Deep Learning for Quality Assurance of Image Registration in Radiotherapy

Nasiq Ziyan

10119757

School of Physics and Astronomy

University of Manchester

Semester 1 MPhys Project Report

Jan 2022

This experiment was performed in collaboration with *Ahmed Maniyar*

Abstract

As patients experience significant anatomical changes during adaptive radiotherapy (ART), contours are manually drawn around organs at risk by a clinical team before each ART session, a process that is time-consuming and taxing on clinical resources. However, contour propagation using displacement vector fields produced by deformable image registration algorithms can automate this process. Though, quality assurance of these propagated contours is necessary for clinical implementation. A convolutional neural network was constructed, trained and validated on 2D PET-CT scans to predict a similarity metric that quantifies the similarity between automated contours and manually-drawn contours. A steadily decreasing training loss over 20 epochs paired with an increasing validation loss during the final epochs was observed, which indicates overfitting to the training set. Methods of reducing model complexity and other regularisation techniques, such as data augmentation, are suggested to mitigate this issue.

Contents

1 Introduction

Radiotherapy (RT), generally regarded as the most effective method of cancer treatment after surgery, involves the use of ionising radiation (typically x-rays) to treat cancer [1]. A course of treatment prescribed by a radiation oncologist is delivered by a clinical team with the aim of shrinking the tumour by permanently damaging the DNA of cancer cells. While DNA damage is suffered by the cells in nearby organs, the side effects are generally temporary due to the active DNA repair mechanisms present within the cell [2]. However, alterations and mutations in the responsible DNA proteins, caused by RT, may lead to erroneous repair, inducing second malignancies [3]. Therefore, accurate and precise treatment delivery is designed to maximise the dose of radiation to the malignant tumour while minimising the dose to surrounding healthy tissue, especially to organs at risk (OARs) [1, 4]. Additionally, as the total dose is fractionated into multiple dose deliveries (depending on the rate of tumour growth [5]), treatment plans typically last up to 7 weeks [1].

For accurate delineation of tumour volumes against the surrounding healthy tissue, multiple imaging modalities, such as CT, MRI and PET, are currently employed [6]. A CT scan is obtained to construct a three-dimensional anatomic model of the patient. Then, additional contrast for further delineation can be provided using MRI techniques while dynamic metabolic information can be acquired using PET. These scans from each modality can be mapped into one coordinate system by a process called multi-modal image registration [6, 7] and OARs can be manually segmented as regions for minimum radiation exposure. However, patients may experience anatomical changes during this treatment period, such as weight loss, organ shape variation and ideally, tumour shrinkage. Therefore, an improved RT procedure capable of modifying a treatment plan accordingly was proposed, known as adaptive radiotherapy (ART) [8].

This manual contouring procedure, the most fundamental task for effective RT, relies on accurate interpretation of the multi-modal image registration [9]. As a result, it is often the most time-consuming procedure during treatment planning and hence, is especially taxing on clinical resources [10]. Moreover, due to the variability among radiation oncologists and subjectivity in interpretation, there is potential susceptibility to inaccuracies during volume delineation [9]. Computational advancements in artificial intelligence, notably deep learning, provide a promising opportunity for automating the primary ART components, such as contouring, image registration, dose calculation and quality assurance (QA) [10, 11]. In particular, deformable image registration (DIR) has been used to automate the deformations observed between clinically drawn contours from scans obtained in different sessions [12]. These deformations correspond to anatomical changes during the course of treatment and the corresponding mapping between two scans is represented by a displacement vector field (DVF). In this DIR method, a radiation oncologist manually draws a contour around OARs during the first session [13]. Then, using a CT scan from the next session as an input, this contour is propagated using DIR such that the contour is deformed on the new CT scan to minimise exposure to OARs in the next session.

Before manually drawn contours can be substituted by the DIR technique for automated contouring in a clinical scenario, quality assurance must be performed to ensure that the contours produced via DIR are sufficiently similar to manually drawn contours. This can be quantified us-

ing similarity metrics. In this project, a deep learning methodology, known as a 2D convolutional neural network (CNN), was constructed to predict the values of these similarity metrics, given the DVF (which describes the deformations produced by DIR). Then, this model was trained on 2D CT scans of brainstem data and evaluated using the known similarity metric between a clinically drawn contour and the DIR-produced contour. This project continues the work carried out in [14, 15, 16].

2 Project Background

2.1 Image Registration

Image registration is the process that relates identical (anatomical) points in two images by a geometric transformation [6]. Generally, the first image (floating image) is deformed to align with the second image (reference image) by this transformation. The corresponding mapping is represented by a displacement vector field (DVF). The registration is considered multi-modal if the images were obtained by different imaging modalities, such as PET and CT. On the other hand, if the images were obtained by the same modality but at different points in time, the registration is multi-temporal. The latter will be considered in this project.

While many geometric transformations are possible to transform the pixels (for 2D images) or voxels (for 3D images) in the floating image to the reference image, three commonly used transformations are outlined here [6]. Firstly, rigid transformations, which preserve the distance between any pair of points, allow for translations, rotations or a combination of both, in all directions. Secondly, affine transformations allow for uniform scaling and shearing, in addition to rigid transformations. However, the distance between any pair of points is no longer conserved, but parallel lines will remain parallel. The final transformation is known as a non-rigid (deformable) transformation which corresponds to a deformable image registration (DIR). In this transformation, each pixel or voxel is assigned a unique displacement vector in the floating image which attempts to map each point to the identical anatomical point in the reference image. Deformable transformations are fitting for this project's application as anatomical changes generally correspond to local deformations. The transformation of the floating image $F(x')$ to the reference image $R(x)$ can be expressed mathematically as [7]

$$F(x' + D(x')) = F(I(x')) = R(x), \tag{1}$$

where x' denotes the position vector corresponding to the pixel or voxel and $D(x')$ denotes the DVF. If each point in the floating image is exactly mapped to the identical point in the reference image, the DIR is considered ideal and is represented by $I(x')$ [7]. However, as the number of pixels is finite and image registration algorithms are imperfect, this ideal transformation is not achieved in practice. An example of an ideal DIR is shown in Figure 1 [7]. This shows the DVF (C) that is required to deform the corners of a square in the floating image (A) to best match the shape in the reference image (B).

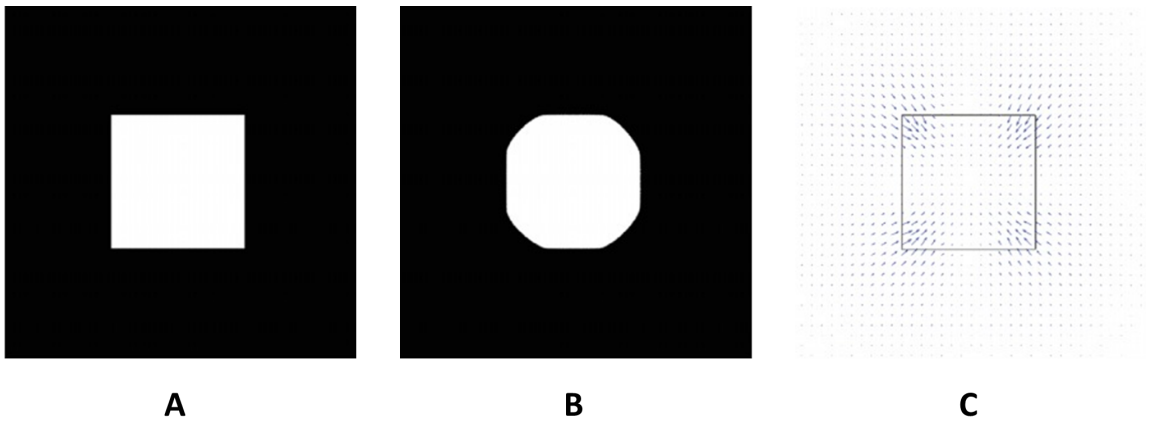

Figure 1: A simple example of deformable image registration (DIR) being applied to a floating image (A) and a reference image (B). DIR deforms the corners of the square in the floating image to align with the reference image. The corresponding displacement vector field (DVF) that achieves this ideal transformation is shown in (C), where the blue arrows represent the magnitude and direction of the DVF at each point in the image [7].

