss has been over-trained.

7.2.3 UNet Backbone Results (Experimental Control Variable)

As a control, the Dice metric evolution of the Medical UNet backbone is given in Figure 7.3. The maximum Dice metric reached is lower than all FocusNets.

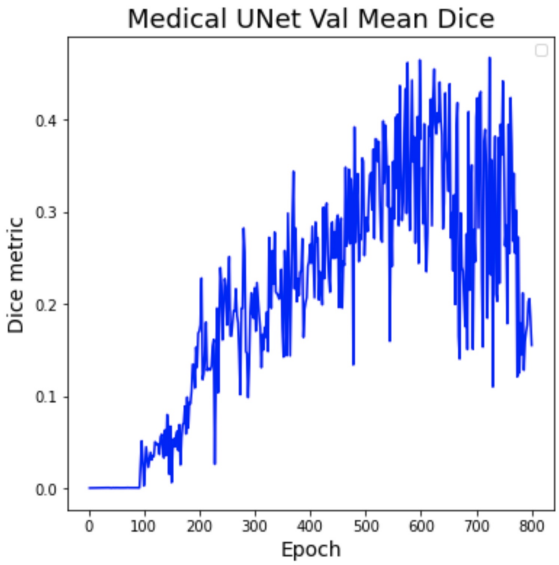


Figure 7.3: Dice metric over 800 training epochs for Medical UNet

A table containing the maximum Dice metrics obtained by UNet and all four FocusNets is given in Table 7.1.

Network Name	Maximum Dice Metric
Medical UNet	0.4669
FocusNet100	0.4971
FocusNet200	0.5405
FocusNet300	0.4726
FocusNet400	0.5191

Table 7.1: Maximum Dice metrics achieved by Medical UNet [27] and all FocusNets

The Dice metric achieved by FocusNet300 in Table 7.1 is lower than expected. This could maybe be re-tested to check the repeatability of the result, or further investigated in future work. However, this network still outperforms the UNet backbone, indicating an architectural improvement.

Chapter 8

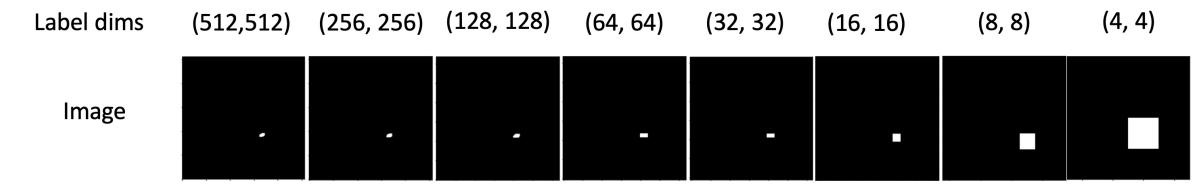
Evaluation

8.1 Exploring FocusNets Further

8.1.1 How Tumour Size Scales with Label Resolution

In Section 6, it was observed that the use of a loss function which is more heavily weighted towards lower resolution outputs is able to pinpoint the ground truth tumour location with a greater accuracy, and give many fewer false positives in other areas of the output prediction. This characteristic of low resolution loss networks was coined as "Focus", and will be numerically analysed in this section of the report.

A label slice containing a tumour was randomly selected from the validation set, and lowered in resolution 7 times, from an initial resolution (512, 512) to a final resolution of (4, 4). Each resolution is inspected to determine the number of tumour pixels it contains, and the number of background pixels it contains. Tumour pixels are represented by ones, and background pixels are represented by zeros. Consequently, the percentage of the label which contains tumour pixels is calculated. The results of this process are given in Figure 8.1.



Label dims	(512,512)	(256, 256)	(128, 128)	(64, 64)	(32, 32)	(16, 16)	(8, 8)	(4, 4)
Image								
No. of ones	294	82	24	8	2	1	1	1
No. of zeros	261,850	65,454	16,360	4088	1022	255	63	15
Ones percentage	0.112	0.125	0.146	0.195	0.195	0.391	1.5625	6.25

Figure 8.1: Percentage of ones, or percentage "tumour" in labels as resolution decreases

From Figure 8.1, it is clear that as the label resolution decreases, the percentage of the label containing a tumour increases. Now, this will be evaluated within the context of the resolutions varying from (96, 96, 96) to (6, 6, 6) used in the final implementation of this project.

8.1.2 Tumour Size Scaling in FocusNets

Since the lowest resolution that is used in LowMCNN is (6,6) image slices, the percentage of ones contained within the image will be between the values for resolutions of (8, 8) and (4, 4) in Figure 8.1. In fact, it will always be equal to $\frac{1}{6*6} * 100 = 2.77\%$. Compared to the original percentage which can vary from 0.05 to 0.8%, this is an increase of anywhere from 3.5 to 55 times. At this stage, it is necessary to recognise that only 2D images have been considered, for ease of representation. If this principle is expanded to 3D images, the enlargement scale at least doubles.

