University of Manchester

MPhys Project Report

Transfer learning using a convolutional neural network based on U-Net (VGG16) to segment primary tumour volumes of cervical cancer patients

Author:
Brijesh Patel
9568975

Supervisors:
Dr Alan McWilliam
Dr Corinne Johnson-Hart

School of Physics and Astronomy
and
The Christie NHS Foundation Trust

May 2019

This project was performed in collaboration with Harry Howell, with special thanks to Dr Andrew Green.

Abstract

In this project, a convolutional neural network based on the U-Net architecture, developed by Dr Andrew Green, was refined to automatically segment the primary tumour volume (gross tumour volume and clinical tumour volume) regions of magnetic resonance image slices in patients with cervical cancer. The image slices used were obtained for 14 patients with up to 4 visits, each with 2 time-points (start and end). The primary tumour regions were then independently delineated manually by a clinician, which constituted the labelled data set for the training. Hyperparameters (such as the learning rate, batch size, number of epochs and optimiser) were adjusted and refined until a model was developed for the gross tumour volume, which produced a mean Dice coefficient value of (0.5124 ± 0.2890) across the test data set, with a maximum of 0.8723 and a minimum of 0.0775. A very similar model was developed too using U-Net to predict the larger clinical tumour volume, which resulted in a mean Dice coefficient of (0.8839 ± 0.0421), with a maximum of 0.9373 and a minimum of 0.7469 across the same test data set.

1 Introduction

1.1 Previous semester review

In the previous semester, a process was developed to extract a large number of quantitative features from MRI images using a Python package called `'radiomics'`. Firstly, a clinician drew contours onto the MRI scans marking regions of interests. `Lua` code was then developed to produce NiFTi files for the masks based on these regions of interest, and to also collate all the individual DICOM image slices into one NiFTi file. Python code was subsequently developed to extract `radiomics` features, and the stable features which changed little over a short time period (i.e. start and end of a visit) were identified. [1]

1.2 Introduction to Artificial Neural Networks

In this semester, the aim was to investigate whether a malignant tumour region could be automatically segmented using deep learning. Clinicians can often end up spending a substantial amount of time manually drawing on the contours slice-by-slice for the tumours so having an automatic method would not only save them time, but the amount of labelled data available for research would increase too. One potential method of providing these segmentations is by using deep learning; a model that is accurate enough could help improve the diagnostic abilities of clinicians by minimising human error and inconsistencies when it comes to image segmentation. The best deep learning model would be not only accurate, but be able to generate segmentations quickly too. [2] The often complex variations between patients as well as the shape, size and location of tumours can make automatic image segmentation a challenging task. Nevertheless, its use in medical imaging has increased recently, especially in the past 5 years due to improvements in technology. [3]

Artificial neural networks (ANN) have been used for many years to segment and classify images. This is a computer vision task which involves the model firstly processing an image and then classifying each pixel through training, which can subsequently be used to produce predictions.[4] Early models were developed to classify dogs and cats based on a sample containing thousands of images in the training sample. [5] ANN uses were however limited until more recently due to lack of enough data and computing power for a given task. Nevertheless, technological advancements and improvements in deep learning techniques have resulted in developments in fields such as speech recognition, machine translation and medical diagnosis with growing success. This concept has also been extended to usage in segmentation of a range of biomedical images too. [6]

There are many architectures for the ANN which can be employed, though convolutional neural networks (CNN) have been gaining popularity due to their simplicity and effectiveness. [4] CNNs have also been shown to outperform other techniques. They were loosely inspired by, and are analogous to, how the human brain is connected by neurons and synapses which transmit signals between each other, enabling the brain to identify objects. [7] In this project, a modified CNN called U-Net was used; this network has previously been used in sectors such as satellite image analysis where each pixel was classified as a building or not-building, with an accuracy of 60%. [8]

In medical imaging, U-Net has been used to segment computerised tomography (CT) scans of organs such as the liver, stomach and gallbladder to accuracies of $(97.1 \pm 1.0)\%$, $(96.1 \pm 7.9)\%$ and $(85.1 \pm 15.7)\%$ respectively (found by using the Dice similarity coefficient). [3] Other models have been used to segment CT images such as the pancreas, which when used with a holistically-nested CNN, produced an accuracy of $(78.0 \pm 8.2)\%$ [9] whereas another group that used U-Net with CT images reported an accuracy of $(82.37 \pm 5.68)\%$ for this organ [10]. U-Net has also been used to segment other organs such as the liver and lung nodules with accuracies in the region of 94% [11] and 95% [12] respectively. One group reported an accuracy

of $(99.72 \pm 0.02)\%$ with U-Net segmenting brain tissues, which even outperformed the results achieved by a human $(99.51 \pm 0.03)\%$. [13] These successes have resulted in U-Net becoming increasingly popular as its largest benefit is that it can produce reasonably accurate results with a small amount of data, unlike most CNN models. [14] This makes it ideal for usage in medical imaging where the size of dataset, and availability of high-quality labelled training data is often limited.

The work completed this semester builds upon the modified U-Net structure developed by Dr Andrew Green who successfully trained it to produce contours for the skeletal muscle region in patients with sarcopenia (progressive muscle wastage) [15]. In our case, for patients with cervical cancer, the primary aim was to use U-Net with a set of pre-trained weights to segment the gross tumour volume (GTV); this is defined to be the "the region in which there is "gross palpable or visible extent and location of malignant growth" [16]. The masks that were predicted for a given image using the model would indicate the probability of the presence of tumour in each pixel.

2 Theory

2.1 Overview

CNNs are generally connected via a series of convolutional and pooling layers containing artificial neurons (nodes) which assign weights to information being passed through it. Transfer learning uses a pre-defined architecture, such as the U-Net-based network built for segmenting sarcopenia regions [15], in order to produce predictions for a similar data set – in this case, the tumour and non-tumour regions of cervical cancer. [17] However, tumours are very difficult to segment compared to normal tissues as they can be very hard to isolate in comparison to normal tissues that generally do not change too much between healthy patients.

When using transfer learning, several pre-trained layers are used with pre-initialised weights from another model, where the coarser features still apply, with subsequent training then performed on the more finer features. [18] Transfer learning is also beneficial when the medical data set is small. However, overfitting whereby a model only works accurately with the original training set and not for new images is a major issue in machine learning models. This risk can be minimised by first splitting the total data set into training, validation and test sets; the training set will refine the model whereas the validation and test sets are used to check for accuracy during the training and predictions.

2.2 Basic CNN structure

A CNN, such as the one in Figure 1, involves an input with subsequent hidden layers performing convolutions and identifying features between them. Each layer contains nodes that multiply input parameters by certain weights and find a weighted sum. [19] This results in increasing complexity with each layer, where early layers may focus on simple features such as points on the image, with latter layers focusing on edges and shapes. U-Net is a modified CNN known as a fully convolutional network (FConv) that only has fully connected layers at the end of the network, enabling for greater flexibility in the training process; layers before this are connected to each other via a small number of specific nodes rather than to all of them in the preceding layer. [20] These specific, 'local' nodes are referred to as the receptive field of the node concerned. [21]

On the other hand, many CNNs are fully connected networks (FCNs)- whereby each node is connected to every node in a subsequent layer rather than those only in the receptive field. In the case of image segmentation, these are impractical as the number of neuron connections would grow significantly with more layers, even for a small image. [22] An FConv also has the

benefit of preserving spatial information through upsampling and downsampling steps, as nodes are connected in the receptive field. [3]

The weights are activated through an activation function which transform the input values and then outputs them, in order to be used in the next layer of nodes. Backpropagation, via a cost function, is then used to adjust the weights through a process called 'gradient descent' by comparing the prediction values with actual values. [4] This process is repeated many times until the weights provide accurate predictions for new data. U-Net, which is a type of FConv, uses supervised learning to train the network – i.e. it learns through using labelled inputs, or in our case, images which have associated drawn masks/delineations. [23]

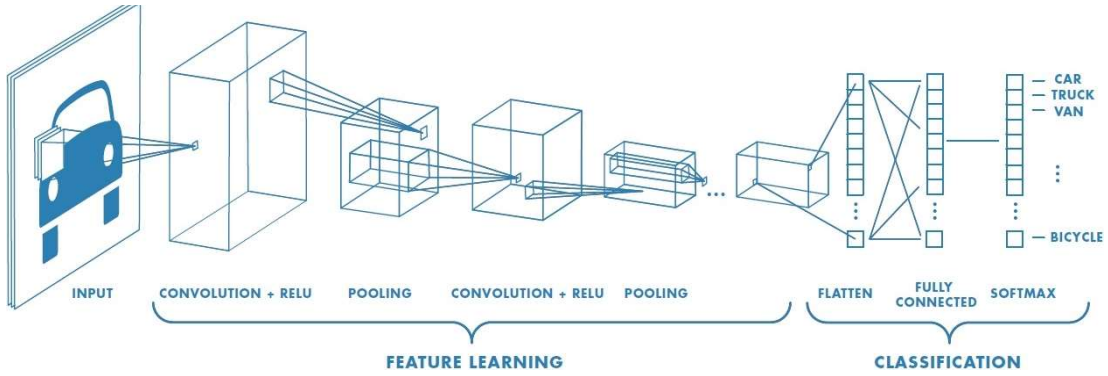

Figure 1: A diagram to show how a CNN classifies a pixel through convolution, `ReLU` activation and pooling layers. [24] The specifics of which have been provided below in the context of a U-Net network. The feature learning part will detect increasing complexity in the images; the fully connected layers in the classification use the `softmax` function to normalise the values output into a probability distribution and ensures that the sum of probabilities for the classifications total to 1. [25]