2.2 Deformable Image Registration in ART

Automatic segmentation is a promising application of DIR in ART, which was originally applied to CT images of lung and abdomen cancer patients [17]. In this ART application, an initial CT scan of the patient is obtained before treatment which acts as the floating image [9]. Then, a radiation oncologist draws contours around the regions where radiation exposure is to be minimised, such as the brainstem or parotid gland. A second CT scan is obtained during the next treatment session which acts as the reference image. The discrepancies between the floating and reference image represent the anatomical changes that have occurred during this time interval. In ordinary RT, the radiation oncologist would have to manually draw a new contour on the second CT scan and this process would need to be repeated over the course of treatment. However, in ART, this process can be automated by first applying DIR on the floating and reference image, both without contours drawn on them. This produces a DVF that corresponds to the anatomical changes. Then, a contour is manually drawn around the organs at risk (OARs) on the floating image which is subsequently deformed by applying the DVF transformation. Ideally, the propagated contour will perfectly delineate the OARs in the reference image (pre-treatment CT scan), accounting for the anatomical changes and ready for dose delivery. Figure 2 illustrates this DIR process for automatic segmentation of the parotid gland [18]. The manually drawn contour around the right parotid gland (blue) was mapped to the deformed contour (green) by applying a DVF transformation. This DIR-produced contour is also known as the registered contour.
To ensure that the DIR-produced contour is sufficient for clinical implementation, it must be quantitatively assessed against a manually drawn contour on the reference image. This contour is known as the ground truth and the similarity between this contour and the DIR-produced contour can be quantified using similarity metrics.

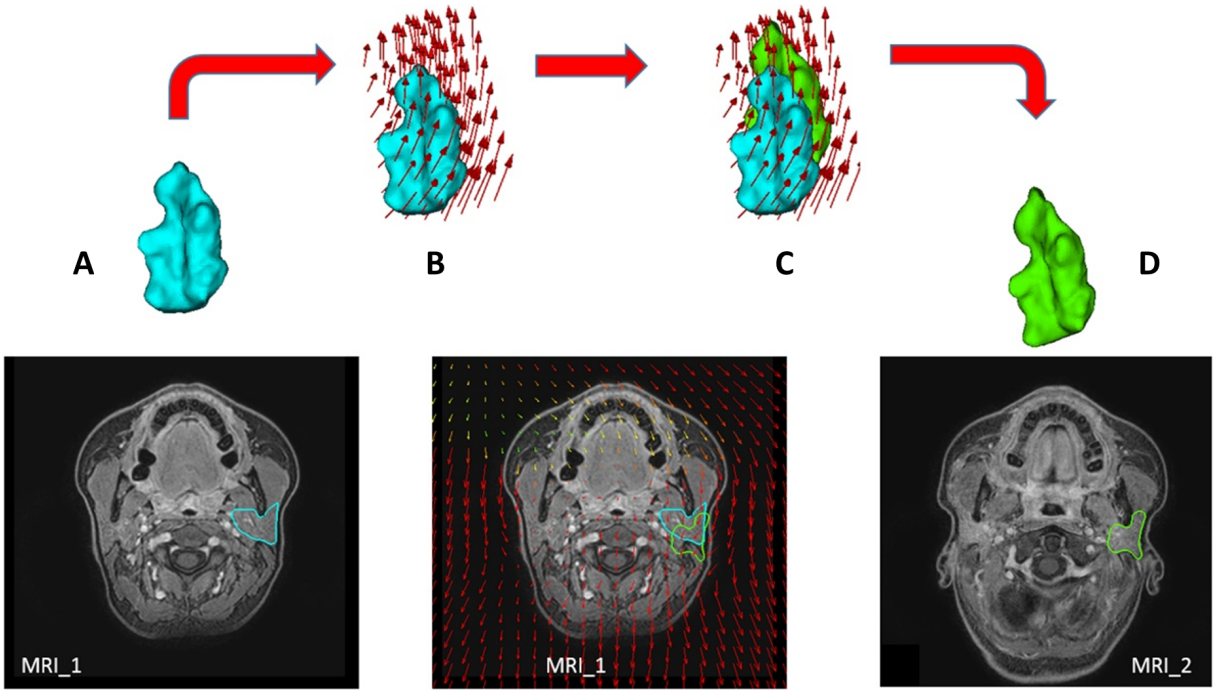

Figure 2: An illustration of automatic contour propagation using DIR on an MRI scan. A 3D contour drawn around a parotid gland (A) was manually segmented on the floating image (MRI_1). Then, a DVF produced by DIR was applied to it (B), resulting in a deformed contour (C). This DIR-produced contour (D) is illustrated on the reference image (MRI_2) [18].

2.3 Similarity Metrics

Similarity metrics have been used to evaluate the accuracy of OAR segmentation in the medical field [19]. These are commonly classified into six categories [20] but only two of these are relevant to this project: overlap based metrics and spatial-distance based metrics. Firstly, the most widely used overlap based metric for measuring the similarity between two surfaces or volumes is the DICE coefficient [18, 20]. It is expressed mathematically as

$$\mathrm{DICE}\,(A, B) = \frac{2\,(A \cap B)}{|A| + |B|} \tag{2}$$

where A and B are the two surfaces or volumes in 2D or 3D, respectively. The DICE coefficient ranges from 0 to 1 where 0 indicates no overlap and 1 indicates a perfect overlap. In a 2D ART application, $|A|$ and $|B|$ may represent the number of pixels in the manually drawn contour and the registered contour, respectively. Then, $A \cap B$ represents those pixels that are within both contours. While the DICE metric is symmetric under the exchange of A and B, it only considers the areas or volumes and not the contour perimeters, which may be problematic in ART. For example, if a small region of an OAR is enclosed by the ground truth contour but is not enclosed by the registered contour, the DICE coefficient will be minimally affected while the exposure to the OAR may be significant. This issue can be mitigated by using a spatial-distance based metric such as the Hausdorff distance (HD) metric.

Firstly, the directed Hausdorff distance from contour A to contour B can be defined as the largest of all distances from a point on the contour of A to the nearest point on contour B [21]. This is shown in Figure 3 and expressed mathematically as [20]

$$h_{\max}(A, B) = \max_{a \in A} \min_{b \in B} || \, a - b \, ||, \tag{3}$$

where a and b are points on the contours A and B, respectively, and $|| \, a - b \, ||$ is the distance between them. Since, $h_{\max}(A, B) \neq h_{\max}(B, A)$ in most cases, the maximum of the two is taken to give a symmetric measure, defined as

$$\mathrm{HD}(A, B) = \max(h_{\max}(A, B), \, h_{\max}(B, A)). \tag{4}$$

HD is often sensitive to outliers as it considers the extrema of the two contours. Since noise appearing in medical images may be perceived as outliers, the HD can be further modified to mitigate this issue. This is achieved by replacing the largest of all distances from a point on the contour of A in Equation 3 by the mean of all these distances which gives

$$h_{\mathrm{mean}}(A, B) = \frac{1}{N} \sum_{a \in A} \min_{b \in B} || \, a - b \, ||. \tag{5}$$

This is known as the mean directed HD which can be made symmetric in an identical manner to Equation 4 to give the mean surface distance (MSD), defined as

$$\mathrm{MSD}(A, B) = \max(h_{\mathrm{mean}}(A, B), \, h_{\mathrm{mean}}(B, A)). \tag{6}$$

Since the CT scans used in this project are 3D, the model that is trained and evaluated in this project predicts 3D MSD values [14].

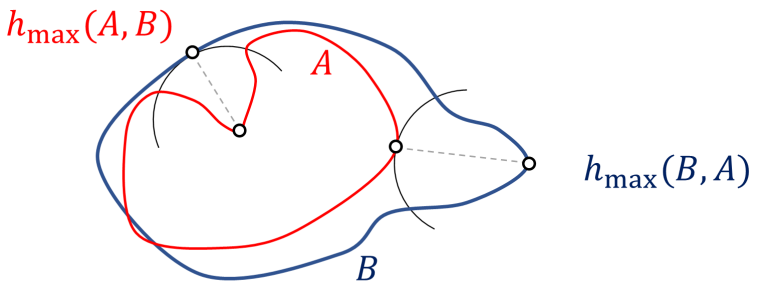

Figure 3: An illustration of the directed Hausdorff distances $h_{\max}(A, B)$ and $h_{\max}(B, A)$, denoted by dashed lines, between two contours A and B. These distances are found using Equation 3. Note that $h_{\max}(A, B) \neq h_{\max}(B, A)$.

