DEEP LEARNING APPROACHES TO MULTI-MODAL BIOMEDICAL IMAGE SEGMENTATION

Nelson Frank

A thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

January 2021

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Nelson Frank**

Entitled: **Deep Learning Approaches to Multi-Modal Biomedical Image Segmentation**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Examiner

Dr. Andrew Delong

_____ Examiner

Dr. Yiming Xiao

_____ Supervisor

Dr. Marta Kersten-Oertel

Approved _____

Dr. Lata Narayanan, Chair of Department

_____ 20 _____ _____

Dr. Mourad Debbabi,
Interim Dean, Gina Cody School of Engineering and Computer Science

Abstract

Deep Learning Approaches to Multi-Modal Biomedical Image Segmentation

Nelson Frank

Deep learning techniques have been shown to produce state-of-the-art performance in segmenting biomedical images. These techniques, however, are highly dependent on the quantity and quality of available training data, as well as their capacity to represent complex relationships, which is dependent on the size of the network and limited by the computational resources of the machine(s) on which they are trained. In this work, we performed two experiments. First, we explored multi-stream model configurations that can leverage available data from multiple *unpaired* biomedical imaging modalities to learn a shared representation. Specifically, segmentation of cardiac CT and MRI was done to see if this learns the shared anatomical features and thus performs better than individual models trained on each modality. Second, we compared our full deep learning segmentation pipeline as applied to *paired* multi-modal brain images against other existing publicly available pipelines. From these experiments, we found that (1) multi-stream architectures can achieve better results in *unpaired* multi-modal segmentation compared to single-stream models, however, the specific configuration with the best performance is in disagreement with previously published results; and (2) careful adjustment of deep learning pipeline configurations to our specific data set and hardware constraints yields improved segmentation accuracy over publicly available state-of-the-art solutions in *paired* multi-modal image segmentation.

Acknowledgments

The completion of this dissertation marks the culmination of my formal education. While I would love to take full credit for my achievements, nothing is done in a vacuum. My parents worked hard to support me and ensure that higher education was easily accessible. My supervisor Dr. Marta Kersten supported me with ideas, suggestions, guidance, and opportunities. Dr. Louis Collins provided guidance and insights during my time in his lab. My peers in the Applied Perception Lab made a welcoming environment and exposed me to many new ideas. Drs. Tristan Glatard, Charalambos Poullis, and Javad Sadri were excellent teachers during my undergrad (among others!) and provided their recommendations for me to enter grad school in the first place. To all of you, thank you!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the early 1970s, computed tomographic (CT) imaging was developed by Godfrey Hounsfield [1]. At the same time, Raymond Damadian determined that nuclear magnetic resonance (MR) could be used to detect brain tumours [2]. Since their introductions, both imaging modalities have become indispensable tools for anatomical visualisation as they are completely non-invasive methods that can provide invaluable clinical information. As such, their uses include, but are certainly not limited to, assessing trauma and anatomical morphology and detecting and delineating pathology such as tumours, infarctions, and aneurysms.

Many of these tasks are very time consuming and must be done by expert clinicians, whose time is particularly valuable. For diagnostic tasks, manual assessment is subject to error with one meta-analysis concluding that about one in ten patients with major vascular events, infections, or cancers are misdiagnosed, half of whom suffer either permanent disability or death as a result [3]. That same study found that less common diseases faced higher rates of misdiagnosis. For delineation tasks, even when performed by highly skilled clinicians, there is a high degree of inter- and intra- rater variability [4]. For these reasons there has been a push to make these tasks more efficient and consistent using automated computer algorithms.

1.1 Medical Image Segmentation

In this thesis, we focus on the specific task of medical image segmentation, i.e. the partitioning of an image into discrete semantic segments. This challenging task is

one of the most essential medical imaging processes, as it aims to identify the pixels that belong to specific organs or lesions (i.e. a region of interest or ROI). Segmentation can be used to perform anatomical delineation directly or to transform images into a simpler form that is amenable to further algorithmic analysis. For example, segmentation of the aortic root from CT angiography images enables automatic vessel measurement for the pre-procedural planning of a trans-catheter aortic valve implantation procedure [5]. Segmentation can also be used for determining tumour boundaries for pre-procedural planning of a resection surgery [6]. An example of biomedical image segmentation is show in Figure 1.

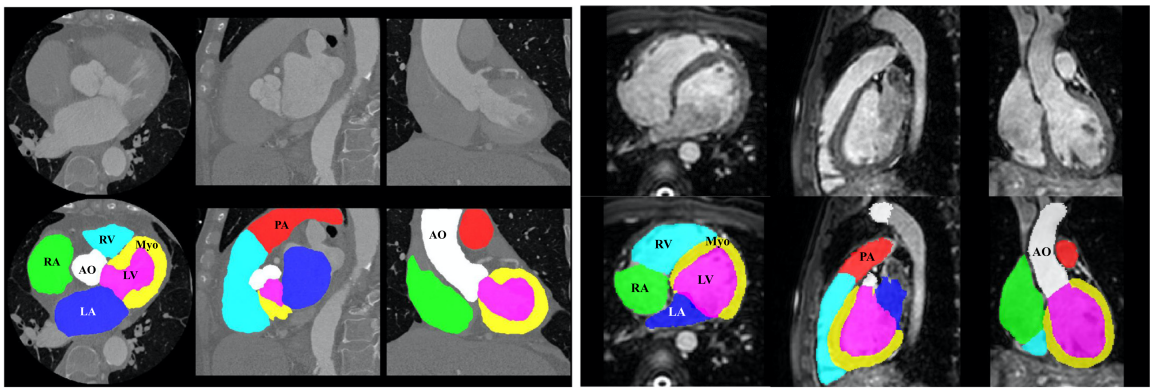


Figure 1: An example of biomedical image segmentation. CT images are on the left and MR images are on the right. The bottom row shows the whole-heart images segmented into seven cardiac substructures. Figure from the Multi-Modality Whole Heart Segmentation challenge [7].

There exist a number of traditional techniques for biomedical image segmentation. Thresholding algorithms assign image elements to a given class when their associated intensities are within the bounds of some thresholds. These thresholds can be assigned using a priori knowledge or determined procedurally using algorithms such as K-Means. Thresholding algorithms work best when the target structure's intensities are relatively homogenous and contrastive, properties that are uncommon to structures of interest in biomedical images [8].

Region growing is a type of segmentation algorithm that uses initial seed points and adds neighbouring image elements if they fit some inclusion criteria. This process is then repeated with the newly included image elements until no remaining image elements fit the inclusion criteria. Region growing often performs better than thresholding as it includes spatial information, however, it depends on the target structure

being relatively homogenous and having high contrast [9]. Additionally, it requires the selection of seed points that needs to be done either manually or by a separate algorithm.

Multi-Atlas segmentation (MAS) is a technique that uses a number of pre-labeled training images (atlases) to segment new images [10]. These atlases are registered to the target image through iterative deformations that seek to maximize spatial alignment while, at the same time, maximizing the plausibility of the deformations. The atlas with the best fit is selected and its deformed labels are used for the new segmentation.

Handcrafted custom algorithms can be designed using a priori knowledge specific to the task at hand. For example, for segmentation of the aortic artery, one system takes advantage of the fact that it appears as a series of circles in axial slices and simply uses a Hough transform circle detector to localize it [11]. Such systems are specific to the tasks for which they were designed and tend to not generalize to new tasks.

While these techniques are still being used today, deep learning based techniques have become prominent for biomedical image segmentation.

1.2 Deep Learning

Deep learning is a type of machine learning methodology that learns both the base features present in data and the relationships between the features to develop high level insights using artificial neural networks. Deep learning for image processing came into prominence in 2012 when Alexnet, a convolutional neural network (CNN), won ImageNet's Large Scale Visual Recognition challenge (LSVR) tasks by enormous margins [12]. Since then, every LSVR challenge winner has been a convolutional neural network. There has been a similar trend for medical segmentation competitions which feature almost exclusively deep learning techniques due to their overwhelming performance [7, 13, 14].

While deep learning has been shown to have excellent performance on image processing tasks, there are some specific issues constraining the performance of deep learning approaches. The outcome of deep learning solutions is highly dependent on the quantity and quality of training data which, especially for medical images, is not

always easily available [15]. Further, the capacity for a neural network to fit complex relationships is constrained by its width and depth. As the network grows, so too does the amount of necessary processing and memory. In this thesis, we present results from our experiments seeking to address these specific issues through network configurations capable of learning from multiple modalities thereby expanding the amount of usable data and through a holistic deep learning solution tuned precisely to operate on the specific data and computation constraints at hand.

1.3 Contributions

In the course of this research, we performed two experiments in the domain of multi-modal biomedical image segmentation. The first experiment sought to validate current practices in multi-stream approaches to *unpaired* multi-modal segmentation in the context of cardiac CT and MR images. In the second experiment we sought to validate a developed full deep learning segmentation pipeline as applied to *paired* multi-modal images against other existing publicly available pipelines. The results of our two studies showed the following:

1. Multi-stream architectures can achieve better results in *unpaired* multi-modal segmentation compared to single-stream models, however, the specific configuration with the best performance is in disagreement with previously published results.

2. Careful adjustment of deep learning pipeline configurations to our specific data set and hardware constraints yields improved segmentation accuracy over publicly available state-of-the-art solutions in *paired* multi-modal image segmentation.

1.4 Organization of the Dissertation

The remainder of this dissertation is organized as follows. We begin in Chapter 2 with a background on deep learning using artificial neural networks and describe many considerations and approaches in their design. We also describe the physical properties captured by different biomedical imaging modalities and discuss aspects

of deep learning specific to biomedical image segmentation. In Chapter 3, we present our experiment in measuring the efficacy of different multi-stream architectures as applied to *unpaired* multi-modal image segmentation. In Chapter 4, we present our experiment validating the performance of our deep learning pipeline as applied to *paired* multi-modal image segmentation. Chapter 5 concludes by summarizing the results of the presented works and suggests opportunities for future works.

Chapter 2

Background

2.1 Deep Learning for Image Processing

Traditional machine learning techniques use handcrafted features and determine relationships between them algorithmically. The defining characteristic of deep learning techniques, in comparison to other machine learning methods, is that deep learning does not rely on handcrafted features, rather features and the relationships between the features are learned *on their own*. In this section, we discuss the core concepts pertaining to deep learning in the specific domain of image processing.

2.1.1 Supervised and Unsupervised Learning

Machine learning algorithms can largely be divided into two classes: supervised learning and unsupervised learning. Their distinction lies in the presence or absence of human-provided ground-truth labels.

In supervised learning, the models are trained on input, ground-truth tuples so that the model can learn a general function mapping input to the expected ground truth. In image processing, typical tasks for supervised learning include, but are not limited to the following:

- **Classification:** Assigning semantic categorical labels to an image from a predefined set of labels, (e.g. determining if a picture is of a cat or a dog).
- **Detection:** Identification of one or more objects in an image with positional information (e.g. bounding boxes around pedestrians in street imagery).

- **Segmentation:** Assignment of semantic categorical labels to each of the elements of a given image (e.g. determining what pixels or voxels are part of a liver).

In unsupervised learning, models are trained without any provided ground-truth labels. These models identify patterns in unstructured or weakly structured data. Unsupervised learning can be used to pre-train supervised learning models using auto-encoders [16] (a type of artificial neural network) or to train generative adversarial networks (GANs) [17]. GANs are a pair of networks trained against each other's outputs. One network is tasked with generating new data with similar characteristics to the training data, while the other network is tasked with distinguishing these generated data from real examples in the training data.

2.1.2 Artificial Neural Networks

Artificial neural networks, often called neural networks (NN), are a type of machine learning technique loosely inspired by the neuronal connections in the human brain. In this subsection, we discuss the organization and mathematical basis that allows NNs to learn.

Forward Pass

Neural networks are composed of many simple units called neurons (see Figure 2). Each neuron is made up of three distinct parts: an input, a hidden layer, and an output. The input can be represented as an arbitrarily sized vector of scalar values. This size is often fixed, although not necessarily. In the hidden layer, for each scalar value in the input, there is an associated scalar weight. The associated values and weights are multiplied together and their products summed, creating a linear transformation. The result is then transformed into the final output by an auxiliary function called the activation function, which usually defines a non-linear transformation allowing the neuron to represent non-linear patterns. Activation functions are further discussed later in this section.