This artificial tumour enlargement phenomenon enables low resolution outputs to be very useful in tackling one of the most difficult problems with lung tumour segmentation: the problem of class imbalance. The tumour and non-tumour classes are significantly more balanced in lower resolution representations. This explains why lower resolution models pay more attention to the tumour class, and focus on segmenting the tumour location correctly, compared to segmenting out the background.

This also explains why LowMCNN predicted tumour size is much larger than the actual tumour size, since the (4,4) image scaled to (512, 512) would contain a tumour of size 128x128 pixels (1/16 of the image), which is equal to 16,384 pixels. The actual tumour occupies 294 pixels, which is only 2% of the 16,384 pixels that would be occupied by the artificially enlarged tumour. Hence, lowering resolution effectively corresponds to enlarging of the tumour by a factor of 50, making it significantly more important in the image.

The higher level of balance between the binary classes that the low resolution labels contain has been shown to be extremely useful for loss function convergence during early training. However, this same effect has been shown to be detrimental to training in later training stages. This can now be explained, since the model is effectively trying to fine-tune to a tumour with a size that is greater than the size of the ground truth tumour. The next section will explain why this is bound to not converge as well as a model which receives the correct sized tumour as a label during training.

8.2 Why High Resolutions are Necessary

Theoretically, a neural network will only be able to learn a perfect tumour segmentation function if it can correctly segment every individual pixel of the tumour. When max pooling with dimensions (2,2) is performed to lower label slice resolutions, the pixel values of four pixels are reduced into a singular pixel value. This has the effect of blurring the border of the tumour, making pixel-precise recovery impossible. This means that once max pooling has been performed only once, it is no longer possible to determine the exact border of a tumour with 100% confidence.

It is clear that the original resolution input is required in order to correctly tune the model to segment the correct border of a tumour, and this is why the full resolution image is weighted the most highly during later training stages.

8.3 Comparison with the best models today

UNETR is a transformer model which has achieved the best performance on the Medical Segmentation Decathlon tasks [19]. A standard version of UNETR was trained for 800 epochs, and the resulting maximum Dice metric obtained on the validation set was 0.6204. This beats all models trained in this project. Although this work has made notable improvements on the original UNet structure for lung tumour segmentation, the overall performance is still inferior to the best model today. However, there are promising future work directions which could leverage the learnings of this project to create models which outperform even UNETR. Potential future work will be covered in the last section of the report.

Chapter 9

Conclusions and further work

9.1 Conclusions

The most basic objective of this project was to demonstrate that an MCNN model losses can converge faster than the loss of their equivalent UNet backbone. This has been demonstrated in both Keras and PyTorch implementations of three dimensional MCNNs. This is the first implementation of a 3D MCNN network, and demonstrates that MCNNs can be successfully applied to data which is more than two-dimensional. Building on this, MCNN parameters have been tested to determine configurations with superior performance. The following configurations have been shown to be most effective:

1. Max pooling is most effective to lower label resolutions.
2. Patch sizes of (96, 96, 96) are a good choice for the Medical Segmentation Decathlon Lung Tumour dataset.
3. A standard one-to-one ratio of positive and negative labels works well for MCNN loss convergence.
4. The number of resolutions used in the final network loss does not matter as much as the dimensions of the lowest resolution. This is because very low resolutions create the largest change in the ratio of foreground to background in the data labels.
5. Loss functions which are more heavily weighted toward lower resolution losses provide faster training convergence at the initial training stages, but do not allow for the Dice metric to spike as much during later training stages.
6. Due to the previous point, the best way to use low resolution losses is to use them mainly during the initial stages of loss function convergence. Higher resolution losses should then be weighted more heavily during later stages of less convergence. This strategy results in a final model with a higher accuracy after training is completed.

These contributions place multi-resolution networks at a stage where their principles can now be applied to all segmentation tasks, both binary and multi-class.

9.2 Further Work

9.2.1 Possible MCNN Design Improvements

In this project, the learning rate was kept constant for training with all MCNN loss functions. It has been explained that MCNN network loss functions which are weighted more highly towards higher resolution loss outputs can be seen as providing the network with fine-tuning. Since fine-tuning is often performed at a lower learning rate, this would be a promising direction to explore.

In this project, the label resolutions for MCNN outputs have been decreased by successive factors of two. It is possible to decrease the resolution by other factors too, e.g. by a factor of 3, or even a factor of 1.5. The correct scaling functions would need to be created to ensure the label occupies the correct number of pixels. If this was achieved, it would result in many more outputs, allowing for a much more gradual transition between different loss weightings, and potentially a much smoother convergence characteristic.