3 Neural Networks in Deep Learning

For a model to predict the similarity metrics discussed in the previous section, it must be provided with inputs and labels. This section introduces the concept of neural networks and how it learns correlations between any given inputs and labels in order to make predictions. Then, Section 4 explains the inputs and labels provided to our neural network.

3.1 Dense Networks

Firstly, deep learning (DL) is a subdivision of machine learning that applies algorithms inspired by the function and neural network structure of the brain [22, 23, 24]. As a result of this similarity, they are often called artificial neural networks or models. These neural networks (NN) are organised into multiple layers of processing, starting with one input layer, ending with one output layer and containing hidden layers in between. Each layer contains a number of neurons, or nodes, that are tasked with learning features which are represented and output as values to the nodes in the layer ahead of it. DL networks have a hierarchical structure which means that complex features learnt in a deeper layer are built from simpler features learnt from a shallower layer [22]. NNs can be categorised as either classification or regression depending on the network's objective. In general, an NN attempts to find a function that maps input variables to output variables [22, 23, 24]. In a classification NN, these output variables are frequently called classes or labels and are represented by the nodes in the output layer. For example, a binary classification NN receives a number of inputs and predicts one of two discrete classes, represented by two nodes in the output layer. On the other hand, a regression NN predicts one or more continuous variables and therefore, must contain one node for each continuous variable in the output layer. Furthermore, if each node in a DL network can output to every node in the layer behind and ahead of it, the NN is referred to as fully connected, or dense. An example of a dense network for binary classification is shown in Figure 4.

The number of nodes in the input layer is equal to the number of input variables, or features, in the data that is being processed [24]. In a dense NN, the input is passed to all nodes in the next layer as an output represented by a connection. This has an associated weight value that represents the strength of the connection. The weight multiplies the output received from the corresponding node which is then summed over all nodes. Each node also has a bias value which is added to the weighted summation over all node outputs in the previous layer. This provides greater flexibility to the NN, which is explained at the end of this section. The weights and biases are continuously updated as the model learns. Hence, they are defined as trainable parameters. Before the weighted sum and bias can be interpreted by the model as a higher-level feature, it is first passed to an activation function. The activation function of a node is inspired by the firing, or activating, of neurons in the brain during brain activity [25]. In an NN, it transforms the node input into a node output which is subsequently passed to all nodes in the next layer. This eventually reaches the output layer and a prediction is produced. If the weight value for a given node is large, the feature learnt by the corresponding node has a large effect on the final prediction [26].

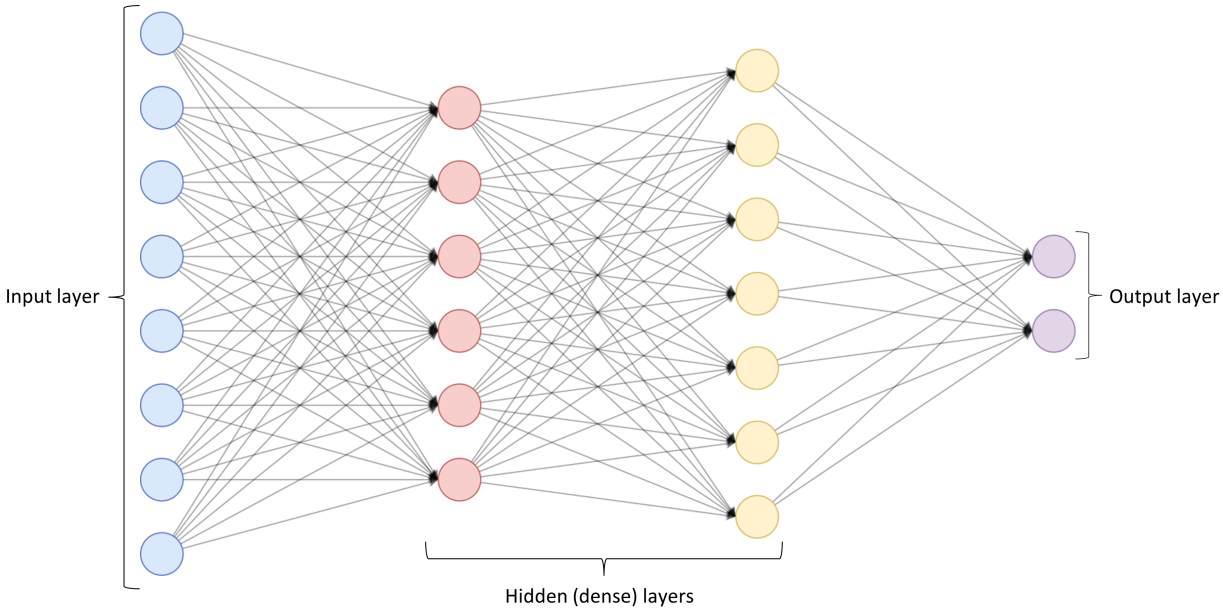

Figure 4: An illustration of a dense neural network that is capable of performing binary classification. It contains an input layer, 2 hidden (dense) layers and an output layer where each layer consists of nodes denoted by circles. All nodes are capable of outputting values to every node in the layer ahead of it, denoted by the direction of the arrows.

To mathematically represent the operations occurring in an NN, the following notation [26] is stated here and illustrated in Figure 5. Firstly, the layers of the NN are indexed as $l = 1, 2, .., L$. The nodes in a given layer l are indexed from $j = 0, 1, .., n-1$ while the nodes in the previous layer $l-1$ are indexed from $k = 0, 1, .., n-1$. Then, the input z for the node j in layer l is denoted as $z_j^{(l)}$. The weight that connects the node k in layer $l-1$ to the node j in the next layer l is denoted as $w_{jk}^{(l)}$. The activation function used for the nodes in layer l is denoted as $g^{(l)}$ while the activation output for a given node j is denoted as $a_j^{(l)}$.

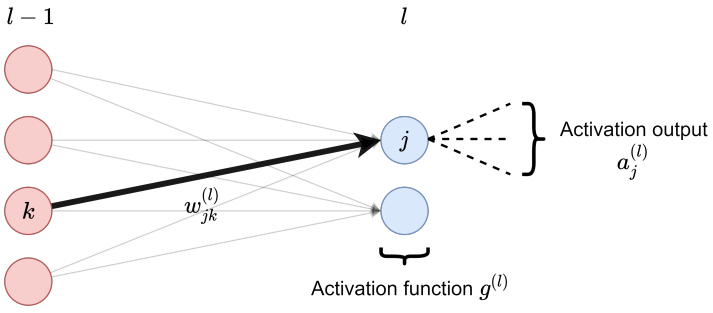

Figure 5: A cropped diagram of a NN illustrating the notation used in this report. A node k in layer $l-1$ connected to a node j in layer l with corresponding weight $w_{jk}^{(l)}$ is indicated by a bold arrow. Activation outputs are indicated by dashed lines.

Thus, the weighted sum input for node j in layer l, including the bias $b_j^{(l)}$, can be expressed

mathematically as

$$z_j^{(l)} = b_j^{(l)} + \sum_{k=0}^{n-1} w_{jk}^{(l)}\ a_k^{(l-1)}. \tag{7}$$

Recalling that the node input is passed into an activation function, the activation output is expressed as

$$a_j^{(l)} = g^{(l)}\left(z_j^{(l)}\right), \tag{8}$$

where the activation output indicates the firing strength of the neuron. There are multiple activation functions typically used in an NN depending on the complexity of the features present in the input data [26]. During one full forward pass, known as an epoch, all the input values from the input data are passed through a series of activation functions before reaching the output layer, where a prediction is produced. Therefore, the final prediction is a composite function of activation functions. If the NN consists of only linear activation functions, the mapping function relating the input to the output will be linear because the composition of multiple linear functions produces another linear function. Therefore, NNs tasked with learning complex features will primarily employ non-linear activation functions. The nonlinear activation function used in this project is the rectified linear unit (ReLU), displayed in Figure 6 and expressed as [22]

$$g^{(l)}\left(z_j^{(l)}\right) = \max\left(0,\ z_j^{(l)}\right). \tag{9}$$

From the perspective of an NN, the threshold value of 0 for neuron activation is somewhat arbitrary. However, since the bias is a trainable parameter, the threshold can be shifted left or right during the optimisation of the model. While the bias is associated with the activation function, the weights are associated and influenced by the previous layer. As a result, the bias is capable of compensating for the inflexibility of the activation function.