A neuron in a neural network is defined using a non-linear function $f : \mathbb{R}^n \to \mathbb{R}$

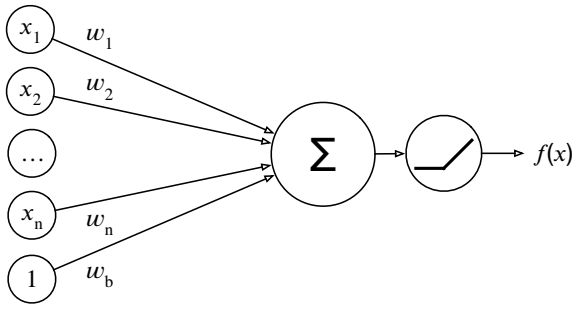


Figure 2: Diagram representing a simple example of a fully connected neuron. The diagram was adapted from [18].

with weights $\vec{w} = [w_1, w_2, w_3, \cdots, w_n]^\intercal$, and activation function g.

$$h(\vec{x}) = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \tag{1}$$

$$f(\vec{x}) = (g \circ h)(\vec{x}) \tag{2}$$

An example of this relationship is illustrated in Figure 2. A single neuron has only a single scalar output. By combining m neurons in one layer, one can define a similar non-linear function $f : \mathbb{R}^n \to \mathbb{R}^m$ given by:

$$h(\vec{x}) = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{3}$$

$$f(\vec{x}) = (g \bigcirc h)(\vec{x}) \tag{4}$$

Where the $\bigcirc$ denotes the element-wise application of the function.

Layers of neurons can be composed together so that the output of one layer feeds forward into the input of a subsequent layer. An example of such connections is shown in Figure 3. Given n such individual layers denoted as:

$$\dot{f}^{(1)}(\vec{x}), \dot{f}^{(2)}(\vec{x}), \dot{f}^{(3)}(\vec{x}) \cdots \dot{f}^{(n)}(\vec{x}), \tag{5}$$

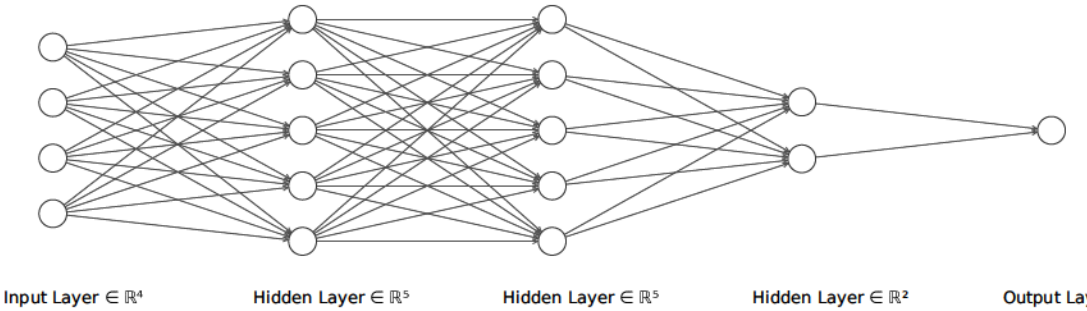


Figure 3: Diagram representing a simple example of a fully connected neural network design. The diagram was created using the online tool NN-SVG [19].

one can recursively define a larger network as:

$$f^{(1)}(\vec{x}) = \dot{f}^{(1)}(\vec{x}) \tag{6}$$

$$f^{(i)}(\vec{x}) = (\dot{f}^{(i)} \circ f^{(i-1)})(\vec{x}). \tag{7}$$

Where $\dot{f}^{(i)}(\vec{x})$ denotes the functions for the individual layer i and $f^{(i)}(\vec{x})$ denotes the functions yielding the connected network's intermediate output by layer i.

Back Propagation

The previous section describes how a neural network performs inference given a set of weights. In this section, we describe how these weights are iteratively determined by a process called stochastic gradient descent, powered by back propagation.

After a forward pass through the network, the output is fed into a loss function $L(\vec{x})$ which returns a scalar value representing the amount of error or cost in the network's output. Loss functions are discussed further in a following subsection. Stochastic gradient descent updates each of the weights of a network by an amount proportional to that weight's contribution to the loss function. The sequential update of the layer's weights W_i^t to W_i^{t+1} is given in equation 8, adapted from [20], where η is the learning rate, a hyper-parameter controlling the rate at which weights are updated.

$$W_i^{t+1} = W_i^t - \eta \frac{\partial L(\vec{x})}{\partial W_i^t} \tag{8}$$

This can be calculated thanks to the chain rule of calculus which states that given

functions $y = f(u)$ and $u = g(x)$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}. \tag{9}$$

We redefine the layer functions in equations 6 and 7 such that the layer's weights W_i are a parameter to the function rather than a property of the function, e.g., $\dot{f}^{(i)}(\vec{x})$ becomes $\dot{f}^{(i)}(\vec{x}, W_i)$. Then using the chain rule, we can derive the following equations adapted from [21]:

$$\frac{\partial L(\vec{x})}{\partial W_{n-1}} = \frac{\partial L(\vec{x})}{\partial f^{(n)}(\vec{x}, W_n)} \cdot \frac{\partial f^{(n)}(\vec{x}, W_n)}{\partial f^{(n-1)}(\vec{x}, W_{n-1})} \tag{10}$$

and for arbitrary layers, we have:

$$\frac{\partial L(\vec{x})}{\partial W_i} = \frac{\partial L(\vec{x})}{\partial f^{(n)}(\vec{x}, W_n)} \cdot \frac{\partial f^{(n)}(\vec{x}, W_n)}{\partial f^{(n-1)}(\vec{x}, W_{n-1})} \cdot \frac{\partial f^{(n-1)}(\vec{x}, W_{n-1})}{\partial f^{(n-2)}(\vec{x}, W_{n-2})} \cdots \frac{\partial f^{(i)}(\vec{x}, W_i)}{\partial f^{(i-1)}(\vec{x}, W_{i-1})} \tag{11}$$

where $f^{(0)}(\vec{x}, W_0)$ represents the initial input to the network.

One can see that for larger networks these gradients have many repeated terms. The back propagation algorithm, introduced by Rumelhart *et al.* [20], takes advantage of this by first calculating the gradients of the layers closest to the final output, memoizing the result, and continuing backwards through the network.

Loss Functions

In our discussion of the back propagation algorithm, we introduce loss, a function $L : \mathbb{R}^n \to \mathbb{R}$ quantifying the amount of error or cost associated with the model's output. Some common loss functions include Mean Squared Error (MSE), Mean Absolute Error (MAE), and Cross Entropy (CE), defined by the equations 12, 13, and 14 respectively where P is the predicted value and T is the expected value.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(P_i - T_i)^2 \tag{12}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|(P_i - T_i)| \tag{13}$$

$$\text{CE} = -\sum_{i=1}^{n} T_i \log(P_i) \tag{14}$$

The most appropriate loss function depends on the goal of the network. MAE and MSE are suited to regression tasks whereas CE is more suited to classification tasks. In practice, loss functions are often custom made for the task at hand. The main constraint on the design of loss functions is that they be able to produce non-zero gradients, as the gradient descent algorithm depends on non-zero gradients to update weights.

Activation Functions

Activation functions control how a signal flows through a network's layers. Their primary purpose is to introduce a non-linear transformation. A network composed of only linear transformations cannot learn non-linear relationships because a series of successive linear transformations is equivalent to some single linear transformation. The activation functions used in a neural network can vary layer by layer. Certain activation functions are used more commonly in the hidden layers with others being more commonly used in final output layers. Here we present some of the commonly used activation functions.

The sigmoid function $\sigma(x)$ returns a value between 0 and 1. This function quickly approaches its limits making it well suited for binary classification. The main disadvantages to this function are its high computational cost and small gradients for input values far from zero.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{15}$$

The Rectified Linear Unit (ReLU), popularized by Nair *et al.* [22] and Glorot *et al.* [23], is currently the most widely used activation function [24]. Its main advantages are its sparse activations, low computational cost, and a non-vanishing gradient for positive inputs. Its main disadvantage is that for input values less than zero, the gradient will always equal zero which prevents a network's weights from being updated resulting in so called "dead neurons" [25].

$$\text{ReLU}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{16}$$

Multiple variations on the ReLU activation function have been proposed to address the dying neurons issue. Two such variations include the Exponential Linear Unit

(ELU) [26] and the Leaky Rectified Linear Unit (LReLU) [27]. Each of these variations change the function for values less than zero so that they have a non zero gradient. In each of these functions, α is a hyper-parameter (a non learned parameter governing the training procedure), controlling the degree to which these adjustments are applied.

$$\text{ELU}(x) = \begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{17}$$

$$\text{LReLU}(x) = \begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{18}$$

The softmax activation function is used for multi-class classification tasks in the final output layer. It is an approximation of the argmax function designed to have non-zero gradients. Softmax calculates a probability value between 0 and 1 for n classes such that they all sum to 1.

$$\text{Softmax}(\vec{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{19}$$

Convolutional Neural Network

In our explanation of a neural network's forward pass, we presented a fully connected network architecture in which every input value has a single associated weight parameter. In practice, this kind of architecture performs poorly in image processing tasks because the number of required parameters is $\Theta(n)$, where n is the number of image elements, incurring high computational complexity, and because the neuron connections do not take into account image elements' relative locations to each other. Convolutional neural networks (CNNs), introduced by Waibel *et al.*[28], use a different type of layer to address these issues.

For the purposes of explanation, we describe convolutional layers according to the 2D case, however, convolution can be applied in an arbitrary number of dimensions. Input to a 2D convolutional layer is a 2D image. The layer consists of a predefined number of kernels, (also known as filters), which are a type of small weight matrix that is discretely convolved over the input image to produce a transformed image. The number of transformed images produced by a convolutional layer is equal to the number of kernels in the layer.

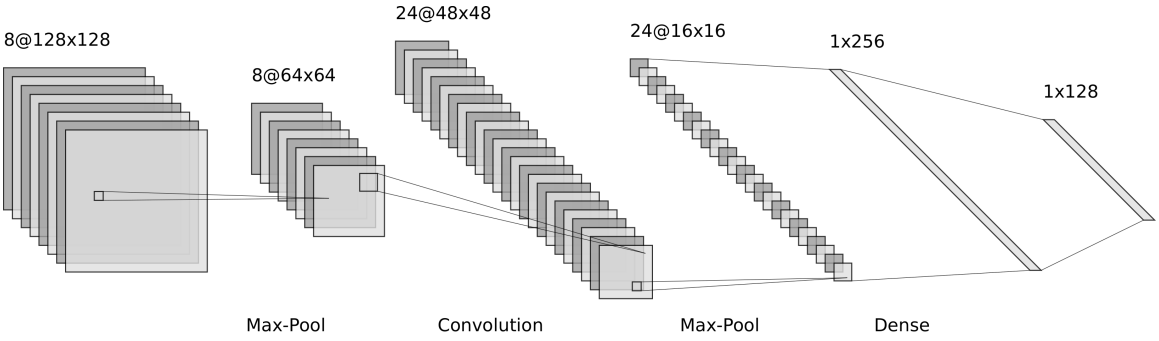


Figure 4: Diagram representing a simple example of a convolutional neural network design. Figure made using the online tool NN-SVG [19].

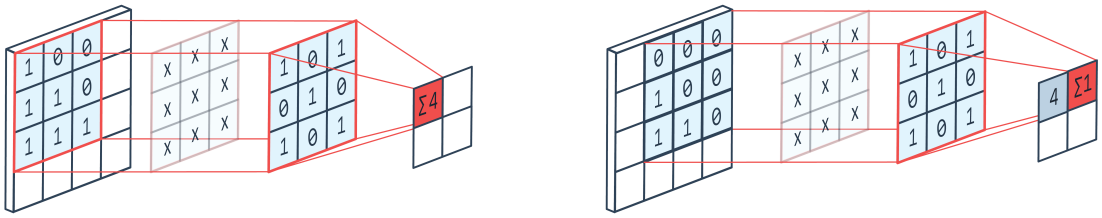


Figure 5: An illustration of a 2D convolution over a small image. Figure from [30].

Intuitively, one can think of discrete convolution as overlapping the kernel onto the input image, multiplying the weights in the kernel by the input image pixel values they overlap, and summing the result to yield a new transformed pixel. This process can be repeated for every possible way the kernel can overlap the input image to yield a fully transformed image. This process is illustrated in Figure 5.

Adapting the formulation of Damelin et al. [29], the 2D discrete convolution of kernel k over image x is given by:

$$\text{conv}(x,k)[i,j] = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} x[i,j] \cdot k[i-n, j-m] \tag{20}$$

For values of i and j where $x[i,j]$ or $k[i,j]$ are not defined, they are considered to be zero.

To save memory and computation time, convolutional NNs usually perform down-sampling between convolutional layers. Traditionally, max-pooling, which is a down-sampling operation yielding the maximum value of patches in the feature maps, has been used. However, more recently, strided convolutions have been used for down-sampling in CNNs with better results [31].

The type of layers composing a network need not be uniform. Many CNN's final layers are fully connected layers. Alternatively, the fully connected layers can be replaced by convolutional layers to form a fully convolutional neural network (FCNN) [32]. An FCNN's main advantage is that the convolution operation can be applied to inputs of arbitrary size. FCNNs are commonly used for semantic image segmentation.