2.3 U-Net: Downsampling

A U-Net-based network, like many FConv networks, consists of two main steps: upsampling and downsampling. In the first step, an image – such as a magnetic resonance imaging (MRI) slice – is input and undergoes repeated 3×3 convolutions which halve the dimensions of the feature maps in each layer, as shown in Figure 2. After each convolution, a rectified linear operation (`ReLU`) and a 2×2 max pooling operation is applied. [14] `ReLU` is a function which defines how the weighted input at a node is transformed before being used in a subsequent node. [26] The max pooling operation involves selecting the maximum pixel value in each 2×2 grid that is scanned across the image; this will essentially reduce the number of parameters in the model, allowing it to be more easily processed by the computer. [27] In the downsampling process, the size of feature maps are reduced (when no padding is used during the convolutions) but the number of feature maps per layer is increased. [28]

During the convolutions, the weights are updated via backpropagation. [29] As the kernel (filter) is scanned across the input image, an output value is generated corresponding to the sum of the element-wise products in the input-kernel convolution as shown in Figure 3. [30] The stride length used by U-Net is set to 1, which means that the kernel will slide by 1 unit across the input matrix (horizontally and vertically), so every value is considered. The kernel used can emphasise elements such as edges and shapes (depending on the values of the matix), and the `ReLU` layer will filter these before passing them to the next layer. [26]

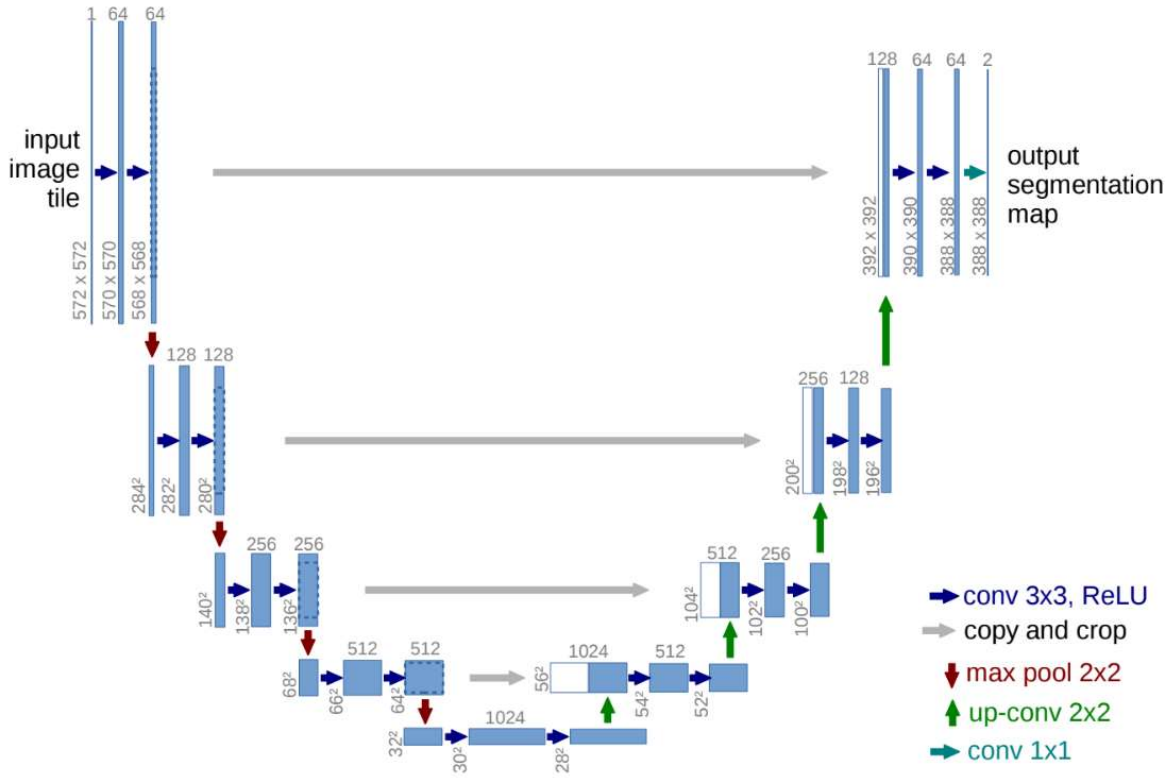

Figure 2: A diagram to show the downsampling and upsampling stages for U-Net; numbers at the top of each bar represent number of features and numbers on the side of each bar represent dimensions of feature map. [14] Downsampling consists of two 3×3 transposed convolutions and one 2×2 max pooling between layers. Upsampling consists of one 2×2 convolution followed by two 3×3 convolutions between layers. [14] The downsampling path is sometimes known as an encoder; each deconvolution 'block' and pooling combination results in the input image gaining different feature representations at multiple levels. The upsampling path is the decoder where the downsampling features are concatenated with the upsampled features. [31]

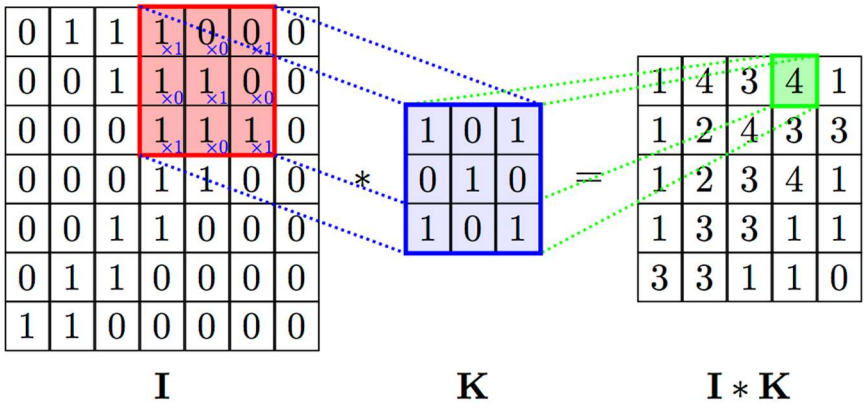

Figure 3: A diagram to show how the convolution procedure works (for the downsampling segment of U-Net). [32] The convolution occurs between the image, I, and the kernel, K to produce the resulting image, which has a reduced size compared to the input image (i.e. there is no 'padding' to preserve the original dimensions). This final matrix is referred to as a feature map, or activation map, and is generated to try and extract only relevant features from the input image. Multiple feature maps are generated using different kernels (though spatial relationships are still maintained with the input values), which all extract different information and these are what make up a layer [33].

2.4 U-Net: Upsampling

A 2×2 transposed convolution is applied after each layer, as shown in Figure 4, and this is followed by 3×3 transposed `ReLU` convolutions which aim to recover the original size of the images. In this process, the feature maps dimensions are doubled after each layer. [14] These upsampled feature maps increase spatial information but reduce feature information and so are thus combined with the higher resolution features extracted from the downsampling process, as exemplified by the horizontal arrows in Figure 2. [31] The final output produced is a prediction – in our case, a mask – which has classified each pixel in the input image. In our case, each pixel was assigned a probability of being a GTV region, and a threshold could also be set to convert it into a binary mask if needed.

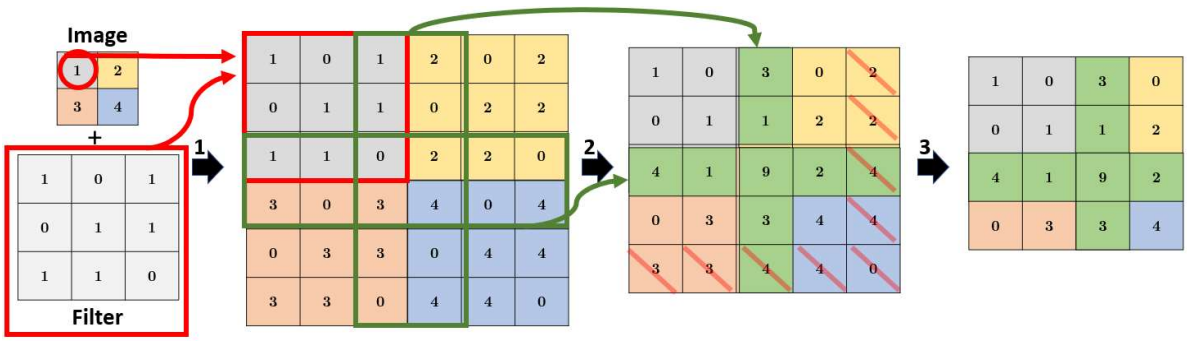

Figure 4: A diagram to show the stages of the transposed convolution process during the upsampling part of U-Net (created using Microsoft PowerPoint). The filter is applied to each of the pixels in the image element-by-element (step 1). The resulting 4 matrices are combined with the values summed when overlapped (step 2). The last row and column are removed resulting in a cropped matrix (step 3). [34]

2.5 Activation and Pooling

During the downsampling process, there are two layers for activation and pooling. The `ReLU` (rectified linear units) activation layer will increase the non-linearity of the CNN model which is necessary as there will be a nonlinear relationship between input and output values; i.e. the input image and the resulting predicted mask. [26] Using linear activation functions have been shown to not be as effective for complex media. [35] At a certain node, the output is defined by the product of the input value and the weights are summed, with a bias added at the end. The bias will shift the output values to provide a better fit for the data. [36]

Gradients of the network are then found using the activation function during the back propagation whereby the derivatives are found layer-by-layer; these gradients are used to adjust the weights during this process. For `ReLU`, input values >0 are equal to the output values, whereas values ≤ 0 are output as 0, as shown in Figure 5. All negative values are hence eliminated. [26] `ReLU` is most commonly used (compared to `sigmoid` and `tanh`) and has been found to be highly effective in many models; this is often the activation function used by default [26].

Pooling is used to reduce the number of features present. This is necessary to achieve 'translational variance' – i.e. to account for the movement of the object (tumour) between input images which will, as a result, also reduce the chances of overfitting the data set and increase the model's ability to generalise.[37] The most common form of pooling is 'max pooling' as shown in Figure 6, whereas the other form is known as 'average pooling'.[38]

Max pooling is more appropriate for the task compared to average pooling (which would average the values in each mini matrix rather than find the max value [39]) when the `ReLU` activation function has been used; only the non-zero values are relevant and only these should

be used with this activation function when reducing the dimensions, in contrast to the average pooling procedure which would utilise the irrelevant zero values as well.