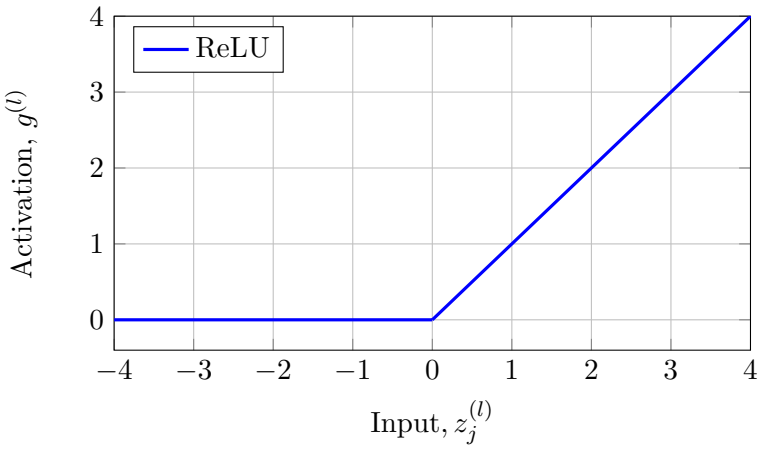

Figure 6: A graph displaying the nonlinear rectified linear unit (ReLU) activation function given by Equation 9.

3.2 Training and Loss

Once an NN model has been constructed, input data can be propagated through the network to the output layer where a prediction is made [24, 26]. These predictions are compared to the desired labels supplied to the model which produces a loss value. A loss value of zero implies that the label has been perfectly predicted. The method of comparison is dependent on the loss function, but the most common method is by taking the mean squared error (MSE) for all activation outputs in the output layer. For a single sample passed to the model, the desired value for a given node j in the output layer L is defined as y_j. Thus, the MSE total loss function C_0, where 0 denotes the first sample, is expressed as [26]

$$C_0 = \sum_{j=0}^{n-1} C_{0_j} = \sum_{j=0}^{n-1} \left(a_j^{(L)} - y_j \right)^2, \tag{10}$$

During each epoch, the trainable parameters (weights and biases) are updated multiple times by an optimisation algorithm with the aim of increasing the accuracy of predictions by minimising the loss function [26]. This process is known as training. A commonly used optimiser is known as stochastic gradient descent (SGD) that minimises the loss by calculating the gradient of the loss function with respect to the trainable parameters. This process is known as backpropagation.

In the case of binary classification, generally, backpropagation will aim to increase the activation output (neuron firing strength) of the desired node in the output layer and decrease the activation output of the undesired node(s), such that the loss is minimised. Since these activation outputs are dependent on the trainable parameters found earlier on in the network, the gradient of the loss function with respect to each trainable parameter must be determined to efficiently minimise the total loss. Backpropagation will be described mathematically by first determining a general equation of the gradient of the total loss function with respect to a given weight connected to the output layer. This will be applied to a specific weight that connects to a node in the output layer and then to a specific weight in the preceding layer. This will illustrate a recursive process, characteristic of backpropagation, that can be used to determine the gradient of the total loss with respect to any weight in a model.

As seen from Equation 10, the loss for a single node j in the output layer L, C_{0_j}, is a function of the activation output $a_j^{(L)}$. This is a function of the activation function $g^{(L)}$, which in turn is a function of the input $z_j^{(L)}$, seen by Equation 8. Finally, this is a function of all weights that connect to this node j in the output layer, seen by Equation 7. By defining a vector that contains all weights that connect to a node j in a layer l as $\mathbf{w}_j^{(l)}$ and summing over all nodes in the output layer, the total loss function can be expressed as [26]

$$C_0 = \sum_{j=0}^{n-1} C_{0_j} = \sum_{j=0}^{n-1} C_{0_j} \left(a_j^{(L)} \left(z_j^{(L)} \left(\mathbf{w}_j^{(L)} \right) \right) \right). \tag{11}$$

By applying the chain rule to Equation 11, the derivative of the loss function with respect to

$w_{jk}^{(L)}$ for the first sample is the general equation:

$$\frac{\partial C_0}{\partial w_{jk}^{(L)}} = \left(\frac{\partial C_0}{\partial a_j^{(L)}}\right)\left(\frac{\partial a_j^{(L)}}{\partial z_j^{(L)}}\right)\left(\frac{\partial z_j^{(L)}}{\partial w_{jk}^{(L)}}\right). \tag{12}$$

By taking $j = 1$ and $k = 2$ as an example, as shown in Figure 7, the first, second and third terms can be evaluated using Equation 10, 8 and 7, respectively. This results in

$$\frac{\partial C_0}{\partial w_{12}^{(L)}} = 2\left(a_1^{(L)} - y_1\right)\left(g'^{(L)}\left(z_1^{(L)}\right)\right)\left(a_2^{(L-1)}\right). \tag{13}$$

For all N training samples, the mean derivative of the loss function is determined, expressed as

$$\frac{\partial C}{\partial w_{12}^{(L)}} = \frac{1}{N}\sum_{i=0}^{N-1}\frac{\partial C_i}{\partial w_{12}^{(L)}}. \tag{14}$$

By taking $j = 2$ and $k = 2$ in the layer $L - 1$ instead, Equation 12 becomes

$$\frac{\partial C_0}{\partial w_{22}^{(L-1)}} = \left(\frac{\partial C_0}{\partial a_2^{(L-1)}}\right)\left(\frac{\partial a_2^{(L-1)}}{\partial z_2^{(L-1)}}\right)\left(\frac{\partial z_2^{(L-1)}}{\partial w_{22}^{(L-1)}}\right). \tag{15}$$

While the second and third terms of this equation can be evaluated in the same way as above, the activation output $a_2^{(L-1)}$ in the first term is no longer an explicit function of the loss function C_0, as shown in Figure 7.

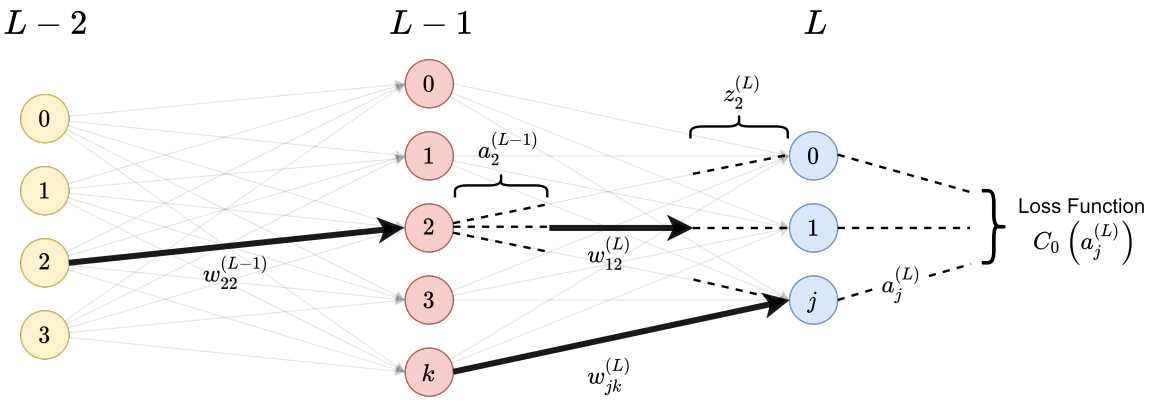

Figure 7: An illustration of the final three layers of a dense neural network. Activation outputs leaving a node and the associated node inputs are indicated by dashed lines while weights assigned to particular nodes are indicated by bold arrows. Nodes are represented by circles and the values j and k are defined such that the node j in a given layer l proceeds node k in the preceding layer $l - 1$.