In terms of data processing, much more exaggerated transformations should be experimented with, including aggressive scaling applied the network input images. This could force the high resolution loss function away from local minima, since the percentage of the label which the tumour occupies would vary greatly.

9.2.2 Possible Applications of Multi-Resolution Outputs

Evaluating the performance of networks against other architectures provides an insight into the overall usefulness of the model in practical applications. However, evaluating the performance of networks against the architecture that they are built to improve upon provides a much deeper insight into the benefits of the model from a research perspective. This is why the MCNNs tested in this project were compared to their equivalent UNet backbone.

Transformer models have consistently resulted in the best performance for medical segmentation tasks over the last year. From this project, it has been shown that multi-resolution outputs can aid the task of image segmentation. Due to this, applying multi-resolution outputs to transformer models could be a promising research pathway.

Appendix A

Github Repository

The code for the main implementations and training results for this project are contained within the following Github repository: MCNN FYP Repository (click me!)

Bibliography

[1] 2022. [Online]. Available: https://www.researchgate.net/publication/322772022_A_Review_on_Image_Processing_Applications_in_Medical_Field. [Accessed: 5- Jan- 2022].

[2] "Metrics to Evaluate your Semantic Segmentation Model", Medium, 2022. [Online]. Available: https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2. [Accessed: 5- Jan- 2022].

[3] "Lung Cancer Fact Sheet", Lung.org, 2022. [Online]. Available: https://www.lung.org/lung-health-diseases/lung-disease-lookup/lung-cancer/resource-library/lung-cancer-fact-sheet#:~:text=Early%20detection%2C%20by%20low%2Ddose,percent%20among%20high%2Drisk%20populations.&text=About%208%20million%20Americans%20qualify,with%20low%2Ddose%20CT%20scans. [Accessed: 6- Jan- 2022].

[4] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998. Available: 10.1109/5.726791.

[5] k. Transfer, "Explain Pooling layers: Max Pooling, Average Pooling, Global Average Pooling, and Global Max pooling. - knowledge Transfer", knowledge Transfer, 2022. [Online]. Available: https://androidkt.com/explain-pooling-layers-max-pooling-average-pooling-global-average-pooling-and-global-max-pooling/. [Accessed: 6- Jan- 2022].

[6] Theaisummer.com, 2022. [Online]. Available: https://theaisummer.com/skip-connections/. [Accessed: 6- Jan- 2022].

[7] "U-Net: Convolutional Networks for Biomedical Image Segmentation", Lmb.informatik.uni-freiburg.de, 2022. [Online]. Available: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/. [Accessed: 7- Jan- 2022].

[8] F. Wang, A. Eljarrat, J. Müller, T. Henninen, R. Erni and C. Koch, "Multi-resolution convolutional neural networks for inverse problems", Scientific Reports, vol. 10, no. 1, 2020. Available: 10.1038/s41598-020-62484-z.

[9] B. Ait Skourt, A. El Hassani and A. Majda, "Lung CT Image Segmentation Using Deep Neural Networks", Procedia Computer Science, vol. 127, pp. 109-113, 2018. Available: 10.1016/j.procs.2018.01.104.

[10] A. Vaswani et al., "Attention Is All You Need", arXiv.org, 2022. [Online]. Available: https://arxiv.org/abs/1706.03762. [Accessed: 9- Jan- 2022].

[11] "LIDC-IDRI - The Cancer Imaging Archive (TCIA) Public Access - Cancer Imaging Archive Wiki", Wiki.cancerimagingarchive.net, 2022. [Online]. Available: https://wiki.cancerimagingarchive.net/display/Public/LIDC-IDRI. [Accessed: 9- Jan- 2022].

[12] "Data Usage Policies and Restrictions - The Cancer Imaging Archive (TCIA) Public Access - Cancer Imaging Archive Wiki", Wiki.cancerimagingarchive.net, 2022. [Online]. Available: https://wiki.cancerimagingarchive.net/display/Public/Data+Usage+Policies+and+Restrictions. [Accessed: 11- Jan- 2022].

[13] "Creative Commons — Attribution 3.0 Unported — CC BY 3.0", Creativecommons.org, 2022. [Online]. Available: https://creativecommons.org/licenses/by/3.0/. [Accessed: 11- Jan- 2022].

[14] S. Tran, T. Nguyen, M. Le, C. Cheng and D. Liu, "TDC-Unet: Triple Unet with Dilated Convolution for Medical Image Segmentation", International Journal of Pharma Medicine and Biological Sciences, vol. 11, no. 1, pp. 1-7, 2022. Available: 10.18178/ijpmbs.11.1.1-7.