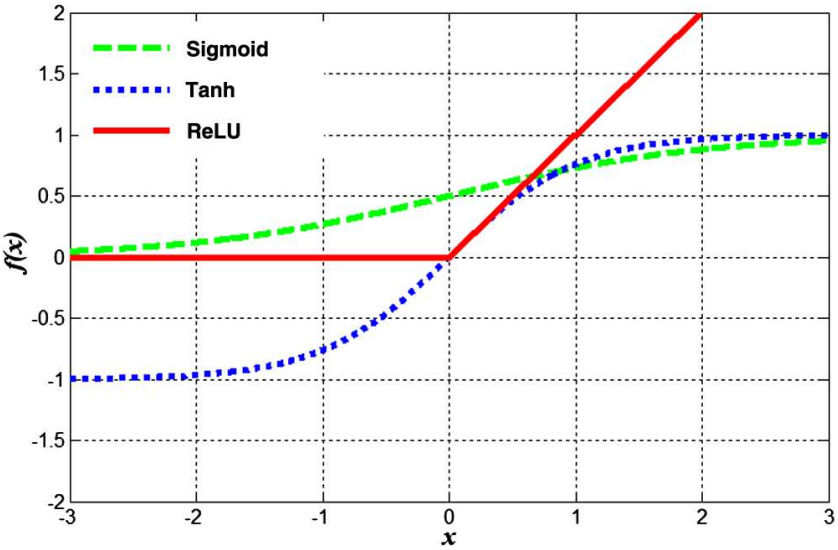

Figure 5: Graphs to show the variation between 3 different activation functions: `Sigmoid`, `Tanh` and `ReLU`. The x represents the input value at the node, and $f(x)$ is the output value from the node, once the activation function has been applied for a given parameter.

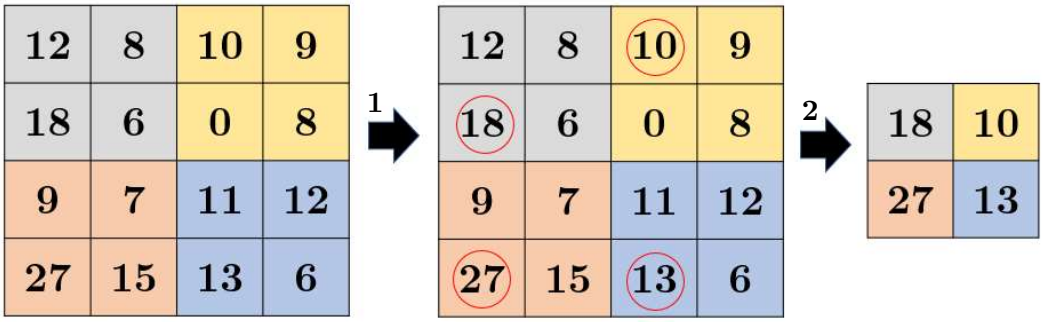

Figure 6: A diagram to show how the max pooling operation is used in U-Net (created using Microsoft PowerPoint). The image is first split into mini 2×2 grids (step 1). The maximum value (circled) is then chosen in each grid, resulting in the output which contains only the maximum values (step 2). An average pooling step would instead find the average rather than the maximum value in each mini 2×2 grid in step 1. [39]

2.6 VGG-16

Before training commences, CNNs such as U-Net would usually be initialised with random weights so millions of images would be required in the training to produce a model that generated accurate predictions [40], without over-fitting the data. Hence, a pre-trained network such as VGG-16, that is based on a modification of U-Net, can be used to provide weights. [41] These weights were previously trained on a database called ImageNet which consisted of 15 million images that were split into 22,000 categories [5], which were then split further into training, validation and test sets. The VGG-16 weights are used in the encoder (i.e. downsampling portion of the CNN) so only the decoder (upsampling) needs to be trained generally. [42] In VGG-16, there are 13 convolutional layers and 3 fully connected layers, with each followed by a `ReLU` activation, and a total of 4 max pooling operations.[43]

2.7 Weight modifications

2.7.1 Dice loss function

A loss function calculates the error between the prediction and actual value for a given data point. [44] For example, the 'Dice loss' function (DLF), which is based on the Dice coefficient, finds the level of overlap between two samples – i.e. the prediction and actual (target) mask

values. A Dice value of 1 represents perfect overlap whereas 0 is no overlap. [45] This is calculated by,

$$\mathrm{DLF} = \frac{2 \mid A \cap B \mid}{\mid A \mid + \mid B \mid}, \tag{1}$$

where A is the prediction matrix, B is the target matrix, $\mid A \mid$ and $\mid B \mid$ are thus the number elements in each matrix, and $\mid A \cap B \mid$ is the number of common elements between these two matrices. [45]

For simplicity, the $\mid A \cap B \mid$ term can be approximated by performing element-wise multiplication between the values of the predicted and target masks [45], followed by finding the sum as shown in Figure 7.

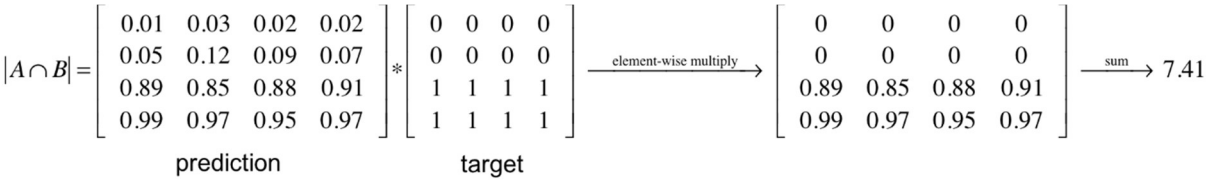

Figure 7: A diagram to show how the value for $\mid A \cap B \mid$ is calculated between two sets of values: a matrix containing the values for each of predictions (A) and the target mask (B) is multiplied together element-by-element and then summed. During the training, the prediction matrix will not be binary (unlike the target matrix), as having non-binary values will be more useful for when the weights are adjusted during the optimisation/backpropagation process. [46]

Values for $\mid A \mid$ and $\mid B \mid$ are generally found by finding the summation of squared values in each of the predictions and target matrices. [47] The factor of 2 in (1) is required since the number of common elements in the denominator is counted twice which needs to be accounted for. [46] Also, for convenience, the 'soft Dice loss' is used (simply by subtracting the DCE from 1) to give the 'loss' and this is what is minimised during the training process [48].

2.7.2 Cost function and gradient descent

In general, a cost function is calculated by finding the mean average of the loss functions. [49] In the context of a CNN, this is performed after an iteration in the training and the best model will minimise this cost function, resulting in the model containing weights that produce the minimum error in the predictions. The cost function uses gradient descent which is an algorithm that attempts to identify the local or global minima of a function. [50] Through iterations, the model begins to converge towards a point where the loss is minimum, which in turn changes the weights of the model slightly through backpropagation. [51] The speed at which the gradient converges is controlled by the learning rate factor which determines the step size as shown in Figure 8; a learning rate too large may miss the local minima (causing the loss function to increase) but a value too small may take a long time to converge (though the cost function may still continue to slowly decrease). [52]

2.7.3 Modified forms of gradient descent

The simplest form of gradient descent is 'batch gradient descent' which will pass the entire data set simultaneously into the network, calculate all the loss function values together and finally, will update the weights. [53] Hence, the CNN can often be updated with very few iterations. However, this can be very difficult to achieve with large data sets, which would require significantly more RAM, GPU RAM and faster CPU speeds, thereby making this

process quite slow. [54] The average gradient for each parameter is calculated during batch gradient descent which is subsequently used to update the weights.

The next form of gradient descent is called 'stochastic gradient descent' (SGD) whereby each sample is passed through the CNN one at a time, resulting in one update to the weights in each pass. This is generally much quicker than before, especially on less powerful computers. Nevertheless, as the loss function is calculated based on one training example, the average loss per iteration may lead to large variations, especially if there are anomalous values. [53] Thus, in this project, 'mini-batch gradient descent' has been used; a small number of samples (batch) are passed through the CNN, with gradients of the loss function calculated after each batch is propagated. The average gradient of the loss function is then calculated for these samples, which subsequently updates the weights of the neurons through back propagation. In each set of iterations (epoch) [51], the step in the epoch which results in the minimal loss in the cost function is selected to update the weights. [53] This method is most commonly used in CNNs and offers the best compromise between the two previous approaches.[55]

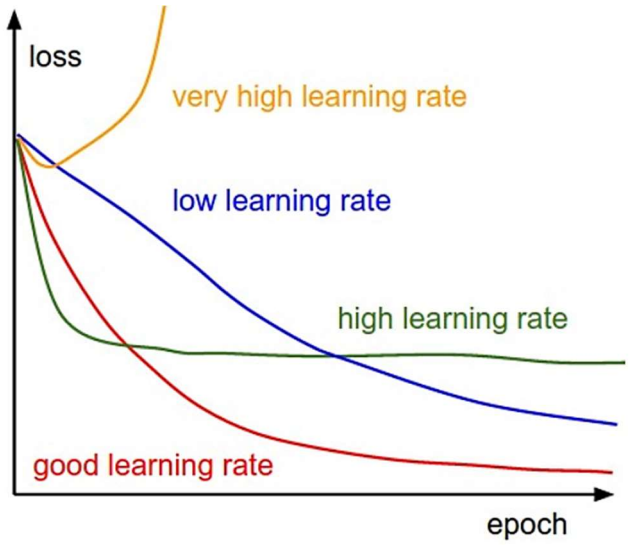

Figure 8: A graph to show amount of loss (i.e. high discrepancy between actual and predicted values) as a function of epoch (iteration) based on different learning rates. [56] Gradient descent is a first-order iterative optimisation algorithm which identifies the local/global minima of a function by moving in the direction of the negative of the gradient at that point. [57] The size of steps determines the speed and accuracy at which the curve converges or diverges. Small learning rates have nearly linear improvements whereas large learning rates are exponential. [56]