In fact, working backwards from the loss function, the activation output $a_2^{(L-1)}$ depends on $a_j^{(L)}$, for all nodes j, which depends on $z_j^{(L)}$, for node $j = 2$, which is explicitly dependent on $a_2^{(L-1)}$.

This can be expressed mathematically as

$$\frac{\partial C_0}{\partial a_2^{(L-1)}} = \sum_{j=0}^{n-1} \left[\left(\frac{\partial C_0}{\partial a_j^{(L)}} \right) \left(\frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \right) \left(\frac{\partial z_j^{(L)}}{\partial a_2^{(L-1)}} \right) \right] \tag{16}$$

$$= \sum_{j=0}^{n-1} \left[2 \left(a_j^{(L)} - y_j \right) \left(g'^{(L)} \left(z_j^{(L)} \right) \right) \left(w_{j2}^{(L)} \right) \right], \tag{17}$$

where the first, second and third terms have been evaluated using Equation 10, 8 and 7, respectively. This result can be used to determine the derivative of the loss function with respect to any weight that connects to node $j = 2$ in layer $L-1$. For our example weight $w_{22}^{(L-1)}$, Equation 17 can be substituted into Equation 13 and averaged over all N samples as before to produce the desired result. Starting from the general equation (Equation 11), this procedure can be used to calculate derivatives with respect to any weight that resides earlier in the NN, successively producing more derivatives. This is the backpropagation procedure followed by the SGD optimiser to determine the gradient of the loss function. Once this is determined, the old weight $w_{jk}^{(l)}$ is updated using

$$w_{jk}^{(l)\prime} = w_{jk}^{(l)} - \eta \frac{\partial C}{\partial w_{jk}^{(l)}} \tag{18}$$

where η is the learning rate, a parameter that scales the amount a trainable parameter is changed after each epoch. The minus sign implies gradient descent such that the loss is minimised [24].

3.3 Convolutional Neural Networks

The NNs discussed so far have been fully connected. However, this is not a requirement. In fact, convolutional neural networks (CNNs) are a class of partially connected, artificial neural networks that have dominated in computer vision tasks, especially in the medical domain [24, 27]. While there are many similarities between a CNN and a dense NN, the main difference is the addition of convolutional layers. These utilise filters, analogous to nodes in dense layers, for pattern detection of features in the input data. Typically, these inputs are tensor objects which represent the pixels or voxels in an image. Similar to dense NNs, these convolutional layers exhibit a hierarchical structure of feature complexity where filters in earlier layers detect simple geometric objects, such as lines and corners, while filters later in the CNN are capable of constructing complex objects from these simple objects.

A convolutional layer also receives input, performs a transformation and adds a bias to produce an output, known as a feature map, which is passed forward through the CNN. However, unlike nodes in dense layers, the transformation is a convolution operation on the input weights. In the 2D case, a filter of size f x f slides, or convolves, across a f x f block of a larger image with M x M pixels. Elements in the feature map are calculated by summing the element-wise products of matching elements in the block of pixels and the filter. Then, a particular number of horizontal and vertical steps are taken by the filter, known as the stride s, until the entire input

has been convolved. A smaller stride value allows for the detection of smaller spatial features in the input image at the cost of increased computation. An example of a convolution operation with a stride of 1 is shown in Figure 8.

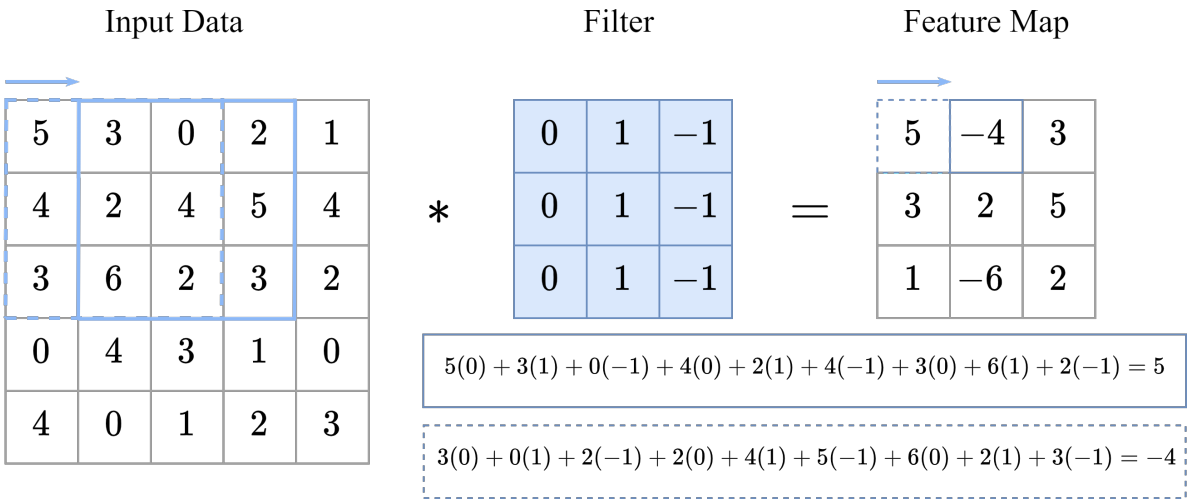

Figure 8: A convolution operation example of a 3 x 3 filter convolving an input image with 5 x 5 pixels and stride $s = 1$. The elements in the matrices represent the brightness values of the pixels. Full calculations for the first and second elements of the feature map are provided.

For a monochromatic input image where positive pixel values represent bright pixels and negative values represent dark pixels, the filter shown in Figure 8 is capable of detecting right edges. Therefore, this filter can be rotated to detect top, bottom and left edges as features in an input image. An example of this is shown in Figure 9. As shown in Figure 8, the feature map output has a reduced dimension size compared to the input data due to the number of unique block positions where the filter can be placed. As this output is passed to other convolutional layers deeper in the CNN, the resulting dimensions of the output may become significantly reduced. As a result, filters in these deep layers may be incapable of extracting complex features. This issue can be mitigated by a technique called zero padding that allows for the preservation of the original input dimensions [24, 27]. This involves adding a border of zero-value pixels around the input image. Along a given dimension, the padding of P zeroes on both sides of the axis increases the dimensions by $2P$. In general, the dimensions of the feature-map output with size M' x M' can be calculated as [28]

$$M' = \frac{M - f + 2P}{s} + 1 \tag{19}$$

Input images may contain more than one value for each pixel. For example, RGB (red, green and blue) images contain three values, referred to as channels, for each pixel. In this case, multiple filters can convolve each channel individually producing multiple feature maps that can be combined in the output [23, 28]. For example, convolving each channel in an RGB image with five filters produces 15 feature maps with colour-specific feature extraction. Typically, the number of filters is doubled with each added convolutional layer due to the increased number of

pattern combinations that can be made from complex shapes.

Since the total number of trainable parameters in a convolutional layer is proportional to the number of filters, the computational load increases significantly with adding convolutional layers. To mitigate this effect, a pooling layer can be inserted after a convolutional layer. Particularly in high-resolution images, the difference between the channel values in adjacent pixels generally corresponds to trivial features [23]. These pooling layers aim to eliminate these redundancies and provide invariance to the feature maps under small translations and rotations of the input image [22, 28]. This reduces overfitting to the data, a frequent problem encountered during training.

The most common type of pooling is max pooling. This involves a filter, typically of size 2 x 2 with a stride of 2, which traverses the feature maps (obtained in the previous convolutional layer) in a similar manner to the convolution operation shown in Figure 8. However, the value for each element in max-pooling output is calculated by determining the maximum value in the corresponding region highlighted by the filter, as shown in Figure 9 (a). Figure 9 (b) shows an input image of the number '4' being passed through a convolutional layer with six filters capable of detecting different types of edges. These are passed on to a max-pooling layer that compresses the features into a lower-resolution feature map. The size of this output image is given by Equation 19 with $P = 0$ as the max-pooling layer does not involve zero padding [28].