Batching

In the subsection on back propagation and stochastic gradient descent, we discussed how, given a single sample input, a neural network estimates the loss gradient and updates the model's parameters. Each update to the model's weights is called a step. A step can be performed for the gradients calculated for each training datum, or for aggregated gradients from batches of data at a time. During training, a forward and back propagation pass through all of the training examples is called an epoch.

The batch size used during training is an important hyper-parameter because it controls the stochasticity of the training and influences training time. Larger batch sizes will estimate the gradient surface more consistently and train faster, but have been found to generalize less well due to their tendency to converge to sharp minima [33]. Furthermore, batch sizes are constrained by the amount of available GPU memory.

2.1.3 Overfitting and Solutions

In machine learning, there is a phenomenon where a model will perform better on the data on which it was trained rather than on new data. This difference in performance is called overfitting and is a result of the model relying on correlations present in the training data that do not generalize to the real data distribution.

In order to measure the real performance of a given model, a portion of the available data must be held out from the training set to be used for testing. In practice, data is usually split into three groups: training, validation, and testing. The training data is used to update the model parameters (i.e. weights), the validation data is used to determine the training hyper-parameters, and the testing data is used only for final evaluation.

Cross validation is a technique used to estimate how well a given model will perform in practice. It produces multiple different partitions of the data set into

train/test splits. Training and testing are performed separately on each split and the results are aggregated for the final evaluation. A commonly used type of cross validation is k-fold. In this method, the data is split evenly into k partitions. In one iteration, $k-1$ partitions are used for training and the remaining partition is used for testing. In total, k iterations are run, with a different partition held out for testing each iteration.

In the remainder of this subsection, we present commonly used techniques for mitigating the effect of overfitting.

Network Capacity and Early Stopping

In deep learning, network capacity is a network's ability to approximate the target function. This is related to the number of layers and parameters comprising the model. When a network's capacity is too great, it tends to memorize spurious correlations specific to the training data rather than generalizing and thus the effect of overfitting increases. Network capacity should be limited to be only as great as needed. There is no known way to determine this theoretically, thus it is typically determined empirically [34].

At a certain point in training, the observed validation loss score reaches a minimum. Meanwhile, the network continues to fit to the training data, the amount of overfitting increases, and the performance of the model diminishes. A solution to this is early stopping. Rather than training for a fixed number of epochs, in early stopping, a stopping condition is defined and training continues until this condition is reached. A common condition is when the tracked validation score has not increased significantly in the past set number of epochs.

Regularization

Regularization is an adjustment to the training protocol placing additional constraints on the model's weights and activations.

L1 and L2 regularization are two similar techniques that penalize the magnitude of a model's activations. In this way, the network is disincentivised from having large weights. Each L1 and L2 add an additional term to the loss function to impose this cost. These terms are given by equations 21 and 22 where λ is an additional hyperparameter controlling the proportional contribution to the loss score, m is the batch

size, and β_i is the ith value in an activation.

$$\text{L1} = \frac{\lambda}{m} \sum_{i=0}^{n} \|\beta_i\| \tag{21}$$

$$\text{L2} = \frac{\lambda}{m} \sum_{i=0}^{n} \|\beta_i\|^2 \tag{22}$$

Another technique regulating the weights of a model is dropout [35]. With dropout, during training time, individual units are temporarily randomly disabled with probability p. Since units are not always enabled, the model cannot depend on any single unit to function and must generalize.

Batch normalization is a technique that makes data normalization a part of the model architecture [36]. It addresses changes in the distribution of the layers' inputs during training. Batch normalization normalizes each scalar feature in the input individually across the batch. In doing so, it reduces 'internal covariate shift' and allows for higher learning rates [36].

The normalization functions are given by equations 23, 24, 25 where x_i denotes the ith value of x over a batch B of size m. γ and β are learned parameters to scale and shift the final transformed value (y_i) and ϵ is a small constant added to prevent division by zero errors.

$$\mu_B = \frac{1}{m} \sum_{i=1}^{m} x_i \tag{23}$$

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2 \tag{24}$$

$$y_i = \gamma(\frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}) + \beta \tag{25}$$

Data Augmentation

The performance of deep learning models is highly dependent on the amount and quality of available data [37]. The more data available, the more the model can generalize to the domain rather than to a few examples. Data augmentation is a technique of artificially increasing the amount of available data by supplementing the existing data with transformations of itself. Common types of data augmentation for

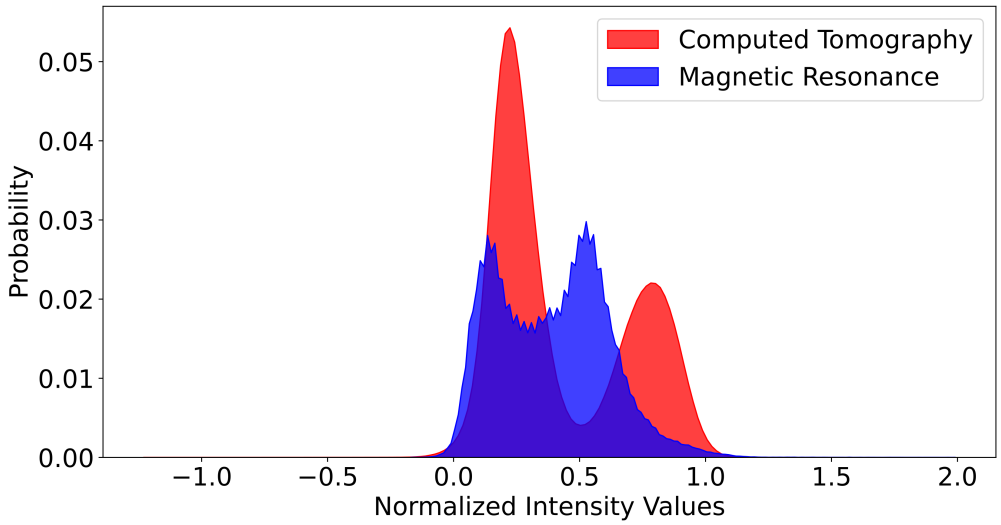


Figure 6: A histogram showing the distribution of normalized intensity values in each a CT image and an MR image from the Multi-Modality Whole Heart Segmentation challenge [7]. For each image, background voxels were excluded from the shown distribution and intensity values were linearly rescaled such that the 0.5 and 99.5 percentile values are transformed to 0 and 1 respectively.

images are translation, rotation, reflection, scale shifts, intensity shifts, and addition of noise (e.g. Gaussian noise). Each of these augmentations serves to make the model invariant to analogous differences in new data.

2.2 Biomedical Images

A computed tomography (CT) or magnetic resonance (MR) scan results in a 3D image composed of discrete volumetric elements called voxels. Each voxel is associated with a scalar value which depends on the modality of image acquisition. The purpose of this section is to describe the physical properties represented by these scalar values in CT and MR images and to highlight that they are inherently different. Structures appearing in one modality may not appear or may appear differently in another modality. This difference in distribution is illustrated in Figure 6.

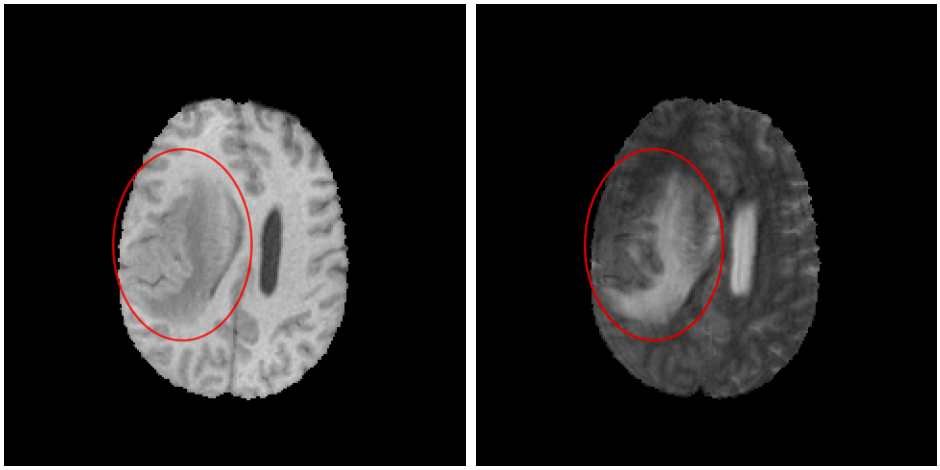
2.2.1 Computed Tomography

Computed Tomography (CT) images are acquired using X-ray radiation. As these rays are projected through the subject's body, their intensities are attenuated by interactions with matter including absorption and scattering. The probability of these interactions is directly proportional to the electron density of the tissues along the rays' paths. Thus electron density is the fundamental property represented by the resulting images. By taking these projective radiographic measurements from many angles around the subject, the signals can be combined to produce a single cross-sectional (slice) image. Combining multiple such slices acquired at different positions produces a volume where each voxel represents the measured local attenuation. This property is typically reported using Hounsfield units (HU), a linear transformation of the attenuation measurement such that the attenuation of distilled water is 0 and the attenuation of air is 1000.

$$HU = 1000\frac{\mu - \mu_{water}}{\mu_{water} - \mu_{air}} \tag{26}$$

2.2.2 Magnetic Resonance

Magnetic Resonance (MR) images are acquired using nuclear magnetic resonance (NMR). Nuclei with either an odd atomic number or odd mass number have spin and are NMR-active. In practice, when imaging human anatomy, this is overwhelmingly hydrogen nuclei due to the high proportion of water present. When placed in a strong magnetic field, the magnetic moments of these nuclei become aligned with the field. A second magnetic field is applied perpendicularly to this field in pulses at radio frequency (RF). These pulses cause the magnetic moments to precess around the first field. This precession causes a change in magnetic field which induces a voltage in detection coils at the frequency of precession with an amplitude proportional to the proton density. This is the signal detected by MR imaging, but not necessarily the property measured. The precession of these magnetic moments can be viewed as having two components, a transverse component and a longitudinal component. After the RF pulses end, the precession decays over time. The time to decay in each component is tissue dependant. A T1 MR image measures the time for longitudinal decay. A T2 MR image measures the time for transverse decay. These acquisition modes strongly affect the distribution of values in the resulting images. Figure 7



(a) T1 MR Image (b) T2 MR Image

Figure 7: Two registered MR images from one subject captured with different acquisition modes. Encircled is a glioma and oedema. Made using data from the 2020 Brain Tumor Segmentation (BraTS) challenge [13].

shows an example of the difference between T1 and T2 acquisition modes.

2.2.3 Common Complications

In addition to measuring different physical properties, there are a number of other potential factors influencing the images' appearances. Both MR and CT imaging capture images in slices and construct a volume by stacking these slices together. Usually the resolution of each slice is isotropic, but the spacing between slices is chosen independently and typically results in anisotropic resolution in one direction. Images may be captured with or without contrast agents which further increases the variability of image appearance. Both modalities are subject to standard imaging complications such as variable resolution, level of quality, signal-to-noise ratio, and contrast-to-noise ratio.

All of these factors must be taken into consideration when designing a robust image processing model.

2.3 Biomedical Image Segmentation

Segmentation in biomedical images is an important step in many clinical applications as it transforms the images into a less complex form more tractable to standard

computer algorithms. In this section, we discuss aspects of deep learning specific to biomedical image segmentation.

2.3.1 Specific Difficulties

Image segmentation in the biomedical domain has some specific difficulties to address in addition to the common segmentation challenges.

Data is sparsely available. Both CT and MR imaging require large, expensive machinery to acquire. The images acquired are a patient's confidential medical information and access is thus restricted by law and ethics. Of the images available, annotation for supervised learning can only be done manually by highly skilled medical professionals. Such annotation is very time-consuming with the organizers of one data set (for the Multi-Modality Whole Heart Segmentation challenge) reporting annotation times of six to ten hours per subject [38].

Due to inter-rater variability, the ground-truth labels used for training have limited consistency. One study analysing the inter-rater variability in delineating lesions in MRI found that there was a large amount of variation in volumes recorded by the raters, raters tended to under predict volume in one modality compared to another, and, while raters agreed on lesion centres, they had 'considerable disagreement' for the lesion borders [4].

The volumetric images captured contain more spatial information than their 2-dimensional counterparts, but this comes at the cost of highly increased computational and memory requirements in their processing. In practice, the complexity of models and the size of the images that they can analyse is highly constrained by the amount of available GPU memory. There have been numerous approaches addressing this difficulty including, but not limited to, 2.5D networks [39], running inferences on patches [40, 41, 42], and combining high resolution detail with low resolution detail in different paths [15, 43].