2.7.4 Optimisers

Generally, backpropagation is an automatic task when using machine learning packages, including those in Python. This is governed through an optimiser, which defines how the gradient descent is performed and one of the most common ones used is adaptive moment estimation (more commonly known as `Adam` [57]). This optimiser employs an attribute called 'momentum' which enables it to converge on the global minimum more quickly; here, a fraction of the previous update vector (coefficient of momentum) is added to the current update vector, which allows it to build up 'momentum' by amplifying the speed of convergence in the correct direction, towards the minimum point. [58]

An adaptive momentum is a modification that means that for each parameter, the mean and the uncentred variance (where mean is not subtracted in the calculation) of the gradients is calculated which determines the training rate. [59] A gradient descent method that involves a constant step size that does not change could cause a gradient to land on a point and overshoot the minimum, unless the parameters were adjusted constantly. [60] In addition, as there is less manual fine-tuning required for adaptive momentum, it allows for a more automated process. `Adam` also uses an adaptive learning rate which dynamically changes the 'effective step size' of the gradient descent. [57] The mean and variance are two quantities (moments) that are used to generate an exponentially weighted moving average using the previously calculated gradients in order to calculate the new weights. The estimate for moving average of the mean gradient [57] for a given parameter, m_t, is given by,

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \ , \tag{2}$$

where m_{t-1} is the mean gradient from the previous step, β_1 (decay of running average factor) is 0.9 and g_t is the gradient in the current step, at iteration, t. The uncentred variance of the gradient, v_t, is likewise given by,

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \ , \tag{3}$$

where β_2 is 0.999 and v_{t-1} is the variance from the previous step. Values for β_1 and β_2 were set as recommended and do not usually have to be changed. [57] This optimiser has been found to be very efficient in the speed of training compared to other optimisers in most cases. [57] The update to the weight, w_t, can be given by the formula,

$$w_t = w_{t-1} - \alpha \frac{m_t}{\sqrt{v_t} + \epsilon} \ , \tag{4}$$

where ϵ is 10^{-8} (to prevent division by 0) [61], w_{t-1} is the previous weight, and α is the initial learning rate that is set at the start of the code. The second term is can be called an 'effective learning rate' as this directly alters the overall size of the updates. [57]

3 Method

3.1 Initial Steps

3.1.1 Data collection

For the purpose of this project, MRI slices were gathered for 14 patients with cervical cancer (each who had several visits to the hospital) from a radiotherapy treatment planning system (known as Raystation). This data consisted of T2-weighted MRI images in the DICOM file format. The contours were manually drawn by a clinician and stored into separate structure files. The delineations of interest were obtained for the GTV and also the low-risk clinical tumour volume (CTV-LR). The CTV-LR is defined to be the "tissue volume that contains a GTV and subclinical microscopic disease which has to be eliminated" [16] plus where there is "potential microscopic tumour spread." [62]

3.1.2 Software and hardware specifications

Python 3.5 was used due to its compatibility with the machine learning-based packages `keras` [63] and `tensorflow` [64]. The `tensorflow` is a backend to the `keras` module and is significantly more efficient when an NVIDIA CUDA-enabled GPU is used, rather than the default CPU-variant of the packages. Hence, a computer with 8GB RAM and 4GB NVIDIA GTX 970 GPU were used during the training. The VGG-16 CNN was used in the transfer learning process and even though an image of any size could be passed as an input when providing predictions, it needed to use images with the same dimensions in each of the x- and y- directions during the training.

3.1.3 Pre-processing

In-house developed software called WorldMatch [65], along with ZeroBraneStudio (using `Lua`), were first used to extract the MRI slices from a set of 14 patients, where each patient had scans for 2 time-points in each visit (i.e. at the start and the end), with each patient having up to 3 visits in total. These MRI slices were filtered and so only slices which had the GTV region were combined into NIfTI files, using scripts from the previous semester. [1] The images

were also cropped to a region that contained not only the GTV, but also some background which was defined by a square box that encompassed the CTV-LR, which was naturally larger than the GTV region, as shown in Figure 9.

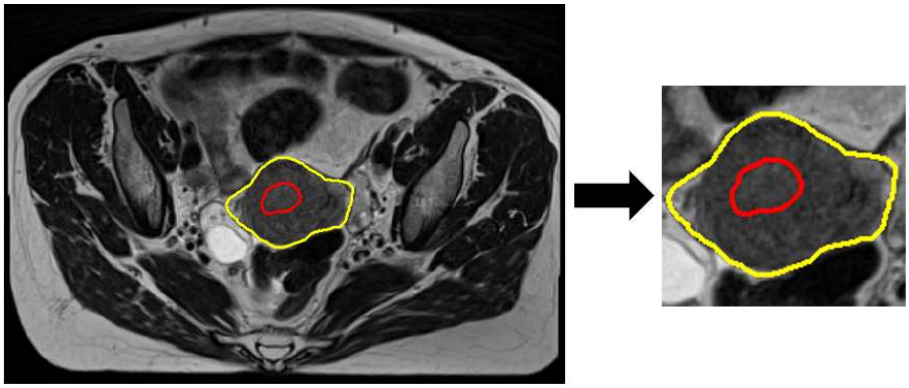

Figure 9: The cropping procedure applied to generate the square region that would be used when training the model. The GTV (red) and CTV-LR (yellow) regions are outlined.

The maximum extents of the largest CTV-LR region was identified in the entire data set (990 slices) to ensure that each NIfTI file would contains images of dimensions 160×160. The cropping was necessary to ensure the images were less resource-intensive when processing them in the machine learning model and would allow the code to run more efficiently. This process was repeated for the corresponding clinician-drawn delineations, producing binary masks which had 2 channels – one which classified a voxel as a tumour (with value 1), and the other as a non-tumour (with value 0). These would be used as the training labels. The NIfTI files were then split into training, validation and test sets using a random number generator. The training set comprised of 832 slices (~85% of the total), the validation had 131 slices (~10% of the total) and the test set had 27 slices (~5% of the total).

Python code was developed, using the `SimpleITK` [66] package, to convert these NIfTI files into multi-dimensional `NumPy` arrays that represented the intensities within the slices. The intensities for these greyscale MRI images ranged from 0.0 (black) to 1.0 (white) for each voxel. This is so they could be processed by the subsequent scripts for training and analysis. Each `NumPy` array was a rank order-4 tensor and consisted of 4 dimensions – image slice number, x-dimension and y-dimensions in the axial plane of each MRI slice, and channel number. For simplicity, each of the 3 channels for each slice would be the same and enabled compatibility with the Python scripts; they could, however, be adapted through applying noise and or by adding filters to emphasise different lines and shapes that could not easily be seen by eye on the raw images. In the end, this resulted in 6 `NumPy` arrays (exported as `.npz` files) in total; each of the 3 folders (training, validation and test) contained 1 collated image array and 1 collated mask array.

3.1.4 Training

To perform the transfer learning, weights for the VGG-16 network were downloaded for use through `keras` and the training image dimensions were set to 160×160. The modified CNN using the `keras` package consisted of sequential layers that represented the upsampling and downsampling steps. The training would occur during the backpropagation steps of the CNN, for which the validation set would be used to evaluate the model once the training had finished by comparing against the model parameters. In essence, the training data set was used to adjust the weights of the CNN whilst the validation set was used to prevent overfitting; these validation images were not trained on during validation but were used to verify whether any increase in accuracy during the training phase also applied to data the CNN has not seen

before. If the accuracy in the training set increased over time but decreased (or remained constant) during validation, then this would be a sign of overfitting. The test set was used with the final model by assessing the accuracy of the predictions with unseen data.

3.1.5 Hyperparameters

The training was split over two steps (both containing a training and validation stage) and each had a different initial learning rate. The first step had a high learning rate which would get very close to the minima whilst the second step would converge more closely. The `Adam` optimiser would automatically refine the learning rate if needed though [57] and the hyperparameters (i.e. starting parameters) used in the code have been detailed below:

- Batch size, y – number of training samples which were simultaneously selected to propagate throughout the CNN. Batches of data were passed through the network as it was unfeasible to pass the entire data set in one iteration, due to computational limitations. A larger batch size can often result in faster training times but would require more GPU memory [67]. Initially, the batch size was set to $y = 16$.
- Epochs, s_1 and s_2 – each consisted of n steps, with a total data set of $z = n \times y$ passing through it in one forward pass. The cost function, as determined by the optimiser, was calculated after each step and an average was found once all samples had been passed through. Initially, the step size was set to $n = 52$ and number of epochs set to $s_1 = 10$ and $s_2 = 10$ for the two steps.
- Learning rate, lr_1 and lr_2 – this determined how much weights were updated at the end of each epoch, by controlling how quickly the model learned. In step one (epochs 1 to 10), $lr_1 = 0.01$ and in step two (epochs 11 to 20), $lr_2 = 0.001$.

3.1.6 Optimiser

A soft Dice loss function was used, as described in Section 2.7.1. The `Adam` optimiser was used to modify the weights during the backpropagation process, as described in Section 2.7.4.

3.1.7 Mask output

The `softmax` function was used to generate the predicted probabilities in the final target matrix, as described in Figure 1. This would happen in the penultimate layer inside the CNN. A `NumPy` array of the same dimensions as the input image file was generated with each value corresponding to a probability of being tumour or not. A Dice coefficient was calculated for each slice in the test set to compare the similarities between the target and predicted masks. In addition, Python code was developed to plot Receiver Operating Characteristic (ROC) curves to compare the true positive rate (proportion of points correctly classified as tumour) and false positive rates (proportion of points incorrectly classified as tumour) between the prediction and target values for various thresholds. The threshold determined the minimum probability required for a voxel to be classified as a tumour.