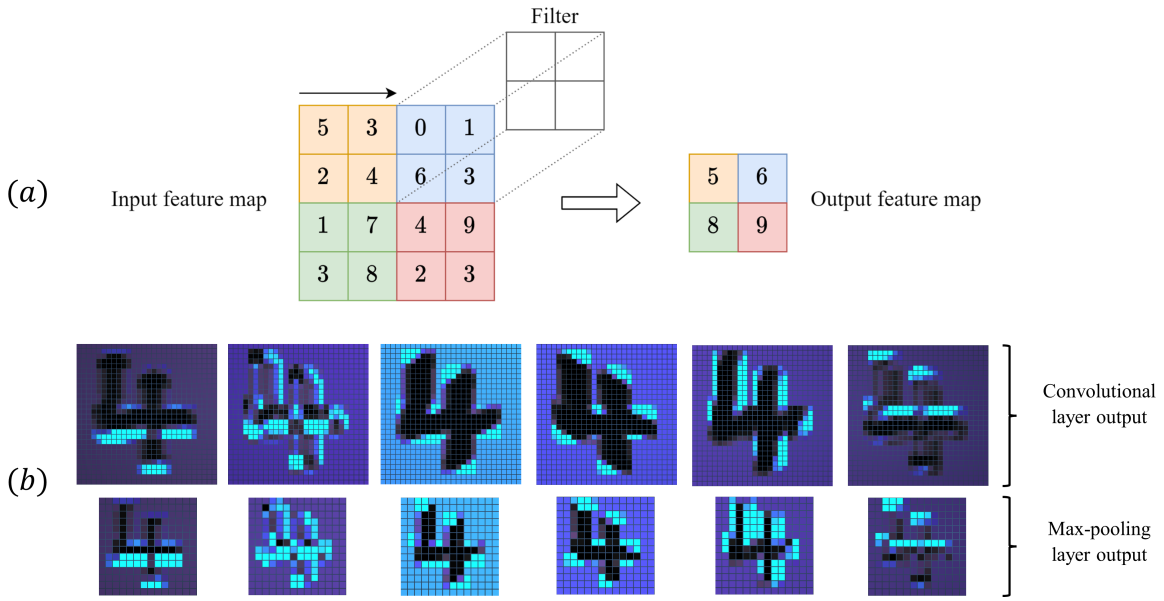

Figure 9: (a) An example to illustrate the max-pooling operation performed in a max-pooling layer. A 2 x 2 filter with stride $s = 2$ pools a 4 x 4 input map to produce an output map with reduced dimensions. (b) A diagram illustrating a 28 x 28 input of the number '4' passing through a convolutional layer with six filters, capable of detecting features, such as left, right, bottom and top edges. These feature maps pass through a max-pooling layer that halves each dimension, resulting in six 14 x 14 feature maps. These images were created using a visualisation tool provided in [29].

To further increase computational efficiency and reduce overfitting of the model, the input data can be normalised, or standardised if the input data is Gaussian distributed, during pre-

processing [24, 27]. Since large values in non-normalised data may also cause instability and reduce training speed, all values in the input data are divided such that they lie on the same scale, usually 0 to 1. The input data can also be divided into a series of batches, usually with 32 samples per batch, for increased efficiency. Large, outlying weights may also be induced by SGD during weight optimisation that can cascade throughout the network, causing instability. This can be mitigated by batch normalisation which normalises the activation outputs of each batch in the preceding layer.

3.4 Overfitting

To evaluate a model's predictive capability on unseen data, as is the case in adaptive radiotherapy (ART), the input data is typically divided into a training, validation and a test set with typical sizes of 70%, 15% and 15%, respectively [22, 27]. When the training set is passed to the NN, the training loss is determined for each forward pass (epoch) of this entire set as the NN attempts to fit to the training data. Then, trainable parameters are updated via backpropagation. Multiple models with different selected parameters (hyperparameters), such as the number of convolutional layers, filters, etc., may be constructed and trained on this set. The generalisability of each model can then be evaluated using the validation set. During each epoch in the training stage, a sample from the validation set is periodically passed to the NN to measure the accuracy of current predictions to unseen data, producing a validation loss value. Trainable parameters are not updated during this stage. A low training loss paired with a high validation loss is a sign of overfitting [23]. Therefore, the validation set provides a tool to monitor this issue. After each model has been trained and validated, generally, the model with the best validation performance is selected for evaluation on the test set [22, 27].

Unlike the training and validation set, no labels are provided to the model during testing. Ideally, the model will fit equally well to the training and test set, indicating strong generalisability to unseen data, which is quantified by the generalisation error. This is the expected value of the loss on a set of unseen data that is perfectly representative of all plausible scenarios that a model was employed to predict on [22]. Since the test set is only a small sample of all scenarios, the performance of the model on the test set, measured by a metric such as MSE, is an estimate of the generalisation error. Furthermore, since the training set is also not perfectly representative of all plausible scenarios, overfitting to the training set is frequently observed. While the best solution for minimising overfitting is by training on more data, this is not feasible in many medical applications [22, 27]. As a result, other solutions include regularisation techniques, defined as any modification to the model that is intended to reduce the generalisation error but not the training error [22]. One such technique is data augmentation [27] which modifies the training data by a transformation, such as small translations, rotations or blurring by adding noise, and passes it to the model as a new input but with the same label. This simulates the addition of more training data, though, the fitting to this data reduces variance, overfitting and thus, increases generalisability. As images obtained through deformable image registration (DIR) may contain various imperfections, data augmentation is an essential procedure in a CNN-based QA of a DIR.

4 Method

4.1 Input Data

The input data for the CNN model constructed in this project were obtained from the Head-Neck Cetuximab collection in the Cancer Imaging Archives [30]. This consisted of 3D PET-CT scans of the brainstem from 94 patients. Each 3D head scan was sliced axially (top-down) to produce multiple 2D slices with 512 x 512 pixels. Although Section 2.2 states that DIR in ART is applied on two images (floating and reference) taken on two separate sessions of a patient undergoing anatomical changes, this was not possible due to the limited amount of available data. However, a single patient at two different times was simulated by performing DIR on two 3D scans corresponding to two different patients, known as inter-patient registration. The procedure outlined in 2.2 was applied, which is explained below.

In this pair, one scan was defined to be the floating scan and the other as the reference scan, both containing manually-drawn contours. From the 94 patients' scans, 1024 such pairings were produced [14, 15, 16]. Recalling that the objective of DIR is to produce a mapping that corresponds to anatomical changes only, the contours on both scans were removed before DIR was applied. Applying DIR to these 1024 pairings of 3D scans produced 1024 registered scans which were split into 18451 registered slices with a corresponding DVF for each slice. The DVF was then applied to the floating scan contour which produced the registered (DIR-produced) contour. This registered contour was then compared to the manually-drawn contour on the reference scan (ground truth) using the 3D-MSD similarity metric. Hence, a metric value was produced for each reference/registered scan (both scans are identical if contours are not present). This metric value is the label supplied to the model.

Each 2D reference slice (without a contour) was concatenated with the corresponding DIR-produced DVF. Then, all slices were appended together into a tensor object. This was the input to our model. As mentioned in Section 3.3, each pixel can contain multiple channels, generally corresponding to the RGB channels. In this project, there are four channels: one colour channel (brightness) and three DVF channels (corresponding to the x-, y- and z-components of the DVF). Given the DVF and reference image for each slice, the model was tasked with predicting the 3D-MSD value. The loss function used to evaluate the model's performance during training and validation was the mean squared error (MSE), given by Equation 10.

Due to the large size of the input data (155 GB) and limited memory (VRAM), numerous pre-processing steps were made to maximise computational efficiency when training and validating the model. Firstly, the training and validation NumPy arrays were saved on disk. Then, these were loaded into our Python environment as memory-mapped arrays [31]. This type of array allows for accessing small quantities of the entire data from the disk without the need to read the entire file into memory. Then, these memory-mapped arrays were converted into a 'Dataset' object which allowed the small data quantities to be sequentially fed into the model [32].

4.2 Model

The CNN model used in this project was originally based on the model found in [14, 15, 16]. Iterative modifications to the hyperparameters were made, resulting in the current model illustrated in Figure 10. The current hyperparameters are stated: the batch size was set to 12 2D slices with each slice having size 512 x 512 pixels and 4 channels in the input. The number of filters in the first, second and third convolutional layer was 32, 64 and 128, respectively. Each filter had size 3 x 3 with stride $s = 1$. All max-pooling layers had size 2 x 2 with stride $s = 2$. The choice of optimiser was 'RMSprop' (Root Mean Square Propagation), which is an extension to SGD. The only activation function used was ReLU and the learning rate η was set to 0.01. A flattening layer is added to convert the multidimensional tensor into a one-dimensional array which can be fed into the final dense layer for prediction.