2.3.2 U-Net

In 2015, to yield precise segmentations with relatively few training images, Ronneberger *et al.* presented a neural network architecture for biomedical image segmentation called U-Net [15]. Since then, it has become highly prevalent in the domain,

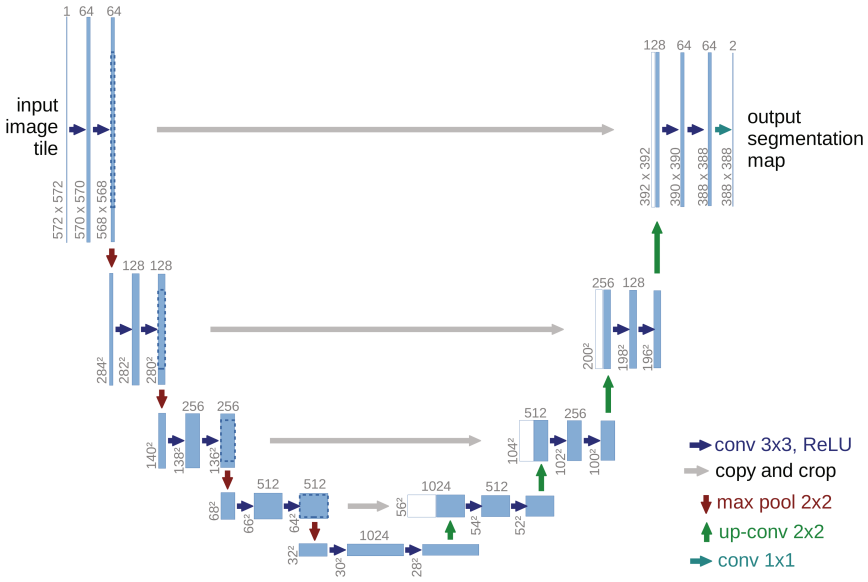


Figure 8: U-Net architecture. Blue boxes represent feature maps. The number of channels is given by the numbers above the boxes. The feature map sizes are given by the numbers to the left of the boxes. Operations are denoted by the coloured arrows as specified in the legend. Figure from the original paper by Ronneberger *et al.* [15].

inspiring many variants. The U-Net architecture, shown in Figure 8, is a type of fully convolutional architecture featuring a contracting (encoding) path in which the feature maps are successively reduced in size, followed by an expanding (decoding) path in which the feature maps are successively increased in size. The two paths are roughly symmetrical in nature and features from the contracting path are concatenated with corresponding feature maps in the expanding path. The main 'U' path allows the network to save memory and computation time by learning low resolution representations of features, while the concatenation paths allow it to retain the fine detail from the early stages.

2.3.3 Evaluation

Choosing an appropriate metric is critical to properly evaluate the performance of a model against a testing data set. For example, in the BraTS 2020 challenge data set [13], the 369 segmentation masks are on average 98.9% background class. In this extremely unbalanced example, when using categorical accuracy as a metric, a model could trivially achieve 98.9% by predicting only the background class.

The two most commonly used metrics in biomedical segmentation competitions are the Dice Similarity Coefficient and Hausdorff Distance [44].

The Dice Similarity Coefficient (DSC or Dice score), given by equation 27, measures the amount of overlap between two binary samples. In this context, it is used to compare the voxels from the ground truth, T, to the voxels in the prediction, P. It can be easily extended for use with multiple classes by calculating the mean of per-class Dice score.

$$\text{DSC} = \frac{2|T \cap P|}{|T| + |P|} \tag{27}$$

Hausdorff distance, given by equations 28 and 29, as presented by [45], measures the greatest smallest distance between two binary samples. That is, for each voxel, t, in the ground truth, T, there is a minimum distance to some voxel, p, in the prediction, P and likewise with voxels from P to T. Hausdorff distance is the maximum of these distances.

There exists variants of Hausdorff distance that, instead of using the maximum of the distances, use a high percentile (such as 95th) of the minimum distances. These variants make the metric less sensitive to outliers.

$$D_{\text{Hausdorff}}(T, P) = \max(h(T, P), h(P, T)) \tag{28}$$

$$h(T, P) = \max_{t \in T} \min_{p \in P} \|t - p\| \tag{29}$$

In this chapter, we gave the background necessary to understand the application of deep learning to image segmentation in biomedical imagery. In the next chapter, we explore the use of different multi-stream neural network structures to learn segmentation from unpaired CT and MR images.

Chapter 3

Multistream Models for Unpaired Learning

3.1 Introduction

Deep learning is increasingly being applied in the medical domain for classification, detection and segmentation. Semantic segmentation of medical images is an important early step in developing clinical applications such as automatic landmark detection for pre-procedural planning of trans-catheter aortic valve implantation [46]. Manual annotation of medical images is an extremely tedious and time consuming process, for example, the manual whole heart segmentation in CT and MRI takes between 6 to 10 hours per subject [38]. Furthermore, even when manual segmentation is done by highly skilled medical professionals, it suffers from both inter-rater and intra-rater variability [4]. Automated segmentation methods have thus been an active area of research, however, large variability in patient anatomy makes automated segmentation methods challenging. In recent years, deep learning has become increasingly popular as it has shown to be an effective technique to perform medical semantic segmentation [7].

The performance of deep learning models is highly dependant on the availability of training data. A recent paper by Valindria *et al.* [47] explored the possibility of learning a shared representation for the segmentation of major organs in unpaired CT and MRI abdominal images using four different multi-stream fully convolutional network architectures. Their results showed that a certain X-shaped dual-stream

model achieved top results over all classes in their data when compared to their single-stream baseline and three other dual-stream candidate architectures. Given that the reproducibility of deep learning techniques can be affected by small changes in environment and hyper-parameters [48], in this paper, we aimed to determine if the performance gains achieved by the representation sharing scheme proposed by Valindria *et al.*[47] could be achieved under similar conditions.

3.2 Related Works

In addition to the work by Valindria *et al.*, there are a number of different approaches to unpaired multi-modal learning in the medical imaging literature. In this section, we describe some of these approaches.

Moeskops *et al.* show that a single network trained on three separate modalities can achieve performance equivalent to three separately trained networks [49]. Zhang *et al.* show performance gains using GAN synthesized images for unsupervised learning with different modalities, artificially translating images from one modality to another [50]. Dou *et al.* use a largely single-stream architecture with an additional loss term inspired by knowledge distillation [51] that minimizes the divergence of prediction distributions between modalities [52]. They reported overall mean Dice scores slightly higher than other multi-modal learning schemes. Finally, Li *et al.* use a combination of both a GAN image translation module and an additional knowledge distillation based loss term in a single network [53]. They reported top Dice scores in every class compared to other multi-modal learning schemes including the work by Zhang *et al.* mentioned earlier.

Interested readers can look to a review of deep learning for cardiac image segmentation by Chen *et al.* [54] and of deep learning solutions to imperfect data sets in medical segmentation by Tajbakhsh *et al.* [55].

3.2.1 Data Set

The MM-WHS data set [7] consists of 60 cardiac CT and 60 cardiac MRI 3D images. The CT and MRI images are *unpaired*, i.e. they are not acquired from the same patients. All images come from real clinical environments and have varying image quality, resolution, and slice thickness. Each image is paired with a ground truth,

(a) CT Image (b) CT Ground Truth (c) MR Image

Figure 9: Example slices from pre-processed images in the MM-WHS data set.

a manual segmentation into seven cardiac substructures: the left and right ventricle blood cavity (LVC & RVC), the left and right atrium blood cavity (LAC & RAC), the left ventricle myocardium (LVM), the ascending aorta (Asc. A), the pulmonary aorta (Pulm. A), and the background (BG).

This data set was released for a 2017 biomedical image segmentation challenge seeking to find and validate the current state-of-the-art algorithms. For this challenge, there were twelve participating teams, nine of which provided results for both CT and MR images. The top performing algorithms (by the Dice Similarity Coefficient) were different for each CT and MR images.

The top performing solution for CT images was a two-stage CNN-based architecture where the first stage operates on a low resolution of the image to localize the region of interest and the second stage operates on the full resolution version of the image cropped around the region of interest to produce the final segmentation [56].

The top performing solution for MR images was from a team from the University of Bern, Switzerland. A description of the method is given by the MMWHS challenge organizers [7]. It features a fully convolutional DenseNet encoder-decoder network. Multi-scale contextual information is included in the encoding path to enhance feature learning.

3.2.2 Valindria's Work

The work by Valindria *et al.* [47] is based on work by Kuga *et al.* which proposed multi-modal encoder-decoder networks with shared skip connections to more fully exploit inter-modal commonalities for multiple related scene recognition tasks [57].

Valindria *et al.* applied this technique to the problem of unpaired CT and MR image segmentation. They sought to determine which parts of dual stream networks are best merged or split. To do so, they used a single-stream FCNN-based architecture for their baseline model and designed four different dual stream model configurations from this with different parts of the streams being merged for each configuration.

For data, they used two different data sets of 3D abdominal images, one of CT images and one of T2-weighted MR images. This data was pre-processed to have isometric 2mm voxel spacing, CT voxel intensities were normalized, and images were cropped to have similar field of view, covering only the regions to be segmented. During training, they apply Gaussian noise and intensity shifts for data augmentation and use weighted class sampling to compensate for class imbalance.

Training was performed on mini-batches of 16 64^3-sized patches for 10,000 steps using Dice loss. For multi-modal training, mini-batches were selected from the separate modalities in an alternating manner.

Their findings were that one particular dual-stream configuration, where the streams were briefly merged between the encoding and decoding stages, had the best overall segmentation performance on every class when compared to both the baseline model and the other dual-stream configurations.

Their code for their experiment is publicly available online at:

`https://github.com/vanya2v/Multi-modal-learning`

3.3 Methodology

We evaluated the four dual-stream configurations proposed by Valindria *et al.* [47] on the semantic segmentation of MRI and CT cardiac images from the MICCAI 2017 Multi-Modality Whole Heart Segmentation Challenge (MM-WHS) [7]. Although we borrow methodological approaches from Valindria's work, in particular the four dual-stream configuration schemes, we differ in many choices including using a different baseline architecture (U-Net vs. FCNN), a different overall training setup with our own hyper-parameters and data augmentations, and we train and evaluate using a different data set. In this way, we aim to determine if the practice of dual-stream learning is sufficiently robust to perform well generally rather than only in one specific case. Our implementation is available at:

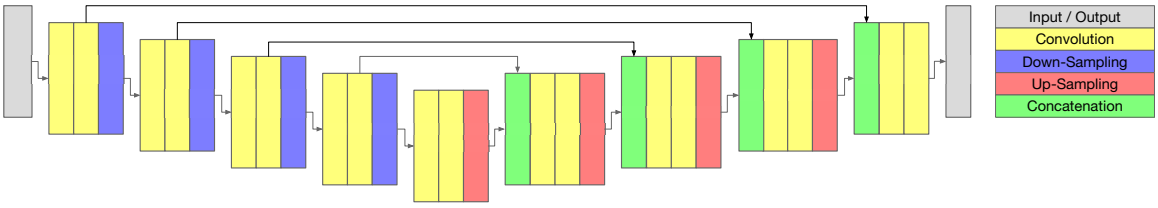


Figure 10: Architecture for Single-Stream U-Net Model

`https://github.com/AppliedPerceptionLab/Multi-ModalitySegmentation`

3.3.1 Data Set Pre-Processing

Prior to training, the images and ground truths were subject to a number of pre-processing steps. Firstly, all images were re-sampled to isometric 1 mm resolution (images were re-sampled using linear interpolation and ground truths using nearest-neighbours). All images and labels were rotated such that the XY axes represented the axial plane. Some labels were truncated in scope to match the data organizers' definitions. Specifically, labels for the pulmonary artery were made to include only the volume between the pulmonary valve and the bifurcation point and labels for the ascending aorta were made to include only the volume between the aortic valve to the superior level of the atria. Lastly, each image's intensities were Min Max normalized into the range [-1,1].

3.3.2 Model Architectures

The architecture for the single-stream (SS) model (Fig. 10) is a fully convolutional network based on U-Net [15] and its 3D counterpart V-Net [58]. It consists of four down-sampling stages followed by four up-sampling stages and one final softmax stage yielding the final output, a vector of class probabilities for each voxel. Each down stage consists of two 3D convolutional layers followed by a down-sampling layer. Down-sampling is performed by 2-strided convolutional layers, which perform well compared to pooling layers [31]. Each up-stage begins by concatenating the output from the previous stage with the pre-down-sampled output from the down-stage of equivalent size. This is followed by two convolutional layers and an up-sampling layer performed by 2-strided transposed convolution.

The number of convolutional kernels in each layer begins at 16 and doubles with

each down-stage, up to 256, then halves with each up-stage. The final output layer uses 8 kernels, one for each output class. Every layer uses 5x5x5 sized convolution kernels and the ELU activation function [26], except for the final layer which uses 1x1x1 sized convolution kernels and softmax activation.

The hyper-parameter selection for the single-stream baseline model is the result of a series of empirical tests validating its performance on the same data set. The presented selection demonstrated the highest validation performance of all of the tested permutations.

Each of the four dual-stream models are pairs of models exactly identical to the single-stream model, but with a subset of their layers shared. The different configurations of layer sharing follow the proposed configurations by Valindria, *et al.* [47] as described below.