3.2 Strategies to improve predictions

3.2.1 Adjustment of hyperparameters

The initial hyperparameters specified in Section 3.1.5 initially produced very poor predictions with no discernible contours so were altered to try and generate reasonable predictions. Accuracy would increase with a greater number of iterations, before overfitting occurred and would also increase with a larger batch size (due to the average gradient of more data being calculated) – although this could only be raised to a maximum of 16 due to memory restrictions on the computer. Using larger mini-batches would result in larger step sizes in the gradient descent, enabling the optimisation algorithm to make progress more rapidly. Hence, the batch size was thus varied from 2 to 16 in increments of 2 since powers of 2 produce the best efficiency

on most hardware setups. [55] The learning rate was also varied between a fast value (0.1) and a slow one (0.001) for each of the two steps.

3.2.2 Image augmentations

The 3 channels were initially defined to be the same. To try and improve results, a Gaussian filter was tested though for one channel. This accentuated the contrasts and lines within the image, and provided improved edge detection, as shown in Figure 10. Moreover, a function within the code was implemented to produce rotations on the input slices; this augmented the `NumPy` arrays originally imported so they could be used as additional, artificially-generated training samples in the batch.

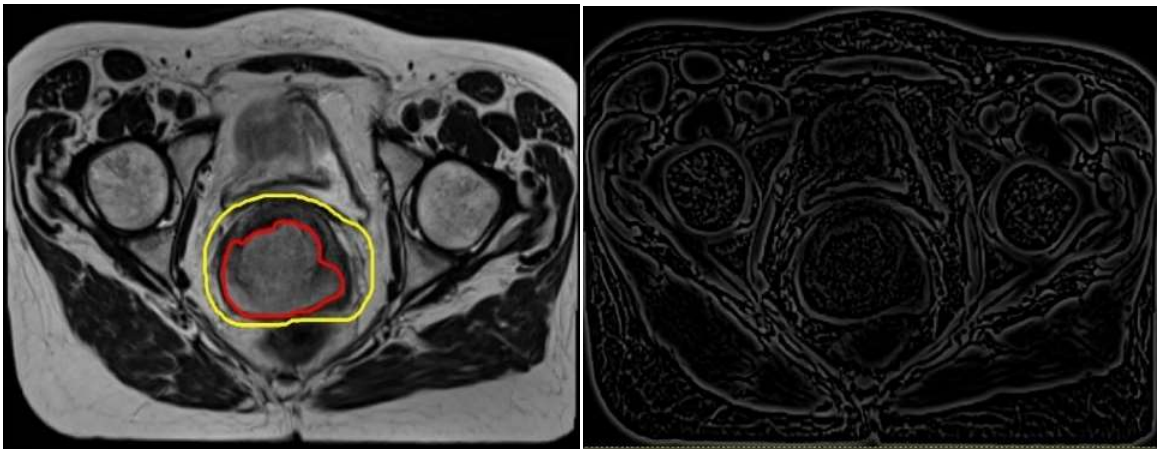

Figure 10: A comparison between one of the original slices that has the GTV contour (red) and CTV-LR contour (yellow) drawn on the left; the same slice which has had a (difference of) Gaussian filter applied to it using the GIMP software is shown on the right.

3.2.3 Deliberate overfitting

Using the Python scripts from before, training and validation data NifTI files for 5 slices each were generated. The aim was to see if the model could be trained to work with a very small data set and produce predictions for previously seen training data. However, despite adjusting the hyperparameters above (such as increasing steps per epoch and number of epochs, with high and low learning rates), the results were still inconclusive and there was no significant improvement of results. This indicated a fundamental issue with the model that could not necessarily be solved by simply adjusting the hyperparameters.

3.2.4 Unfreezing layers

Within the imported VGG-16 model, some layers were frozen so could not be adapted – this ensured some of the learning from ImageNet was fully retained. By unfreezing some of the earlier layers, this gave further flexibility to the training model to modify more weights from previous layers. This, combined with the aforementioned image augmentations, would allow this extra artificially generated data to be used more effectively.

3.2.5 Alternative activation functions

`ReLU` was used by default for the activation but this activation function can be fragile because it may update in such a way that so the gradient remains 0 indefinitely, resulting in no changes to the gradients (known as 'the dying `ReLU` problem'). [26] This may have been the case, so the `sigmoid` and `tanh` functions were used to see if there was any improvement in results. Due to the curve for sigmoid, as shown in Figure 5, small changes in the input value, x, would

result in a large change in the output value at the node, $f(x)$. Hence this would push the input values at the nodes to the extremes and is desirable when classifying pixels.

The `tanh` function is similar, but the gradient is steeper compared to sigmoid, and can also produce negative output values unlike `ReLU` and `sigmoid`. Nevertheless, both of these functions suffer from the vanishing gradient problem; the gradient tends towards 0 for very large and very small values, which means little learning can occur in the lower layers, so the training can become very slow, requiring a substantial number of epochs. `Sigmoid`-like functions can also have slower training times compared to ReLU, due to the latter being less computationally expensive with simpler mathematical calculations. [26] Nevertheless, as U-Net had a small number of layers, this was not important, and the `tanh` activation resulted in best model with the most reasonable predictions when ran for a small number of total epochs (20).

3.2.6 CTV segmentations

The method specified in this section was adapted to train the model on the CTV tissue, instead of the GTV. The slices used were cropped to the same size as before (160×160) and the CNN was trained on the CTV delineations, with the same-sized data sets as before. This was conducted in order to verify whether the difficulties in segmentation were with the GTV, or with elements of the U-Net model. The results for the CTV also served as comparison against the GTV to see if the CTV (which had a more discernible boundary) was indeed easier to delineate with this architecture.

4 Results

4.1 Original predictions (using `ReLU`)

As mentioned, the initial U-Net CNN resulted in very poor predictions with no discernible contours and was unsuccessful in isolating the GTV regions. However, once the activation function was changed from `ReLU` to `tanh`, results improved considerably. Examples of the parameters that were tested for both the GTV and CTV have been provided in the Appendix (Table 1).

4.2 Working model (using `tanh`)

The following hyperparameters produced the most reasonable results for the GTV:

- Training: $s_1 = 10$, $s_2 = 20$, $n = 52$, all layers are trainable
- Image augmentations: no rotations, no Gaussian filter
- Optimiser: `Adam`
- Activation: `tanh`
- Loss function: soft Dice
- Learning rate: $lr_1 = 0.001$, $lr_2 = 0.0001$

The CTV was also delineated with the only change to the above being $s_2 = 10$. Even though the training period was shorter for the CTV, it was found that the CTV produced considerably more accurate delineations compared to the GTV, which will be discussed in Section 5.

A Dice coefficient value was calculated for each slice in the test set (comparing the target and predictions), and the slices with the best and worst values are shown in Figure 11 for the GTV and Figure 12 for the CTV. For the GTV, the Dice coefficient values ranged from 0.8723 to 0.0775 and the mean was calculated to be (0.5124 ± 0.2890) for thet test data set. In addition, the accuracy and loss (1-accuracy) were calculated at the end of each epoch by determining the average of all the loss functions calculated during each epoch. The graphs to show these

changes have been plotted in Figure 13. There is an increase in accuracy of the training for the first 10 epochs, which then plateaus whilst the validation accuracy does not increase at all. In the second set of 20 epochs, the validation and test accuracy continued to improve.

For the CTV delineations, the mean was calculated to be (0.8839±0.0421) with a minimum of 0.7469 and a maximum of 0.9373. Clearly, this model seemed to provide superior and more consistent predictions compared to the model trained for the GTV. The graphs for the accuracy and loss can be found in Figure 14; compared to the GTV, it can be seen the CTV model trained continuously with smoother changes and less visible fluctuation.

The ROC curves in Figure 15 show less variance in the curve shapes for the CTV compared to the GTV. The curves for the CTV have higher true positive rates and lower false positive rates indicating this model is superior in correctly identifying the region of interest (i.e. CTV) as the CTV whilst also minimising the number of regions where the model incorrectly identifies the non-CTV regions as CTV, for a range of given thresholds. In addition, the Area Under Curve (AUC) is higher for the CTV which indicates the CTV model is more accurate in its predictions in distinguishing between the CTV and non-CTV region, compared to the GTV and non-GTV regions in the other model.

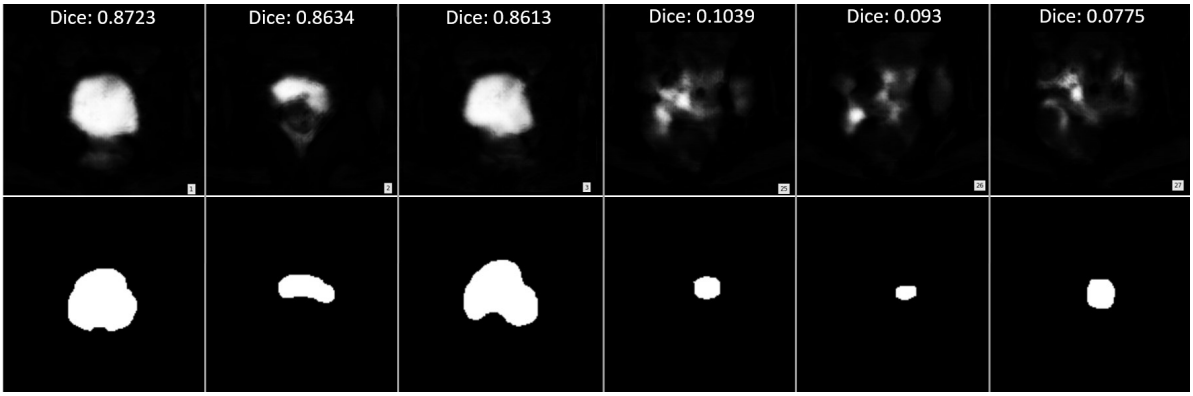

Figure 11: A collection of the GTV slices showing comparison between the predicted masks (top) and target masks (bottom) for the test data set. The first 3 columns show the masks which yielded the best Dice values, whereas the last 3 columns show slices which had the worst values. The predictions are not binary and the contrast scales with probability (i.e. brighter regions have a higher probability of being the GTV).