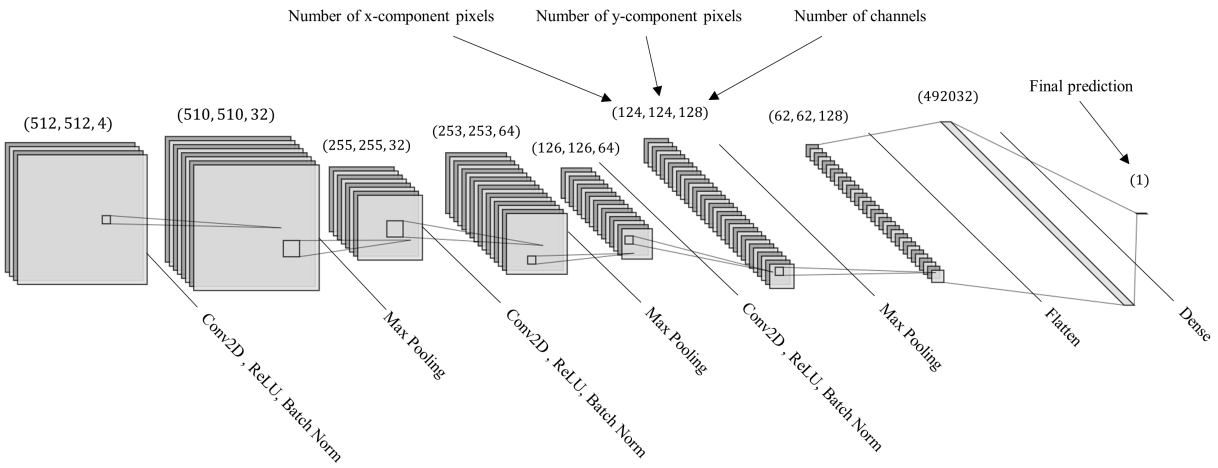

Figure 10: An illustration of the CNN structure used in this project. The type of layer and dimensions of a single 2D scan are indicated on the diagram. This image was produced using a visualisation tool provided in [33].

5 Results and Discussion

The CNN ran for 20 epochs and the training and validation loss was monitored over this course, shown in Figure 11. Ignoring the validation peak on the 8^{th} epoch, the training and validation loss are approximately equal from the 5^{th} to the 15^{th} epoch. This indicates that the model is not overfitting during this period. However, from the 15^{th} epoch onwards, a significant difference between the validation and training loss is observed which may suggest overfitting and an increasing generalisation error. The optimal number of epochs cannot be obtained from the figure as the training loss is still decreasing in the 20^{th} epoch. Overfitting may be due to several aspects observed in the 2D scans. For example, noise and artefacts present on the scans may be 'memorised' by the model as general features, especially if these are present on the first several scans. These correspond to multiple local minima on a graph of the loss function plotted against

the weights. As a result, the gradient descent algorithm (RMSprop) may become stuck in any of these minima.

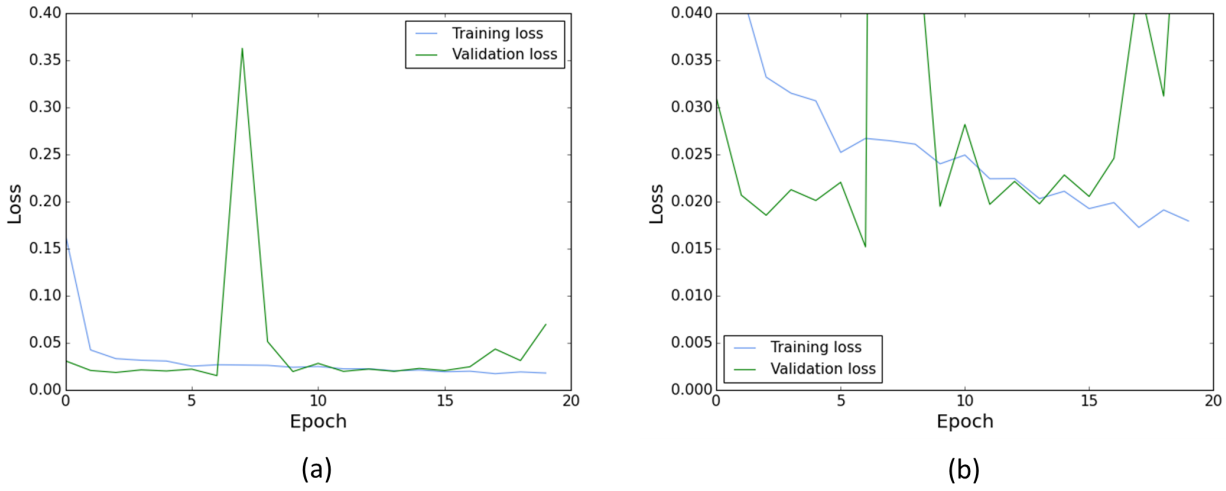

Figure 11: (a) A graph of the training and validation loss over 20 epochs, obtained during training of the CNN model. The graph shows a steadily decreasing training loss where the lowest value of 0.0173 was reached on the 18^{th} epoch. (b) A magnified version of (a).

Overfitting can be alleviated by reducing the complexity of the model. One such method could be achieved by removing the final convolutional layer whose filters are producing feature maps of excessive depth. Removing the following max-pooling layer may also reduce overfitting as this layer enhances the intricacies of the input feature maps. Instead, an average-pooling layer can replace this which may consolidate the feature map depth as opposed to exceeding it. Another method could be reducing the rate that complexity increases throughout the model. This may be a preferable attempt at a solution due to the nature of the 2D CT scans. As mentioned in Section 3.3, the number of filters are typically doubled with each successive convolutional layer added; since the feature complexity increases moving through the network, the number of ways to combine these features to produce features with an even higher complexity increases. However, in this project, CT scans are composed of relatively simple 2d shapes in a single colour channel. As a result, the rate of feature complexity may diminish early on in the network. While there are three more channels for the DVF components, variance in the field diminishes with reducing pixel dimensions in the feature map. Thus, the field complexity may be diminished at a particular depth in the network.

It is unclear why the validation loss is lower than the training loss during the first several epochs. While the model makes use of regularisation techniques, this is only applied during training but is deactivated during validation which may be a possible explanation. Also, there is a possibility that the patient data in the validation set are less varied (by chance) than the training set. The large spike in validation loss is a common consequence of the limited batch size. Some batches during the 8^{th} epoch likely contained samples that are unrepresentative of samples found in the training set. This divergence can often be magnified by the batch-

normalisation implementation in TensorFlow. Besides removing the batch-normalisation layer, the ratio between the training and validation sets can be decreased for increased representation in the validation set. Overall, these preliminary results have significant potential for improvement through various untested optimisation techniques.

6 Conclusion

A 2D convolutional neural network was constructed to perform quality assurance of brainstem contours on PET-CT scans, produced by deformable image registration. These contours were compared to ground truth contours by a similarity metric known as the 3D-mean-surface-distance (3D-MSD) which is a modification to the Hausdorff distance. The model was trained to predict similarity metrics by supplying it with reference CT scans and its corresponding displacement vector field (DVF; produced by the registration of the reference scan with a floating scan) as the input, and the associated similarity metric as the label. The model successfully ran for 20 epochs with a steadily decreasing loss. An increasing validation loss during the final epochs was observed, indicating overfitting of the model to the training set. Potential mitigation of this issue may be achieved by reducing model complexity or through advanced regularisation techniques. If successful, the introduction of artificial errors by data augmentation, such as translations, rotations and added noise, is necessary to simulate defects that are commonly produced during DIR in real-life clinical scenarios. If the model is invariant under these augmentations, a threshold for similarity metric predictions that quantify the registration quality can be determined by clinicians. Thus, low predictions determined by the CNN during an adaptive radiotherapy session can be flagged to the clinical team for manual review.