Dual-Stream Model Version 1 (DV1): This architecture features separate input and down-sampling layers for each CT and MRI, but a shared up-sampling and final output (Fig.11).

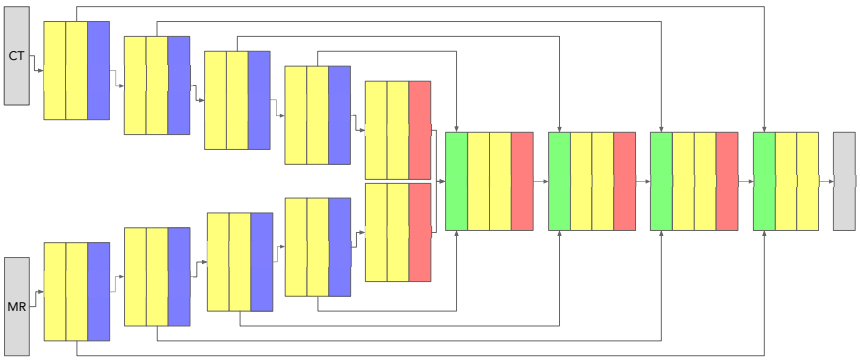


Figure 11: Architecture for Dual-Stream U-Net Model Version 1

Dual-Stream Model Version 2 (DV2): This architecture features separate input and one down-sampling stage for each CT and MRI, but the rest of the architecture is shared (Fig.12).

Dual-Stream Model Version 3 (DV3): This architecture features a shared input and down-sampling path, but separated up-sampling and outputs (Fig.13).

Dual-Stream Model Version 4 (DV4): This architecture features a single shared stage at the end of the down-sampling phase, but has otherwise individual input and output streams (Fig.14).

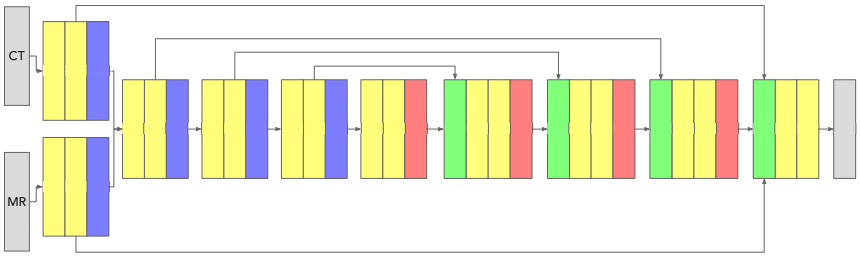


Figure 12: Architecture for Dual-Stream U-Net Model Version 2

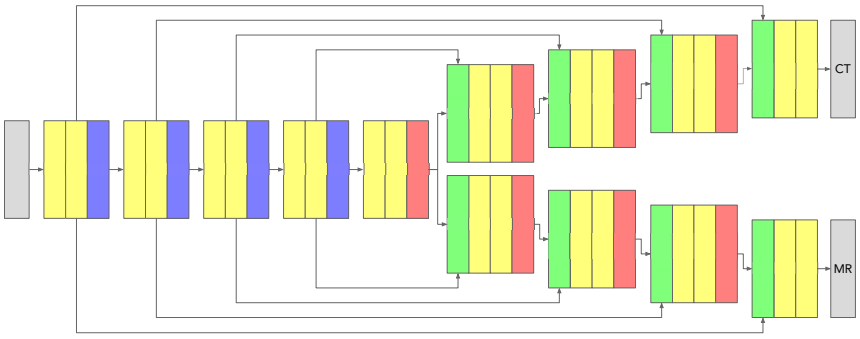


Figure 13: Architecture for Dual-Stream U-Net Model Version 3

3.3.3 Loss and Optimization

For training, we use a modified version of the weighted multi-class Dice loss score used in [59]. We chose to use Dice loss because the Dice Similarity Coefficient is the primary evaluation metric used in biomedical image segmentation and, in this way, we optimize for it as directly as possible. Weighting was used to mitigate the deleterious effects of high class imbalance. For each class, a Dice coefficient is calculated by:

$$\text{Coef}_c = \frac{2(\sum_{i=1}^{N} t_{ci} p_{ci}) + \epsilon}{\sum_{i=1}^{N} t_{ci} + \sum_{i=1}^{N} p_{ci} + \epsilon} \tag{30}$$

Where $p_{ci} \in [0, 1]$ represents the predicted probability of class c at voxel i, $t_{ci} \in \{0, 1\}$ represents the truth of class c at voxel i, N represents the number of voxels in the ground-truth image, and $\epsilon = 10^{-7}$ is used to correct for division by zero.

The final Loss score is then calculated as:

$$\text{Loss} = \frac{1}{\sum_{w_c \in W} w_c} \sum_{c \in C} (w_c(1 - \text{Coef}_c)) \tag{31}$$

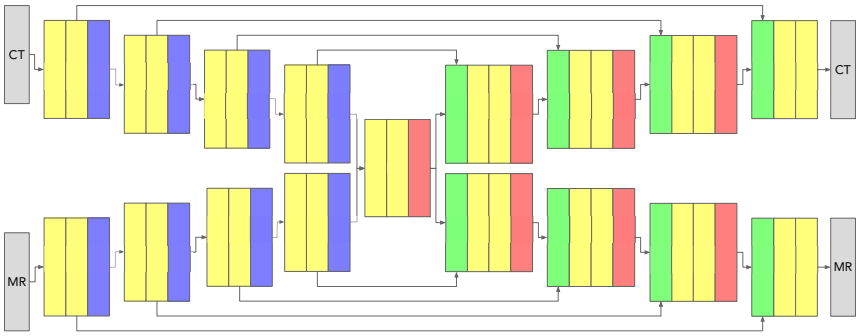


Figure 14: Architecture for Dual-Stream U-Net Model Version 4

Where the weights W are pre-calculated across the entire training data set as:

$$W_c = 1 - \frac{L_c}{N} \tag{32}$$

Where N represents the total number of voxels and L_c represents the total number of voxels for label c.

The models are trained using the Adam optimizer [60] with a learning rate of 10^{-4}, $\beta_1 = 0.9$, and $\beta_2 = 0.999$.

3.3.4 Data Augmentation

During training, we randomly apply a number of data augmentations to improve the generalizability of the resulting model including: (1) addition of Gaussian noise with $\mu = 0.0$ and $\sigma = 0.2$ (p=0.3); (2) random shift in image intensity between -20% to 20% (p=0.5); (3) image quality adjustment with Gaussian smoothing (p=0.3); (4) volume rotation between -15° to 15° around the vertical axis (p=0.5); and (5) horizontal flipping of the volume's transverse plane (p=0.5). The data augmentations were chosen for being the most commonly used augmentations for similar tasks and their specific parameters were determined ad hoc.

3.3.5 Hardware and Software Configuration

All networks were implemented in Keras [61] with TensorFlow 2.0.0 [62] backend. Models were trained and run on a NVIDIA GeForce RTX 2080 TI GPU with 11 GB VRAM.

3.3.6 Training Procedure

The fully convolutional structure of the models allow for variable input image sizes, however, due to memory constraints, there is a practical limit on overall input size. To overcome this limitation, during training, the input images are split into 64^3 sized volumetric patches and training is performed with batches of 20 patches. Patches where the ground truth contains only background class labels are filtered out.

The data set was split volume-wise into training, validation, and testing sets with an 80%:10%:10% ratio.

Two instances of the single-stream model were trained, one for each the CT and MR modalities. Each instance was trained for 125 epochs where each epoch was a complete run through the training data, about 60 batches. Each dual-stream model was trained with alternating batches of CT and MRI data. Each dual-stream instance's epochs were a combined 120 batches in length.

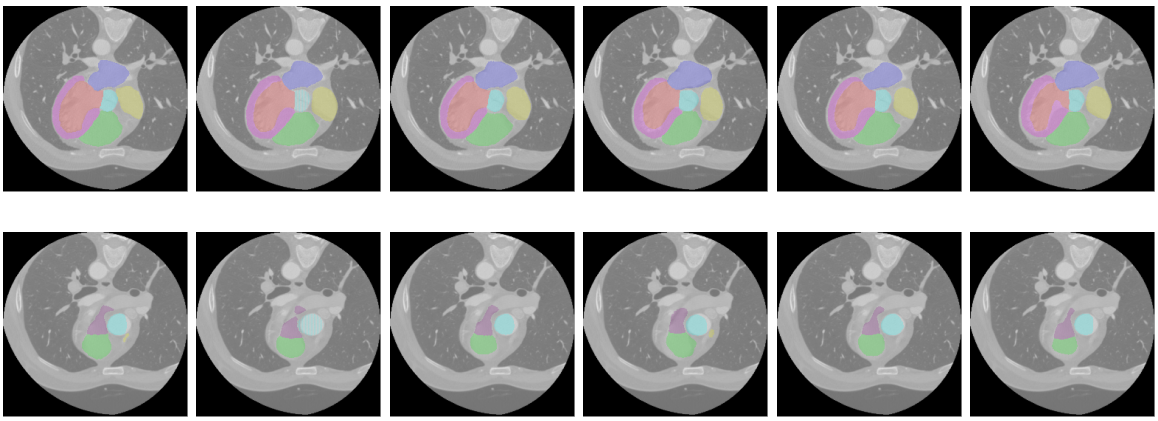
At the end of each epoch, the model was evaluated against the validation data using the mean of each class' Dice similarity coefficient calculated using equation 33. For each model architecture, the epoch with the highest mean validation Dice similarity was selected as the candidate for testing.

$$\text{Coef}_c = \frac{2|T_c \cap P_c| + \epsilon}{|T_c| + |P_c| + \epsilon} \tag{33}$$

Testing was performed by calculating the Dice coefficients for each class over the entirety of each image in the testing set. For the CT images, inference was done on each whole image volume. MR images in the data set were larger on average so inference was performed on 256x256x128 sized grid-based patches. Patch predictions were then recombined into whole image predictions before evaluation.

3.4 Results and Discussion

The mean Dice scores calculated from testing each model architecture are presented in Table 1 and Table 2 and qualitative examples of their testing segmentations are shown in Figures 15 and 16. We found no model performed best universally. The model with the best performance on CT images was the Dual-Stream Model Version 3 (DV3) with a mean Dice score of 0.881 which was an increase of 0.052 over the single-stream baseline on CT images (SSCT) and had the second best performance



(a) Truth (b) SSCT (c) DV1 (d) DV2 (e) DV3 (f) DV4

Figure 15: An example of segmentation results on a CT image from the testing set by each architecture. The predicted labels for class are labeled as red (LVC), green (RVC), blue (LAC), yellow (RAC), fuchsia (LVM), cyan (Asc. A), and purple (Pulm. A).

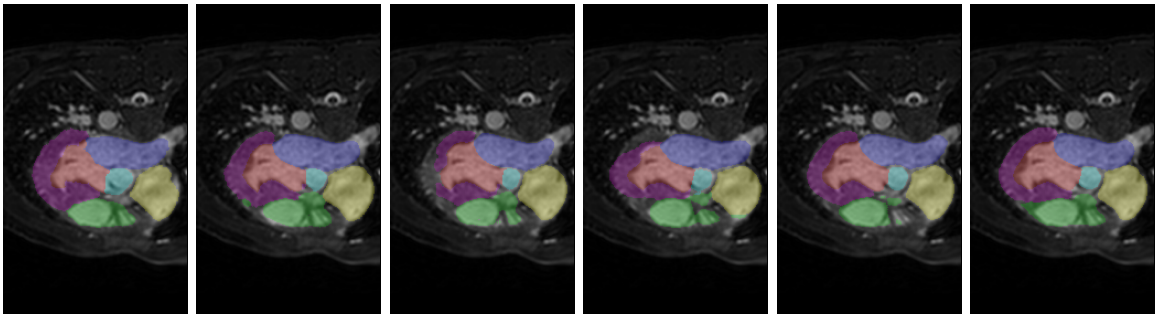
on MR images. The model with the best performance on MR images was the Dual-Stream Model Version 4 (DV4) with a mean Dice score of 0.793, which was an increase of 0.132 over the single-stream baseline on MR images (SSMR), however, DV4 also performed poorly on CT images, under-performing the baseline.

Table 1: CT class and mean Dice scores for each model architecture. Bolded numbers identify the best results. Mean is computed over non-background classes.