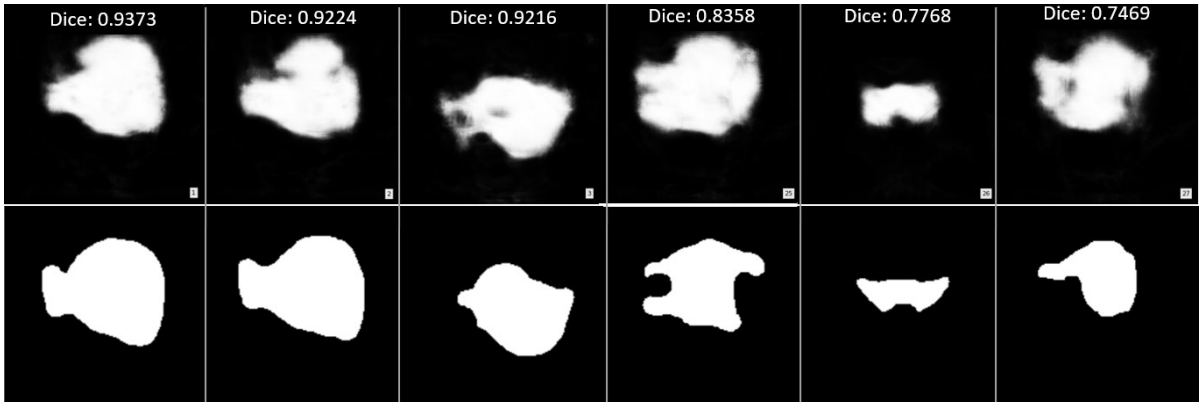

Figure 12: A collection of the CTV slices showing the comparison between predicted masks (top) and target masks (bottom) for the test data set. The first 3 columns show the masks which yielded the best Dice values, whereas the last 3 columns show slices which had the worst values. The predictions are not binary and the contrast scales with probability (i.e. brighter regions have a higher probability of being the CTV).

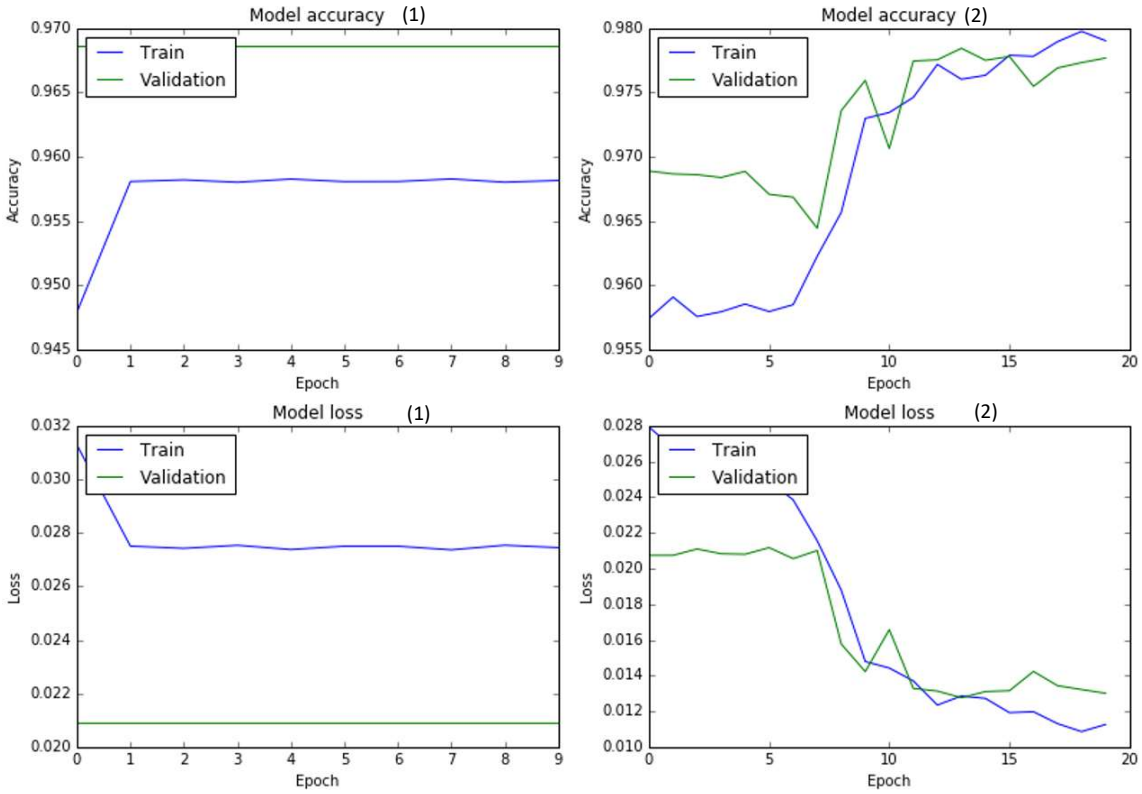

Figure 13: Graphs to show how the model accuracy and loss changed for the training and validation data sets for the **GTV**. The first step of 10 epochs ($lr_1 = 0.001$) are on the left, and the second step of 20 epochs after step 1 ($lr_2 = 0.0001$) are on the right.

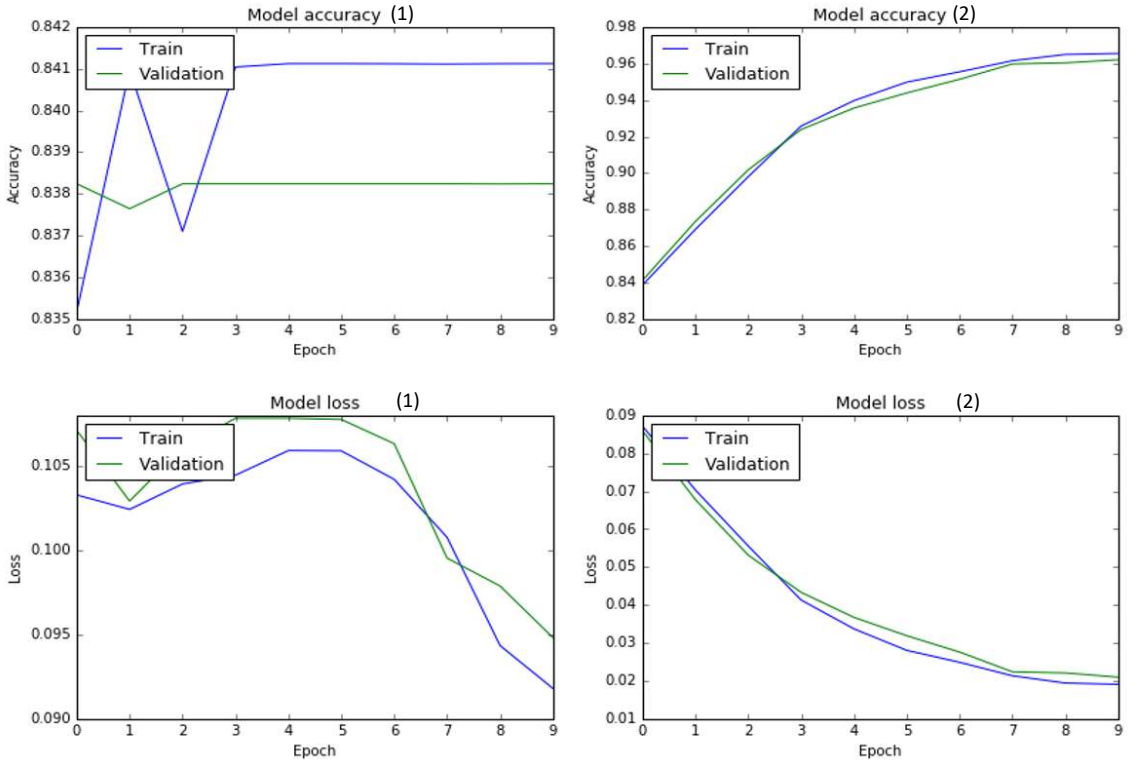

Figure 14: Graphs to show how the model accuracy and loss changed for the training and validation data sets for the **CTV**. The first step of 10 epochs ($lr_1 = 0.001$) are on the left, and the second step of 10 epochs after step 1 ($lr_2 = 0.0001$) are on the right.

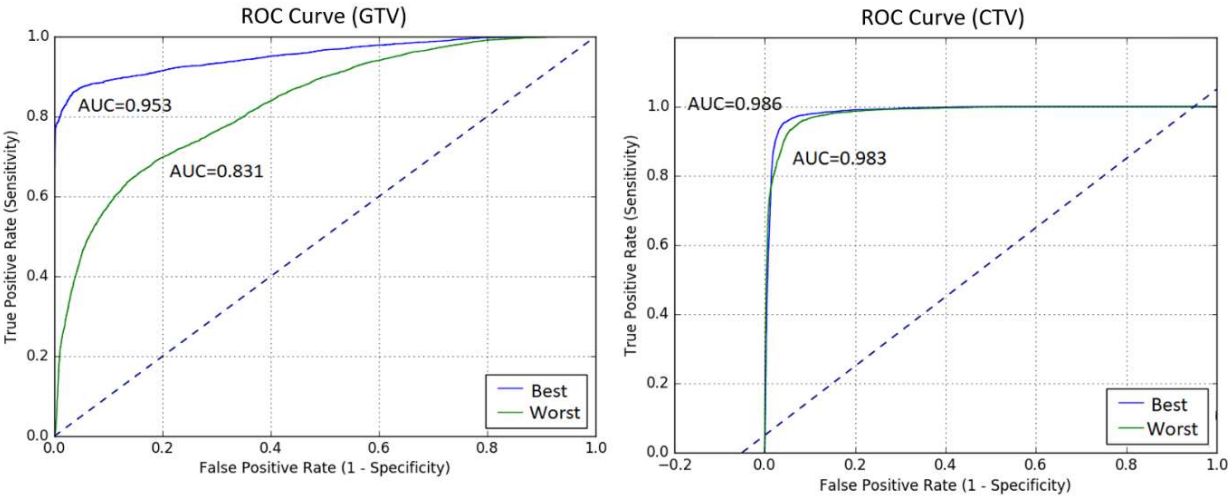

Figure 15: Graphs to show Receiver Operating Characteristic (ROC) curves for the GTV (left) and CTV (right) slices with the best Dice coefficient and the worst coefficients. The Area Under Curve (AUC) has been provided below each curve and the dotted line (blue) is when true positive rate = false positive rate (i.e. proportion of points correctly classified as tumour is equal to proportion of points incorrectly classified as tumour).