References

[1] Cancer Research UK, "What is radiotherapy?" https://www.cancerresearchuk.org/about-cancer/cancer-in-general/treatment/radiotherapy/about, Nov. 2020, [Accessed: 28-12-2021].

[2] A. Torgovnick and B. Schumacher, "DNA repair mechanisms in cancer development and therapy," *Frontiers in Genetics*, vol. 6, p. 157, 2015. [Online]. Available: https://www.frontiersin.org/article/10.3389/fgene.2015.00157

[3] C. B. Dracham, A. Shankar, and R. Madan, "Radiation induced secondary malignancies: a review article," *Radiation Oncology Journal*, vol. 36, no. 2, pp. 85–94, jun 2018. [Online]. Available: https://doi.org/10.3857%2Froj.2018.00290

[4] Cancer.org, "How radiation therapy is used to treat cancer," https://www.cancer.org/treatment/treatments-and-side-effects/treatment-types/radiation/basics.html, Dec.

[5] B. Jones and D. A. Morgan, "Radiotherapy fractionation," in *Radiobiological Modelling in Radiation Oncology*. The British Institute of Radiology, Jan. 2007, pp. 51–78. [Online]. Available: https://doi.org/10.1259/9780905749839.chapter04

[6] K. K. Brock, S. Mutic, T. R. McNutt, H. Li, and M. L. Kessler, "Use of image registration and fusion algorithms and techniques in radiotherapy: Report of the AAPM radiation therapy

committee task group no. 132," *Medical Physics*, vol. 44, no. 7, pp. e43–e76, may 2017. [Online]. Available: https://doi.org/10.1002%2Fmp.12256

[7] S. Oh and S. Kim, "Deformable image registration in radiation therapy," *Radiation Oncology Journal*, vol. 35, no. 2, pp. 101–111, jun 2017. [Online]. Available: https://doi.org/10.3857%2Froj.2017.00325

[8] D. Yan, F. Vicini, J. Wong, and A. Martinez, "Adaptive radiation therapy," *Physics in Medicine and Biology*, vol. 42, no. 1, pp. 123–132, jan 1997. [Online]. Available: https://doi.org/10.1088%2F0031-9155%2F42%2F1%2F008

[9] S. Arculeo, E. Miglietta, F. Nava, A. Morra, M. C. Leonardi, S. Comi, D. Ciardo, M. S. Fiore, M. A. Gerardi, M. Pepa, S. G. Gugliandolo, L. Livi, R. Orecchia, B. A. Jereczek-Fossa, and S. Dicuonzo, "The emerging role of radiation therapists in the contouring of organs at risk in radiotherapy: analysis of inter-observer variability with radiation oncologists for the chest and upper abdomen," *ecancermedicalscience*, vol. 14, jan 2020. [Online]. Available: https://doi.org/10.3332%2Fecancer.2020.996

[10] K. K. Brock, "Adaptive radiotherapy: Moving into the future," *Seminars in Radiation Oncology*, vol. 29, no. 3, pp. 181–184, jul 2019. [Online]. Available: https://doi.org/10.1016%2Fj.semradonc.2019.02.011

[11] M. F. Chan, A. Witztum, and G. Valdes, "Integration of AI and machine learning in radiotherapy QA," *Frontiers in Artificial Intelligence*, vol. 3, sep 2020. [Online]. Available: https://doi.org/10.3389%2Ffrai.2020.577620

[12] N. Hardcastle, W. A. Tomé, D. M. Cannon, C. L. Brouwer, P. W. Wittendorp, N. Dogan, M. Guckenberger, S. Allaire, Y. Mallya, P. Kumar, M. Oechsner, A. Richter, S. Song, M. Myers, B. Polat, and K. Bzdusek, "A multi-institution evaluation of deformable image registration algorithms for automatic organ delineation in adaptive head and neck radiotherapy," *Radiation Oncology*, vol. 7, no. 1, jun 2012. [Online]. Available: https://doi.org/10.1186%2F1748-717x-7-90

[13] W. J. Beasley, A. McWilliam, N. J. Slevin, R. I. Mackay, and M. van Herk, "An automated workflow for patient-specific quality control of contour propagation," *Physics in Medicine and Biology*, vol. 61, no. 24, pp. 8577–8586, nov 2016. [Online]. Available: https://doi.org/10.1088%2F1361-6560%2F61%2F24%2F8577

[14] L. Walker, "Quality Assurance of Image Registration in Radiotherapy using Deep Learning," *University of Manchester*, 2020.

[15] L. Fernández, "Deep Learning for Quality Assurance of Image Registration," *University of Manchester*, 2020.

[16] S. Bridger, "Deep Learning Deformable Image Registration Quality Assurance Summer 2021: A Summary of Findings," *University of Manchester*, 2021.

[17] R. Shekhar, P. Lei, C. R. Castro-Pareja, W. L. Plishker, and W. D. D'Souza, "Automatic segmentation of phase-correlated CT scans through nonrigid image registration using geometrically regularized free-form deformation," *Medical Physics*, vol. 34, no. 7, pp. 3054–3066, jun 2007. [Online]. Available: https://doi.org/10.1118%2F1.2740467

[18] S. Broggi, E. Scalco, M. L. Belli, G. Logghe, D. Verellen, S. Moriconi, A. Chiara, A. Palmisano, R. Mellone, C. Fiorino, and G. Rizzo, "A comparative evaluation of 3 different free-form deformable image registration and contour propagation methods for head and neck MRI: The case of parotid changes during radiotherapy," *Technology in Cancer Research & Treatment*, vol. 16, no. 3, pp. 373–381, feb 2017. [Online]. Available: https://doi.org/10.1177%2F1533034617691408

[19] R. Varadhan, G. Karangelis, K. Krishnan, and S. Hui, "A framework for deformable image registration validation in radiotherapy clinical applications," *J. Appl. Clin. Med. Phys.*, vol. 14, no. 1, p. 4066, Jan. 2013.

[20] A. A. Taha and A. Hanbury, "Metrics for evaluating 3d medical image segmentation: analysis, selection, and tool," *BMC Medical Imaging*, vol. 15, no. 1, aug 2015. [Online]. Available: https://doi.org/10.1186%2Fs12880-015-0068-x

[21] D. Huttenlocher, G. Klanderman, and W. Rucklidge, "Comparing images using the hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 850–863, 1993. [Online]. Available: https://doi.org/10.1109%2F34.232073

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. London: MIT Press, 2016.

[23] S. Skansi, *Introduction to deep learning: From logical calculus to artificial intelligence*, 1st ed. Cham, Switzerland: Cambridge University Press, 2018.

[24] E. Charniak, *An introduction to deep learning*. Boston, MA, USA: Addison-Wesley Educational, 2018.

[25] K. Gurney, *An introduction to neural networks*, 1st ed. London: UCL Press, 1997.

[26] Deep Lizard, "Deep Learning Fundamentals - Intro to Neural Networks," https://rb.gy/mmc1a8, Nov. 2017, [Accessed: 04-01-2021].

[27] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, jun 2018. [Online]. Available: https://doi.org/10.1007%2Fs13244-018-0639-9

[28] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *ArXiv*, vol. abs/1603.07285, 2016.

[29] A. W. Harley, "An interactive node-link visualization of convolutional neural networks," in *ISVC*, 2015, pp. 867–877.

[30] W. R. Bosch, W. L. Straube, J. W. Matthews, and J. A. Purdy, "Data from Head-Neck_Cetuximab," 2015. [Online]. Available: http://doi.org/10.7937/K9/TCIA.2015.7AKGJUPZ

[31] NumPy Documentation, "numpy.memmap," https://numpy.org/doc/stable/reference/generated/numpy.memmap.html, Nov. 2021, [Accessed: 05-01-2021].

[32] TensorFlow Documentation, "tf.data.dataset," https://www.tensorflow.org/api_docs/python/tf/data/Dataset, Sep. 2021, [Accessed: 05-01-2021].

[33] A. LeNail, "NN-SVG: Publication-ready neural network architecture schematics," *Journal of Open Source Software*, vol. 4, no. 33, p. 747, jan 2019. [Online]. Available: https://doi.org/10.21105%2Fjoss.00747