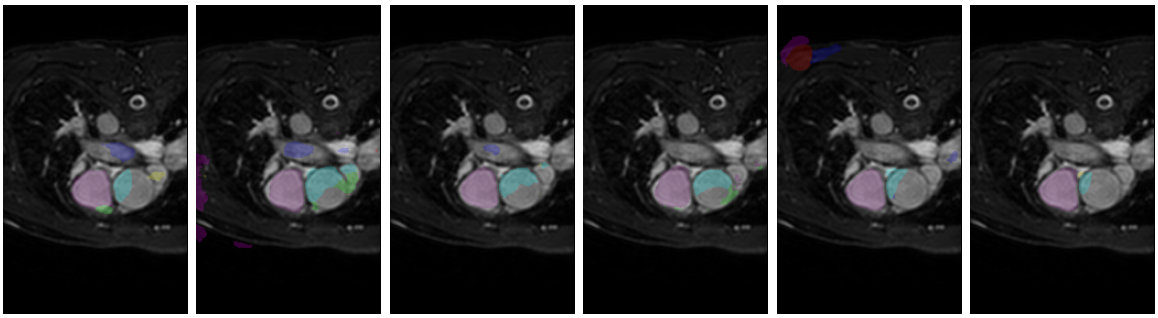
Name	**BG**	**LVC**	**RVC**	**LAC**	**RAC**	**LVM**	**Asc. A**	**Pulm. A**	**Mean**
SSCT	0.993	0.901	0.871	0.914	0.874	0.828	0.607	0.807	0.829
DV1	0.994	0.858	0.875	**0.921**	**0.881**	0.779	0.923	0.811	0.864
DV2	0.989	0.803	0.768	0.825	0.735	0.733	0.772	0.753	0.770
DV3	**0.995**	**0.905**	**0.887**	0.911	0.872	**0.849**	**0.929**	**0.818**	**0.881**
DV4	0.991	0.867	0.813	0.826	0.755	0.729	0.789	0.649	0.776

Neither of the single-stream baseline models ever achieved the top performance for any class. For MR, both DV3 and DV4 achieved better than baseline performance over every class, but for CT there was no model that universally outperformed the baseline model.

Segmentation performance for MR images was, for the most part, worse than for CT images, which is consistent with findings by Zhuang *et al.* [7], however the gains in MR segmentation performance were much larger than the gains in CT segmentation performance. All dual-stream models achieved greater than baseline performance on



(a) Truth (b) SSMR (c) DV1 (d) DV2 (e) DV3 (f) DV4



(g) Truth (h) SSMR (i) DV1 (j) DV2 (k) DV3 (l) DV4

Figure 16: An example of segmentation results on an MR image from the testing set by each architecture. The predicted labels for class are labeled as red (LVC), green (RVC), blue (LAC), yellow (RAC), fuchsia (LVM), cyan (Asc. A), and purple (Pulm. A).

MR images. Only DV1 and DV3 achieved higher than baseline performance on CT images.

In Table 3, we present the difference between the per class Dice scores from the single-stream model and the average of all of the dual-stream models for each modality. For CT, the dual-stream models performed worse on most classes compared to the single-stream model with the exception of one class that performed significantly better, and the background class which performed marginally better. For MR, the dual-stream models performed better on most classes with only one exception that preformed marginally worse.

For CT segmentation, the organizers of the MMWHS data set reported that, out of the ten evaluated algorithms, the highest Dice score achieved was 0.908 and the average Dice score achieved was 0.859. For MR segmentation, the highest Dice score achieved was 0.874 and the average Dice score achieved was 0.844. Taking these numbers at face value would indicate that our best models were comparable to the

Table 2: MR class and mean Dice scores for each model architecture. Bolded numbers identify the best results. Mean is computed over non-background classes.

Name	BG	LVC	RVC	LAC	RAC	LVM	Asc. A	Pulm. A	Mean
SSMR	0.982	0.833	0.801	0.738	0.454	0.600	0.717	0.486	0.661
DV1	**0.994**	0.890	0.767	0.786	**0.818**	0.721	0.715	0.537	0.748
DV2	0.993	**0.915**	0.779	0.778	0.772	0.692	0.680	0.490	0.729
DV3	0.991	0.867	0.810	0.827	0.756	0.726	0.736	0.647	0.767
DV4	0.993	0.873	**0.814**	**0.841**	0.806	**0.750**	**0.792**	**0.676**	**0.793**

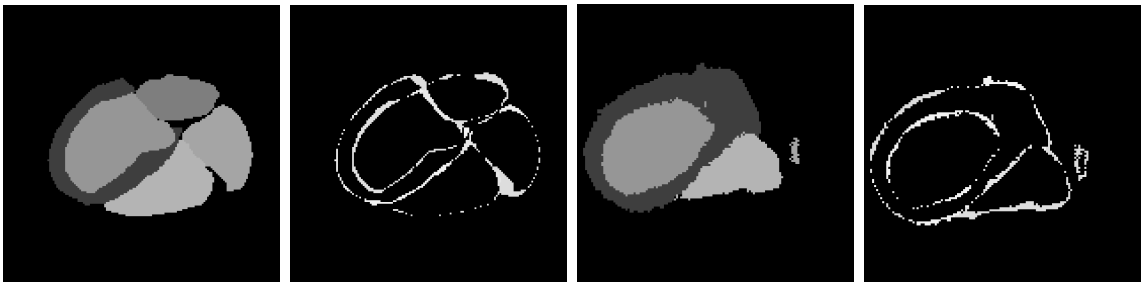
Table 3: Average change in per-class Dice scores across all dual-stream models. Mean is computed over non-background classes. All values are rounded to two significant figures. Positive values are in bold.

Modality	BG	LVC	RVC	LAC	RAC	LVM	Asc. A	Pulm. A	Mean
CT	**0.00075**	-0.043	-0.035	-0.043	-0.063	-0.060	**0.25**	-0.049	-0.0062
MR	**0.011**	**0.053**	-0.0085	**0.070**	**0.33**	**0.12**	**0.014**	**0.10**	**0.098**

top algorithms for CT segmentation and below average for MR segmentation. That being said, this experiment was designed to evaluate multi-stream models against an analogous single stream model rather than to compare against the state-of-the-art. In doing so, we employed different pre-processing to the image data and a different train/test image split. For this reason, these numbers are not directly comparable.

Qualitative inspections of the error produced by any of the networks show that, for the most part, erroneous voxel classifications do not have arbitrary spatial distribution. The networks almost always correctly determine the general locations of the cardiac structures and disagree on the precise structure boundaries by a few voxels. This pattern is illustrated in Figure 17.

In a follow-up analysis of the different network architectures, we captured the intermediate activations from each of the 32 layers for each network architecture for each modality. For the dual-stream architectures, the separate modality streams were treated as separate networks that happened to share certain layers. For each layer we counted the number of dead activations, activations where every voxel intensity was -1, i.e. the minimum possible value of the ELU activation function. The number of dead activations in each layer for each architecture and modality are presented in Figure 18. From this we make two main observations: (1) The number of dead activations tended to be highest at the bottom of the networks, i.e closest to the most down-sampled activations and the layers with the most kernels; and (2) For each



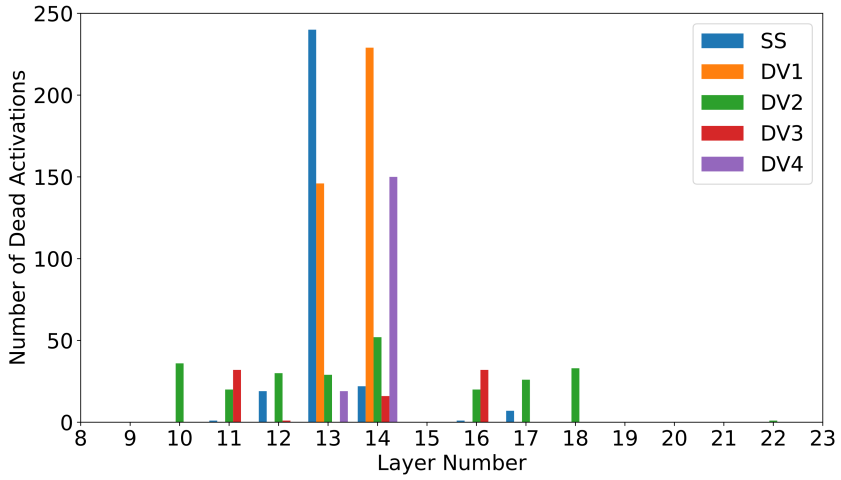
(a) CT Truth (b) CT Error (c) MR Truth (d) MR Error

Figure 17: An example of typical spatial distributions of erroneous voxel classifications by the top performing multi-stream architectures on each CT and MR modalities. On the left are the ground-truth segmentation masks and on the right are the locations of misclassified voxels for the same image. For CT, inference was performed by architecture DV3 and for MR, inference was performed by architecture DV4.

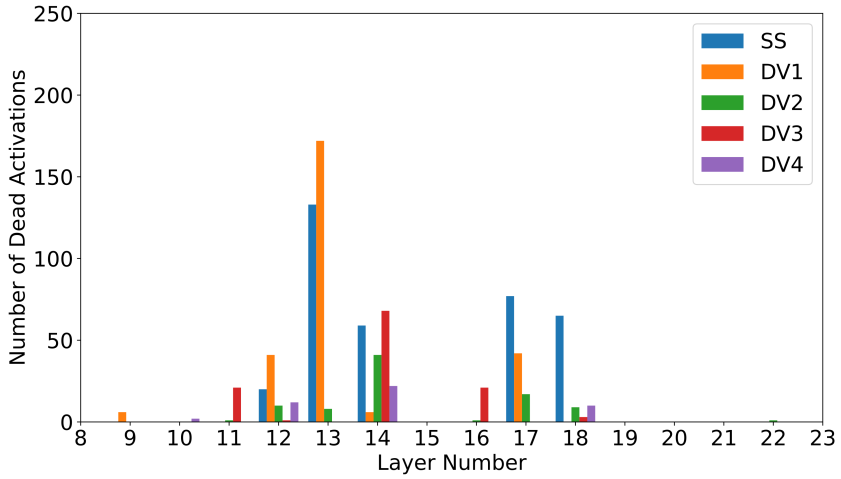
CT and MR, the architecture with the best testing performance, i.e. DV3 and DV4 respectively, was the architecture with the fewest overall number of dead activations.

3.5 Conclusion

In the work by Valindria *et al.* [47], their X-shaped architecture (equivalent to DV4) had universally better performance compared to all other tested models on all classes. Our results showed no top performing model, however, two models showed a strong improvement over baseline for CT images and all dual-stream models showed even stronger improvement over baseline for MRI images. On average, dual-stream models performed worse for CT images and better for MRI images. This suggests that representation sharing schemes like these dual-stream architectures may be more beneficial for some modalities than others. For CT, specific dual-stream architectures may improve performance compared to traditional architectures. Since the publishing of the Valindria *et al.* paper, numerous others have cited it assuming that the X-shaped architecture performs best. Here we show that this may not always be the case. This may be due to different data sets with different acquisition properties, parameters, or stochastic effects, but emphasizes the importance of testing various architectures and realizing no one solution might be better universally.



(a) CT Dead Activations



(b) MR Dead Activations

Figure 18: Number of dead activations in each layer for each multi-modal architecture on each modality. Layers not shown in chart had no dead activations.

Chapter 4

Deep Learning Solutions for Brain Tumour Segmentation

Segmentation of biomedical images is an important step in many clinical applications. It is a transformation of complex signals into discrete semantic structures making automated tasks for visualisation and quantification more tractable. Brain tumour segmentation, specifically, is used for surgical planning to accurately determine tumour borders. A better segmentation for this task may enable maximal tumour resection while preserving maximal healthy brain tissue. Studies have shown that complete removal of a brain tumour, i.e. gross total resection, results in longer overall survival and longer progression-free survival for patients in the short term [63, 64, 65].

Manual image segmentation is a painstaking and time-consuming process that must be performed by clinical experts. Additionally, studies have found there to be both high inter- and intra-rater variability in expert manual segmentation [4]. For this reason, automated segmentation methods have been a highly active area of research.

In the following chapter, we describe a custom deep learning solution to brain tumour segmentation and evaluate it against existing publicly available state-of-the-art solutions. Our aim is to achieve better performance through more careful customization around the constraints of data and computing resources than can be achieved through pre-made solutions.

4.1 Pre-made Solutions

Two state-of-the-art deep learning methods were used for comparison to our work: Deep Medic and No New Net. These were chosen as benchmark networks because they have been previously validated on publicly available data sets with strong reported results and have easy-to-use, publicly available code.

4.1.1 Deep Medic

Deep Medic, which was proposed by Kamnitsas *et al.*, is a multi-scale 3D convolutional neural network for segmentation of 3D biomedical images [43]. It achieved top ranking performance on both the ISLES 2015 challenge and the BraTS 2015 challenge [66]. Deep Medic seeks to overcome the computational complexity of large 3D images by using a dual pathway architecture where one pathway operates on low resolution patches and the other operates on a high resolution subset of the given patch. Thus one pathway learns fine detail features while the other learns the lower resolution context. In this way, it is able to operate on arbitrarily large images. The Deep Medic architecture is shown in Figure 19.