5 Discussion

The results were deemed to be mixed with the GTV working model as there was large variation between the several results shown in Figure 12 with target-predicted mask similarities ranging from around 8% to 87%. In Figure 13, the model had high training and validation accuracy from the start of the training onwards. This indicates the gradient converged on the local minima very quickly. The validation accuracy was initially higher than the training accuracy though this was likely due to the poorer training losses in the earlier steps in each epoch causing the average training loss calculated to be reduced overall, compared to the validation accuracy that only used the final, refined model at the end of each epoch. Since, gradient became 0 after one epoch for both the training and validation accuracies, this suggests the initial learning rate was set too high; the step size was very large and so the parameters were changing too rapidly, preventing them from stabilising at a lower value.

When the learning rate was decreased, both of the training and validation accuracies continued to increase and did not plateau for the GTV, suggesting that the model had started to improve. There were no signs of overfitting which would occur if the model extracted features based on the more fine noise within the images, therefore limiting its ability to generalise, causing the validation data accuracy – i.e. for a dataset which was not used directly in the training – to decrease as training continued. The CTV showed improvement throughout the whole training process with validation and training accuracies being quite close; there were no signs of overfitting here either. The accuracies from start to end increase by 2% for the GTV so the training should be continued for more epochs with a reduced learning rate; it should be noted that the accuracy was very high as most of the voxels within the image would be classified as background, and the tumour region could be very small in comparison. The CTV accuracies improved by 13% in comparison, with no signs of overfitting here either.

The worst predictions for the GTV (Figure 11) looked to be much worse than the worst predictions for the CTV (Figure 12), as reflected by the minimum Dice coefficient values. The model clearly struggled with smaller GTV regions. Upon discussion with the clinician responsible for drawing these contours, it was found that despite following the Embrace II protocol [68] (a set of guidelines which advise on how the contours are drawn), the GTV can sometimes be hard to isolate accurately by eye, depending on the nature of the tumour. This can make

the GTV quite difficult to draw a contour around and so the boundary can have a greater margin of error as a result. The CTV-LR has a more discernible boundary by eye and is easier for the clinicians to identify, which reflects its greater success in the training.

Before 2017, no automatic segmentation methods for cervical cancer had been published with the results for the first study for the auto-delineation of cervical cancer tumours released in this year. [69] Here, they used Fishers Linear Discriminant Analysis and achieved Dice coefficients with a reported maximum of 0.44. In comparison, the maximum value obtained using U-Net was 0.87 in our study which is a large increase, suggesting that the U-Net model is very promising. The study also found the use of dynamic contrast enhanced (DCE) images to provide increased accuracy in delineation compared to T2-weighted images; however, even though we did not have access to these, Raystation had diffusion-weighted images (DWI) and Dixon slices available which could be used independently or with the other channels during the training, to see if the changes in contrast have any significant effect. [69] Nevertheless, using these DWI or Dixon images in conjunction with the T2-weighted images could result in the accuracy to decrease if there is substantial misalignment (as they may have been acquired at different time-points).

The model developed in this project also did not put any constraints on the location, shape and size of the tumour when providing the final prediction. The GTV by nature is localised and will always contain many voxels which are adjacent and connected, so constraints which check for any isolated regions (i.e. small areas which are disconnected from the 'main' region of tumour voxels) could be implemented in order to acquire improved segmentations; this was used in the aforementioned auto-delineation of cervical cancer study [69] and led to much higher success in another study for delineating the pancreas. [3] Once these islands have been found, their values can be reduced to 0, leaving only one main tumour region. This would improve predictions with smaller GTV regions and eliminate the detached regions shown in Figure 11 for the worst slices.

6 Conclusions

This project aimed to delineate the tumour regions, in particular the GTV and CTV (for comparison), of given MRI images using the machine learning architecture U-Net. The results have been promising when it comes to delineating more observably prominent tumours, with a reported mean Dice similarity coefficient of (0.5124 ± 0.2890) for the GTV model and (0.8839 ± 0.0421) for the CTV model. Contouring the GTV by eye can be quite a difficult task and so further work needs to be conducted to improve the accuracy of the output results, especially for the GTV; potential ideas which could be carried out include applying different augmentations or filters onto the images, to run the training for a greater number of epochs (with a reduced learning rate), to use more trainable layers (such as in the VGG-19 model [41]) or potentially alternative optimisers (such as `leaky ReLU` which uses a shallow gradient for negative values [26]). Nevertheless, studies which have attempted to delineate cervical cancer tumours have been quite limited, and U-Net shows improvement over previously used techniques such as Fishers Linear Discriminant Analysis. [69]

The CTV trained well using the available data, but as the GTV region is harder to delineate, greater training data would be highly beneficial. However, this can be quite difficult to do in practice since medical imaging data sets are often limited; even if public MRI datasets were used, there would be a lack of clinician-drawn contours, so this would not help in the training process. A workaround would be to artificially increase the size of the data set by augmenting the data available, as described in Section 3.2.2. However, this may not necessarily provide a significant level of improvement and could potentially cause the training accuracy to decrease. These ideas could nonetheless be attempted by the MPhys students in the following semester.

It is likely that as technology improves, there will be greater GPU memory available which would enable the processing of more data simultaneously, with quicker runtimes. Auto-delineation of tumours would especially help those working in research who may require such contours and if the model becomes highly successful with the GTV, the transfer learning process could be repeated with these weights for other regions of interest, in other applications, or be used to further the research into `radiomics` from the previous semester.

References

[1] B. Patel, "Advanced image characterisation using radiomics on multi-parametric magnetic resonance images of cervical cancer patients across multiple time-points," Manchester, 2019.

[2] G. Litjens *et al.*, "A survey on deep learning in medical image analysis," *Med. Image Anal.*, vol. 42, no. 1, pp. 60–88, Dec. 2017.

[3] H. R. Roth *et al.*, "Deep learning and its application to medical image segmentation," 2018.

[4] J. Gu *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognit.*, vol. 77, no. 1, pp. 354–377, May 2018.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *ImageNet Classification with Deep Convolutional Neural Networks*, 2012, p. 9.

[6] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. E. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. 41, 2018.

[7] R. Geirhos, C. R. M. Temme, J. Rauber, H. H. Schütt, M. Bethge, and F. A. Wichmann, "Generalisation in humans and deep neural networks," Ithaca, Aug. 2018.

[8] I. Demir *et al.*, "DeepGlobe 2018: A Challenge to Parse the Earth through Satellite Images," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, vol. 2018-June, pp. 172–17209.

[9] H. R. Roth *et al.*, "Spatial aggregation of holistically-nested convolutional neural networks for automated pancreas localization and segmentation," *Med. Image Anal.*, vol. 45, no. 1, pp. 94–107, Apr. 2018.

[10] Y. Zhou, L. Xie, W. Shen, E. Fishman, and A. Yuille, "Pancreas Segmentation in Abdominal CT Scan: A Coarse-to-Fine Approach," Baltimore, 2016.

[11] P. F. Christ *et al.*, "Automatic liver and lesion segmentation in CT using cascaded fully convolutional neural networks and 3D conditional random fields," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9901 LNCS, pp. 415–423.

[12] B. Ait Skourt, A. El Hassani, and A. Majda, "Lung CT image segmentation using deep neural networks," in *Procedia Computer Science*, 2018, vol. 127, pp. 109–113.

[13] M. Kolařík, R. Burget, V. Uher, K. Říha, and M. Dutta, "Optimized High Resolution 3D Dense-U-Net Network for Brain and Spine Segmentation," *Appl. Sci.*, vol. Applied Sc, no. 9, p. 404, 2019.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," Freiburg, 2015.

[15] M. A. Green A, Cipriano C, Osorio E Vasquez, Weaver J, van Herk M, "Automated sarcopenia assessment and its predictive power in lung cancer radiotherapy patients," in *Automated sarcopenia assessment and its predictive power in lung cancer radiotherapy patients*, 2018, p. 1.

[16] L. L. Gunderson, J. E. Tepper, and J. A. Bogart, *Clinical radiation oncology*. Saunders/Elsevier, 2012.

[17] M. D. Kohli, R. M. Summers, and J. R. Geis, "Medical Image Data and Datasets in the Era of Machine Learning—Whitepaper from the 2016 C-MIMI Meeting Dataset Session," *J. Digit. Imaging*, vol. 30, no. 4, pp. 392–399, Aug. 2017.

[18] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018.

[19] G. Aoki and Y. Sakakibara, "Convolutional neural networks for classification of alignments of non-coding RNA sequences," *Bioinformatics*, vol. 34, no. 13, pp. i237–i244, Jul. 2018.

[20] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," Montreal, Mar. 2016.

[21] H. Le and A. Borji, "What are the Receptive, Effective Receptive, and Projective Fields of Neurons in Convolutional Neural Networks?," 2017.

[22] X. Wu, "Fully convolutional networks for semantic segmentation," *Comput. Sci.*, vol. IEEE Trans, no. 39, pp. 640–551, 2015.