[15] M. Lin, Q. Cai and J. Zhou, "3D Md-Unet: A novel model of multi-dataset collaboration for medical image segmentation", Neurocomputing, 2021. Available: 10.1016/j.neucom.2021.12.045.

[16] Tran, S., Cheng, C., Nguyen, T., Le, M. and Liu, D., 2021. TMD-Unet: Triple-Unet with Multi-Scale Input Features and Dense Skip Connection for Medical Image Segmentation. Healthcare, 9(1), p.54.

[17] Kushnure, D. and Talbar, S., 2021. MS-UNet: A multi-scale UNet with feature recalibration approach for automatic liver and tumor segmentation in CT images. Computerized Medical Imaging and Graphics, 89, p.101885.

[18] Xu, G., Wu, X., Zhang, X. and He, X., 2022. LeViT-UNet: Make Faster Encoders with Transformer for Medical Image Segmentation. [online] arXiv.org. Available at: ¡https://arxiv.org/abs/2107.08623¿ [Accessed 13 January 2022].

[19] Hatamizadeh, A., Tang, Y., Nath, V. and Yang, D., 2022. UNETR: Transformers for 3D Medical Image Segmentation. [online] Openaccess.thecvf.com. Available at:

¡https://openaccess.thecvf.com/content/WACV2022/papers/Hatamizadeh_UNETR_Transformers_for_3D_Medical_Image_Segmentation_WACV_2022_paper.pdf¿ [Accessed 13 January 2022].

[20] GitHub. 2022. research-contributions/UNETR/BTCV at master · Project-MONAI/research-contributions. [online] Available at: ¡https://github.com/Project-MONAI/research-contributions/tree/master/UNETR/BTCV¿ [Accessed 13 January 2022].

[21] Cite This For Me. 2022. Save Time and Improve your Marks with CiteThisForMe, The No. 1 Citation Tool. [online] Available at: ¡https://www.citethisforme.com/¿ [Accessed 13 January 2022].

[22] Bahdanau, D., Cho, K. and Bengio, Y., 2022. Neural Machine Translation by Jointly Learning to Align and Translate. [online] arXiv.org. Available at: ¡https://arxiv.org/abs/1409.0473¿ [Accessed 14 January 2022].

[23] A. Winkler, "The NIFTI File Format", https://brainder.org/2012/09/23/ the-nifti-file-format/, 2022. [Online]. Available: https://brainder.org/2012/09/23/ the-nifti-file-format/. [Accessed: 15- Apr- 2022].

[24] "3D-Convolutions and its Applications", Medium, 2022. [Online]. Available: https://biplabbarman097.medium.com/3d-convolutions-and-its-applications-6dd2d0e9e63f. [Accessed: 12- May- 2022].

[25] "Data — MONAI 0.9.0 Documentation", Docs.monai.io, 2022. [Online]. Available: https://docs.monai.io/en/stable/data.html. [Accessed: 14- May- 2022].

[26] 2022. [Online]. Available: https://www.researchgate.net/publication/331082965_Left-Ventricle_Quantification_Using_Residual_U-Net_9th_International_Workshop_STACOM_2018_Held_in_Conjunction_with_MICCAI_2018_Granada_Spain_September_16_2018_Revised_Selected_Papers. [Accessed: 11- May- 2022].

[27] "Lung cancer statistics", Cancer Research UK, 2022. [Online]. Available: https://www.cancerresearchuk.org/health-professional/cancer-statistics/statistics-by-cancer-type/lung-cancer#: :text=Lung%20cancer%20mortality,deaths%20(2017%2D2019). [Accessed: 8- Jan- 2022].

[28] "3D-Convolutions and its Applications", Medium, 2022. [Online]. Available: https://biplabbarman097.medium.com/3d-convolutions-and-its-applications-6dd2d0e9e63f. [Accessed: 4- Feb- 2022].

[29] "MONAI - Home", Monai.io, 2022. [Online]. Available: https://monai.io/. [Accessed: 11- Apr- 2022].

[30] "Training with PyTorch — PyTorch Tutorials 1.11.0+cu102 documentation", Pytorch.org, 2022. [Online]. Available: https://pytorch.org/tutorials/beginner/introyt/trainingyt.html. [Accessed: 09- Apr- 2022].

[31] "Inference methods — MONAI 0.9.0 Documentation", Docs.monai.io, 2022. [Online]. Available: https://docs.monai.io/en/stable/inferers.html. [Accessed: 25- May- 2022].

[32] "What Is A Convolutional Layer?", Analytics India Magazine, 2022. [Online]. Available: https://analyticsindiamag.com/what-is-a-convolutional-layer/. [Accessed: 04- Jan- 2022].

[33] "Medical Segmentation Decathlon", Medicaldecathlon.com, 2022. [Online]. Available: http://medicaldecathlon.com/. [Accessed: 12- Mar- 2022].