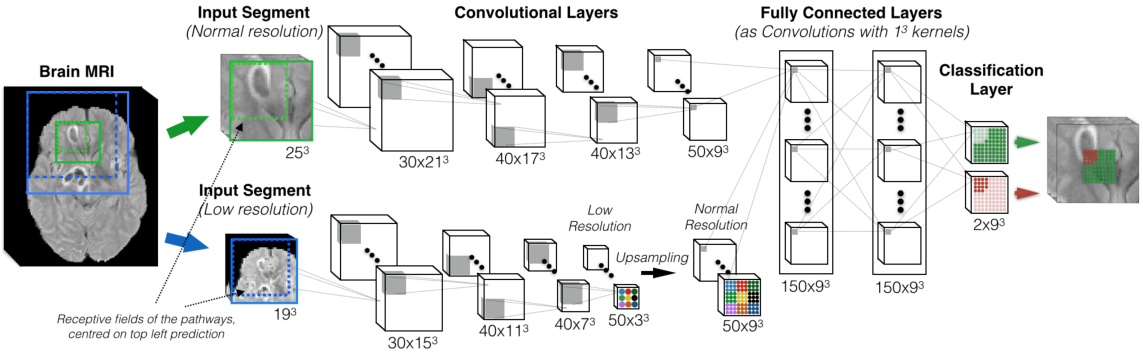


Figure 19: Deep Medic's model architecture. Figure from [66]

4.1.2 No New Net

No New Net (nnU-Net), which was developed by Isensee *et al.*, is a full deep learning pipeline for 3D biomedical image segmentation [67, 68]. No New Net's stated goal is to provide a standardized baseline against which to compare other algorithms, provide an 'out-of-the-box' segmentation method usable by non-experts, and provide a modular

framework in which new architectures and methods can be easily integrated and evaluated. Rather than relying on manual intervention for hyper-parameter selection and configuration, nnU-Net features a pre-planning script that analyses the provided data and uses heuristics to infer model and procedure hyper-parameters including, but not limited to batch size, patch size, data augmentation, and network topology. After training, nnU-Net tests different post-processing and ensembling strategies to further improve segmentation performance. No New Net was tested against 19 different biomedical segmentation data sets and reports state-of-the-art or near state-of-the-art results on each. A comparison of nnU-Net's procedure compared to the current practice of manual intervention is shown in Figure 20.

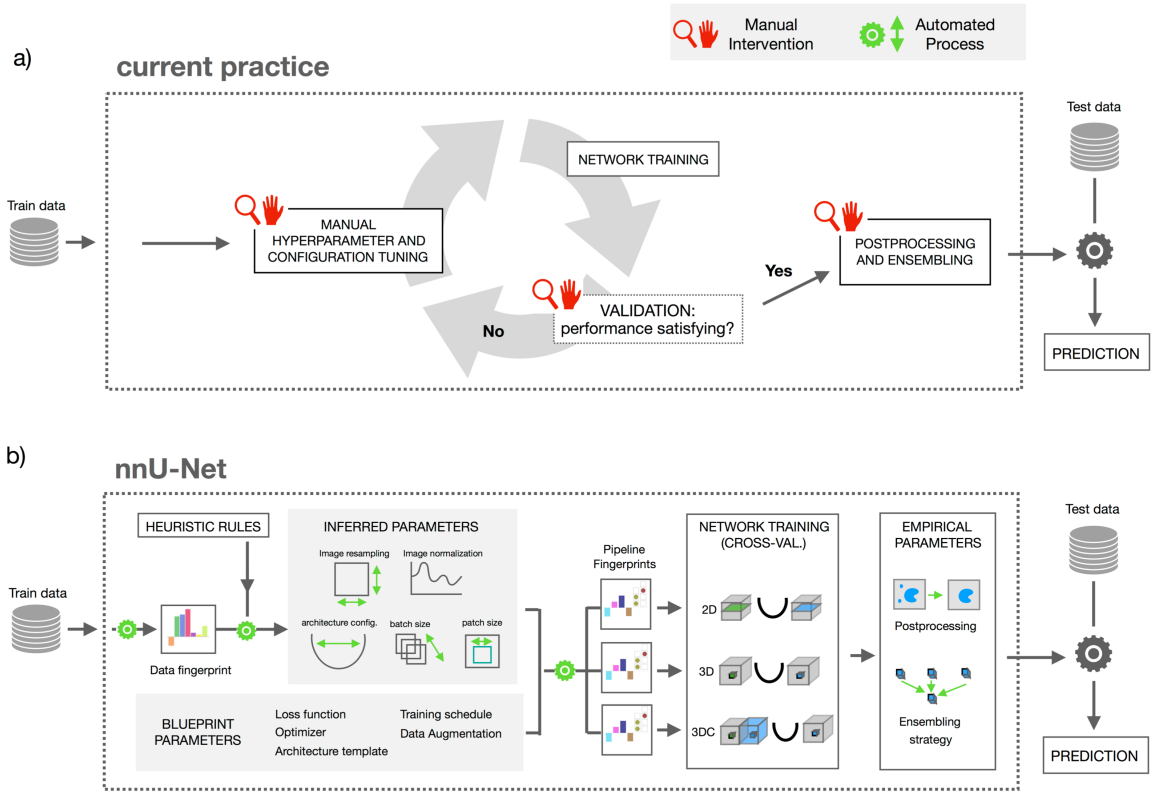


Figure 20: A comparison between nnU-Net's framework and the current practice. Figure from [67].

4.2 Methodology

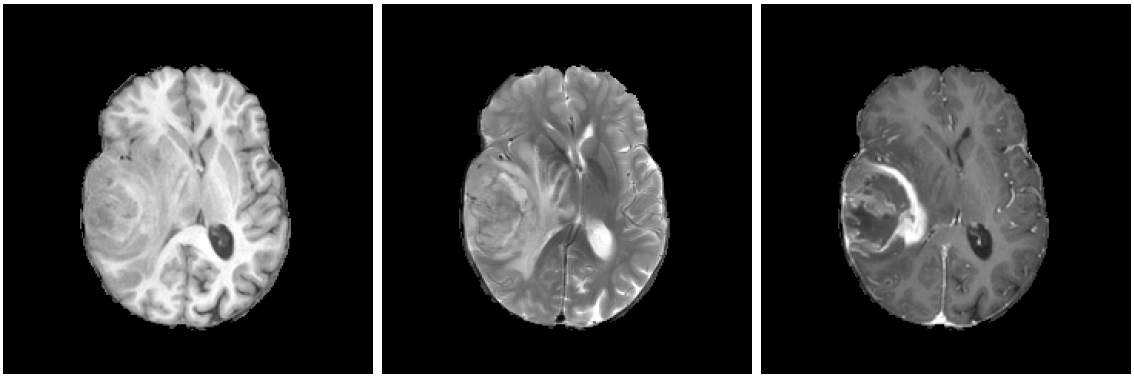
The Brain Tumour Segmentation (BraTS) data set [13] is used for an annual challenge seeking to identify and evaluate the current state-of-the-art methods in segmenting brain MR images. The primary task of the challenge is to segment a specific type of brain tumour known as a glioma. The 2020 publicly available version of the data set consists of 369 sets of images. Each set consists of four MR images from a single subject with different acquisition modes and one ground-truth segmentation mask. The MR images are T1 weighted, T2 weighted, post-contrast T1 weighted (T1CE), and T2 Fluid Attenuated Inversion Recovery (FLAIR). All images are pre-processed to be co-registered, interpolated to isometric 1mm^3 resolution, and skull stripped.

The ground truths were created by one to four raters and approved by expert neuro-radiologists. The ground-truth masks are segmented into four distinct categorical labels, namely: background, oedema, enhancing tumour, and non-enhancing tumour core.

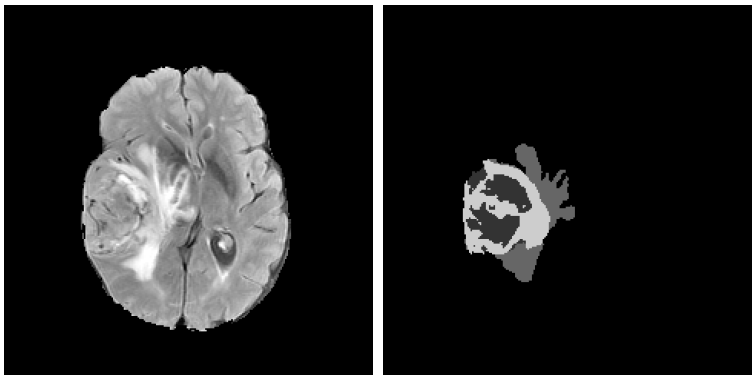
Each image's resolution is 240x240x155. The brain is centred in each image and the XY axes always represent the axial plane. An example of a set of images from one subject is shown in Figure 21.

4.2.1 Model Architecture

We used a fully convolutional architecture based on U-Net by Ronneberger *et al.* [15]. Input to the model is a 3D volume of arbitrary size with four channels, one for each MR acquisition mode. The model consists of four down-sampling blocks followed by four up-sampling blocks, and finally one softmax output block. Each block begins with a batched normalization layer followed by three 3D convolutional layers, followed by a re-sampling layer. For the down-sampling blocks, re-sampling is performed by 2-strided convolutional layers. For the up-sampling blocks, re-sampling is performed by a 2-strided transposed convolution. Prior to batched normalization in the up-sampling blocks, the output from the previous layer is concatenated with the output of the pre-sampling layer of the down-sampling block of the same size. All convolutional layers use the ELU activation function [26] and use padding to maintain consistent image output shapes. The final output block is identical to the sampling blocks except that the re-sampling layer is replaced by a convolutional layer with a



(a) T1 (b) T2 (c) T1 Contrast Enhanced



(d) FLAIR (e) Segmentation Mask

Figure 21: Example slices for each modality and the ground truth from one set of images in the BraTS 2020 data set [13]

.

softmax activation function yielding the predicted probability for each class for each voxel.

The number of convolutional kernels in each layer begins at 16 and doubles with each down-sampling block up to 256 at the bottom block. It then halves with each up-sampling block. The final layer has four kernels, one for each output class. There are three sizes of convolution kernels used: 3x3x3 (small), 5x5x5 (medium), and 7x7x7 (large). Larger kernels are used on blocks operating on higher resolution activations and smaller kernels are used on blocks operating on lower resolution activations. In this way, we allow for more detailed relationships on high resolution activations, while preserving resources on low resolution activations. The use of smaller kernels operating on small activations also mitigates the redundant effect of padding. The first two and last two blocks use large kernels, the third and third-to-last use medium

kernels, and the middle three blocks use small kernels. This model architecture is illustrated in Figure 22.

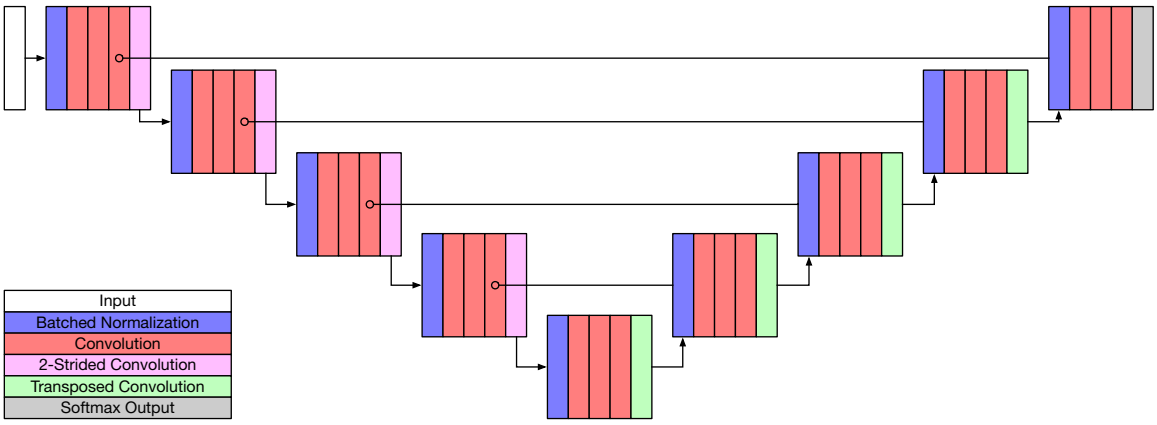


Figure 22: Architecture for U-Net model used for brain image segmentation

4.2.2 Loss and Optimization

As in Chapter 3, for training, we use a modified version of the weighted multi-class Dice loss score used in [59]. This metric was chosen because the Dice Similarity Coefficient is the primary evaluation metric used for BraTS competition and, in this way, we optimize for it as directly as possible. Weighting was used to mitigate the deleterious effects of high class imbalance. The loss score was calculated according to equations 30, 31, and 32.

The models are trained using the Adam optimizer [60] with a learning rate of 10^{-4}, $\beta_1 = 0.9$, and $\beta_2 = 0.999$.