[23] N. Japkowicz, "Supervised versus unsupervised binary-learning by feedforward neural networks," *Mach. Learn.*, vol. 42, no. 1–2, pp. 97–122, 2001.

[24] TowardsDataScience, "CNN_Image_Features," *Medium*. [Online]. Available: https://cdn-images-1.medium.com/max/2400/1*vkQ0hXDaQv57sALXAJquxA.jpeg. [Accessed: 06-May-2019].

[25] A. Bendale and T. Boult, "Towards Open Set Deep Networks," Colorado Springs, Nov. 2015.

[26] D. Pedamonti, "Comparison of non-linear activation functions for deep neural networks on MNIST classification task," Edinburgh, 2018.

[27] C.-Y. Lee, P. W. Gallagher, and Z. Tu, "Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree," San Diego, 2016.

[28] T. Falk *et al.*, "U-Net: deep learning for cell counting, detection, and morphometry," *Nat. Methods*, vol. 16, no. 1, pp. 67–90, 2019.

[29] O. Oktay *et al.*, "Attention U-Net: Learning Where to Look for the Pancreas," London, Apr. 2018.

[30] X. Huang, "Convolutional Neural Networks In Convolution," Shenzhen, 2018.

[31] H. He *et al.*, "Road Extraction by Using Atrous Spatial Pyramid Pooling Integrated Encoder-Decoder Network

and Structural Similarity Loss," *Remote Sens.*, vol. 11, no. 9, p. 1015, Apr. 2019.
[32] P. V, "Conv2D_Image," *GitHub*, 2016. [Online]. Available: https://github.com/PetarV-/TikZ/tree/master/2D Convolution. [Accessed: 12-May-2019].
[33] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A Survey of the Recent Architectures of Deep Convolutional Neural Networks," Islamabad, Jan. 2019.
[34] Thom Lane, "Transposed Convolutions explained with… MS Excel! – Apache MXNet – Medium," *Medium*, 2018. [Online]. Available: https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8. [Accessed: 06-May-2019].
[35] C. C. Özkan and F. S. Erbek, "The Comparison of Activation Functions for Multispectral Landsat TM Image Classification," *Photogramm. Eng. Remote Sens.*, vol. 69, no. 11, pp. 1225–1234, 2003.
[36] P. D'Souza, S.-C. Liu, and R. H. R. Hahnloser, "Perceptron learning rule derived from spike-frequency adaptation and spike-time-dependent plasticity," *Proc. Natl. Acad. Sci.*, vol. 107, no. 10, pp. 4722–4727, 2010.
[37] V. E. Balas, S. S. Roy, D. Sharma, and P. Samui, *Handbook of Deep Learning Applications*, vol. 136. London: Springer, 2019.
[38] V. Suárez-Paniagua and I. Segura-Bedmar, "Evaluation of pooling operations in convolutional architectures for drug-drug interaction extraction," *BMC Bioinformatics*, vol. 19, no. Suppl 8, p. 209, 2018.
[39] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8818, pp. 364–375.
[40] A. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng, "On Random Weights and Unsupervised Feature Learning," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 1089–1096.
[41] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2014, p. 14.
[42] M. Svanera *et al.*, "Transfer learning of deep neural network representations for fMRI decoding," Brescia, 2019.
[43] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning Filters for Efficient ConvNets," in *Pruning Filters for Efficient ConvNets (ICLR 2017)*, 2016.
[44] K. Janocha and W. M. Czarnecki, "On Loss Functions for Deep Neural Networks in Classification," Krakow, Feb. 2017.
[45] L. Fidon *et al.*, "Generalised Wasserstein Dice Score for Imbalanced Multi-class Segmentation using Holistic Convolutional Networks," London.
[46] J. Jordan, "An overview of semantic image segmentation.," *jeremyjordan.me*, 2018. [Online]. Available: https://www.jeremyjordan.me/semantic-segmentation/. [Accessed: 07-May-2019].
[47] R. R. Shamir, Y. Duchin, J. Kim, G. Sapiro, and N. Harel, "Continuous Dice Coefficient: a Method for Evaluating Probabilistic Segmentations," London, 2018.
[48] R. Lguensat, M. Sun, R. Fablet, E. Mason, P. Tandeo, and G. Chen, "EddyNet: A Deep Neural Network For Pixel-Wise Classification of Oceanic Eddies," New York City, Nov. 2017.
[49] D. Ruppert, "The Elements of Statistical Learning: Data Mining, Inference, and Prediction," *J. Am. Stat. Assoc.*, vol. 99, no. 466, pp. 567–567, 2004.
[50] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," in *19th International Conference on Computational Statistics, Keynote, Invited and Contributed Papers*, 2010, pp. 177–186.
[51] Y. Bulatov, "Derivation of backpropagation equations," *Swart. Coll.*, pp. 1–5, 2012.
[52] H. Hayashi, J. Koushik, and G. Neubig, "Eve: A Gradient Based Optimization Method with Locally and Globally Adaptive Learning Rates," Ithaca, Nov. 2016.
[53] S. Ruder, "An overview of gradient descent optimization algorithms," Dublin, Sep. 2016.
[54] Q. V Le, A. Coates, B. Prochnow, and A. Y. Ng, "On Optimization Methods for Deep Learning," in *Proc. International Conference on Machine Learning*, 2011, pp. 265–272.
[55] C. C. Aggarwal, *Neural networks and deep learning : a textbook*, 1st ed. New York City: Springer, 2018.
[56] K. Parthy, "CS231n Convolutional Neural Networks for Visual Recognition," 2018.
[57] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference for Learning Representations*, 2014.
[58] X. Chen, S. Liu, R. Sun, and M. Hong, "On the convergence of a class of Adam-type algorithms for non-convex optimization," 2018, p. 30.
[59] A. Géron, *Hands-On Machine Learning with Scikit-Learn: Concepts, Tool, and Techniques to Build Intelligent Systems*. 2017.
[60] D. Veit, "Neural networks and their application to textile technology," in *Simulation in Textile Technology: Theory and Applications*, 1st ed., Cambridge: Woodhead Publishing, 2012, pp. 9–71.
[61] E. Joergensen, "Deep Learning on HPC Systems," 2017.
[62] R. Pötter, E. Fidarova, and P. Petrič, "Target volume definition (GTV, HR-CTV and IR-CTV): GYN GEC-ESTRO Recommendations," 2013.
[63] F. Chollet, "Keras Module," *Keras*, 2015. [Online]. Available: https://keras.io. [Accessed: 12-May-2019].
[64] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," Chicago, 2016.
[65] M. Van Herk, "WorldMatch software." Netherlands Cancer Institute, Amsterdam, p. 1, 2016.
[66] B. C. Lowekamp, D. T. Chen, L. Ibáñez, and D. Blezek, "The Design of SimpleITK," *Front. Neuroinform.*, vol. 7, no. 1, p. 10, 2013.
[67] P. Goyal *et al.*, "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour," 2017.
[68] K. Tanderup *et al.*, "EMBRACE-II: Image guided intensity modulated External beam radiochemotherapy and MRI based adaptive BRAchytherapy in locally advanced CErvical cancer," 2015.
[69] T. Torheim *et al.*, "Autodelineation of cervical cancers using multiparametric magnetic resonance imaging and machine learning," *Acta Oncol. (Madr).*, vol. 56, no. 6, pp. 806–812, May 2017.

Appendix

	Optimiser	Activation function	Image augment.	Layers unfrozen	Learning rate (step)	Epochs (step)	Prediction observable
GTV	SGD	ReLU	False	True	10^{-3} (1) 10^{-4} (2)	10 (1) 20 (2)	No
	Adam	tanh	False	True	10^{-3} (1) 10^{-4} (2)	10 (1) 30 (2)	Yes
	Adam	tanh	False	True	10^{-3} (1) 10^{-4} (2)	10 (1) 20 (2)	Yes
	Adam	tanh	False	False	10^{-3} (1) 10^{-4} (2)	10 (1) 20 (2)	No
	Adam	tanh	True	True	10^{-3} (1) 10^{-4} (2)	10 (1) 20 (2)	No
	Adam	tanh	False	True	10^{-3} (1) 10^{-4} (2)	10 (1) 20 (2)	No
	Adam	tanh	False	True	10^{-3} (1) 10^{-4} (2)	4 (1) 8 (2)	No
	Adam	tanh	False	True	10^{-3} (1) 10^{-4} (2) 10^{-5} (3)	10 (1) 20 (2) 30 (3)	Yes
CTV	Adam	tanh	False	True	10^{-3} (1) 10^{-4} (2) 10^{-5} (3)	10 (1) 20 (2) 30 (3)	Yes
	Adam	tanh	False	True	10^{-3} (1) 10^{-4} (2)	10 (1) 20 (2) 50 (3)	Yes
	Adam	tanh	False	True	10^{-3} (1) 10^{-4} (2)	10 (1) 20 (2)	Yes
	Adam	tanh	False	True	10^{-3} (1) 10^{-4} (2)	10 (1) 20 (2)	Yes

Table 1: A table of some of the parameter combinations which were test for the GTV and CTV models.

Risk Assessment

This project was completed in the experiment will be carried out in both the Christie NHS Foundation Trust hospital and the School of Physics and Astronomy Schuster building. All work was carried out via the use of personal laptops in this project.

Hazard Identification:

- Use of portable electrical equipment.
- Prolonged use of display screen equipment, computer keyboard and mouse.

Risks involved:

- Increased risk of repetitive strain injury (RSI) and/or eye strain from prolonged laptop use.
- Laptops and portable electrical equipment such as chargers and cables cause a trip hazard.

Actions to be taken:

- Ensure that any electrical equipment used at the Christie hospital or in the Schuster building have PASS labels (which have not been expired) before use.
- Take regular breaks to relieve eye strain and sit upright in stable, back-supporting chairs where possible.