4.2.3 Data Augmentation

During training, we randomly apply data augmentations on both the intensities and spatial distributions of the image voxels. Prior to intensity augmentations, the intensity values per channel are each linearly re-scaled such that the 99.5th percentile and the 0.5th percentile are 1.0 and 0.0 respectively. The following intensity augmentations are applied identically to each channel with probability p: (1) addition of Gaussian noise with $\mu = 0.0$ and $\sigma = 0.2$ ($p = 0.5$); (2) random shift in image intensity between -0.2 and 0.2 ($p = 0.5$); and (3) Gaussian smoothing with $\sigma \in [0.5, 0.9]$

($p = 0.3$). After intensity augmentations are applied, each channel's intensity values are normalized to have a zero mean and unit variance.

The following spatial augmentations are applied identically to each channel: (1) volume rotation between -5° to 5° around the x-axis ($p = 0.5$); (2) volume rotation between -2.5° to 2.5° around the y-axis ($p = 0.5$); (3) volume rotation between -2.5° to 2.5° around the z-axis ($p = 0.5$); (4) selection of eight random patches of size 128x128x128 ($p = 1$); and (5) horizontal flipping of the XY-plane ($p = 0.5$).

4.2.4 Hardware and Software Configuration

All networks were implemented in Keras [61] with TensorFlow 2.1.0 [62] as backend. Models were trained and run on a NVIDIA Titan Xp GPU with 12 GB VRAM with data augmentations performed on an Intel Xeon v4 CPU.

4.2.5 Training Procedure

The network was trained with a 5-fold cross validation scheme where, for each fold, 80% of the data was used for training and the remaining 20% was withheld for validation. For this data, we used the BraTS 2018 data set, a subset of the BraTS 2020 data set. We used this subset because, at the time, the official BraTS 2020 validation and testing data were not available to us so we reserved the difference between these two sets for the final evaluation.

In addition to the data augmentations applied to the data online, all produced patches were filtered so that the network would not train on any images for which the ground truth contained exclusively background labels. The network was trained for 300 epochs on batches of four patches. Each epoch consisted of a full pass through all of the training data. The models were trained using the Adam optimizer [60] with a learning rate of 10^{-4}, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. Training took roughly three days to complete.

At the end of each epoch, the model was evaluated against the validation data using the same loss function, categorical accuracy, and the mean of each class' approximated hard Dice similarity coefficient calculated using

$$\text{Coef}_c = \frac{2|T_c \cap P_c| + \epsilon}{|T_c| + |P_c| + \epsilon}, \tag{34}$$

where P_c and T_c represent the prediction and ground-truth masks for class c, and ϵ is a very small number used to correct for division by zero errors. These validation metrics were used to determine appropriate hyper-parameters such as learning rate and network depth.

4.2.6 Inference Procedure

The model architecture allows it to perform inference on 3D volumes of nearly arbitrary size. There are two limitations on the inference size:

1. The amount of available VRAM. The sizes of the expanded convolution tensors and resulting activations are proportional to the input size. Attempting to perform inference on too large of an image will result in an out-of-memory error and crash. One solution is to perform inference on patches of the input image and then later fuse the patch predictions into a single segmented image. Doing so can reduce overall performance since the individual patches lose the global context present in the whole image. Because the size of the images in the BraTS data set is constant, the complexity of the model architecture was specifically engineered to be able to perform inference on a single full image using the available system GPU without any additional steps.

2. The number of down-sampling blocks. For each down-sampling block, the resolution is cut in half along each axis to be later doubled by the up-sampling blocks. If the input size to any down-sampling block is not divisible by two along some axis then the output size is truncated and later up-sampling will not reproduce the original input size which is incompatible with the concatenation step. Because of this, any such model with n down-sampling blocks requires input sizes to be divisible by 2^n along each axis. Given that our model has four down-sampling blocks, each axis must be divisible by $2^4 = 16$. To resolve this issue, images not divisible by 16 along some axes are symmetrically zero-padded along those axes to the nearest multiple of 16. After prediction, this extra padding is removed from the result. Given that all the BraTS images are 240x240x155 in size, this effectively means that they are each augmented with five empty slices to the z-axis.

Before input to the model for inference, images are normalized in the same way

as during training, i.e., their intensities are re-scaled such that they have a zero mean and unit variance. Running the normalized images through the network yields the predicted probabilities for each class for each voxel. The argmax function is applied voxel-wise to produce the final predicted segmentation. Alternatively, instead of applying the argmax function immediately, we use the resulting models from each of the five folds to individually predict class probabilities, sum each of the class probabilities voxel-wise, and then apply the argmax function. In this way, we create an ensemble network using all of the network folds.

4.2.7 Running Benchmark Networks

In the following section, we describe the models and parameters used for running the two benchmark datasets.

No New Net

No New Net has multiple pre-made configurations including a 2D U-Net, a 3D full resolution U-Net, and a 3D U-Net cascade featuring a 3D low resolution U-Net followed by a 3D full resolution U-Net. For the purpose of this experiment, we trained and tested using the 3D full resolution U-Net. Running nnU-Net requires minimal effort. Input filenames must adhere to a specific format and the ground-truth segmentation mask labels must be sequential starting with zero for background. Following this, we run the provided pre-processing script and then start training with another provided script.

No New Net was trained using all default parameters, using five folds and its own data partitioning scheme. Each fold failed to trigger the built-in early stopping condition and trained until the preset maximum 1000 epochs (of 250 batches each), about five days each. We trained and evaluated only one pre-made configuration due to the high time cost.

Deep Medic

Similar to nnU-Net, Deep Medic comes with multiple pre-made configurations. For the purpose of this experiment, we trained and tested using the provided configuration named 'modelConfig_wide1'. Before running the training procedure, Deep Medic

requires explicit specification of all the input files and partitioning, names of output files, number of input channels, and number of output classes. Additionally, input images must be pre-processed to have zero mean and unit variance and segmentation mask labels must be sequential starting with zero for background.

Deep Medic was then trained according to its default parameters.

4.2.8 Testing

For all networks, for training, we used the BraTS 2018 data set. For testing, we used all of the 84 samples from the available BraTS 2020 data set that were not present in the 2018 data set. For each fold of our own network, the ensemble of our network, the Deep Medic network, and the highest performing fold of nnU-Net, we ran inference on each of the 84 images and computed the per-class hard Dice score using equation 34.

4.3 Results and Discussion

The distributions of Dice scores achieved on the testing set by each fold and ensemble of our network is presented in Figure 23 and the distributions achieved by the benchmark solution and our ensemble are presented in Figure 24. Tables 4 and 5 present mean and standard deviation summary statistics of the same data.

The enhancing tumour class had notably poor performance compared to the other classes on each fold and model. In the BraTS data set, this is the only label that is not present in every image. We conjecture that this resulted in diminished performance for two reasons: (1) The models were less exposed to this class during training and thus less able to learn patterns associated with it; and (2) The design of the Dice similarity coefficient is such that if there are no instances of a given class in a ground-truth segmentation mask then even just a single false positive prediction of this class will result in an overall Dice score of nearly zero ($\frac{\epsilon}{1+\epsilon}$). This would also explain why this is the only class to ever achieve a 1.0 Dice score in any model.

For our network, the ensemble of all folds achieved the highest mean Dice scores for each class compared to any of the individual folds. For most of the individual folds, the increase in Dice scores was significant, however, Fold 3 achieved mean scores very close to the ensemble with similar distribution. When comparing the standard deviations of each fold and the ensemble, there was no single consistent top

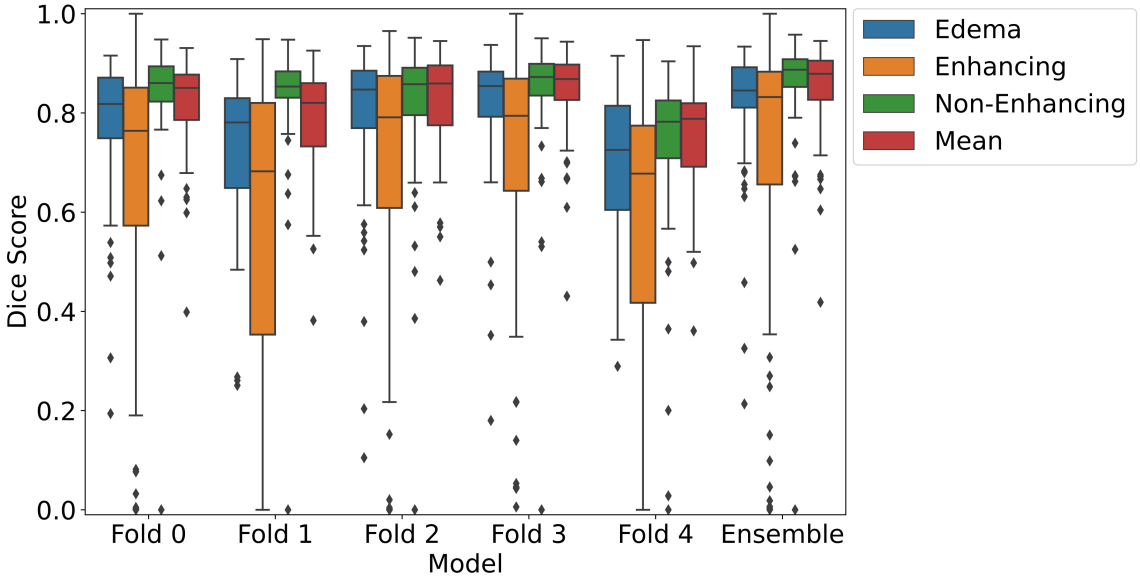


Figure 23: Distribution of per-class Dice scores for each model fold and ensemble as evaluated on the testing data.

Table 4: Summary of testing statistics for benchmark architectures for each model fold and ensemble. Numbers in bold indicate the column maximum for means and column minimum for standard deviations. Class label abbreviations: Oed is Oedema, Enh is Enhancing Tumour, N-Enh is Non-Enhancing Tumour

Model	**Mean**				**Standard Deviation**			
Class	**Oed**	**Enh**	**N-Enh**	**Mean**	**Oed**	**Enh**	**N-Enh**	**Mean**
Fold 0	0.781	0.664	0.841	0.762	0.133	0.268	0.114	0.116
Fold 1	0.730	0.589	0.838	0.719	0.145	0.291	**0.110**	0.129
Fold 2	0.794	0.683	0.818	0.766	0.148	0.272	0.135	0.132
Fold 3	0.814	0.708	0.848	0.790	0.124	**0.248**	0.119	**0.115**
Fold 4	0.692	0.576	0.733	0.667	0.154	0.274	0.162	0.146
Ensemble	**0.819**	**0.718**	**0.860**	**0.799**	**0.121**	0.264	0.117	0.118

performer.

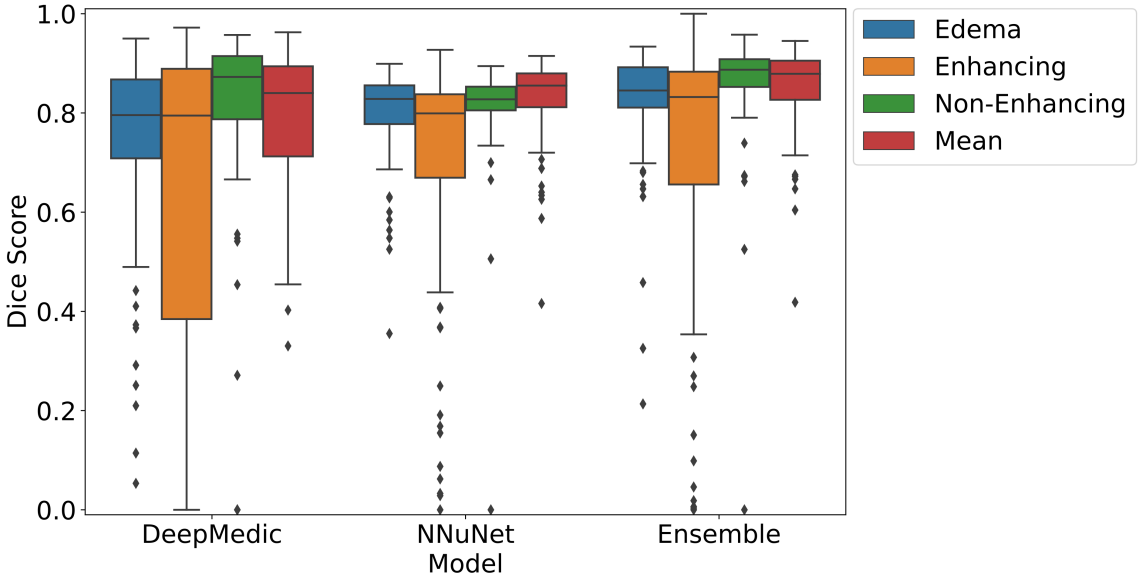


Figure 24: Distribution of per-class Dice scores for each benchmark architecture and our ensemble as evaluated on the testing data.

Table 5: Summary of testing statistics for benchmark architectures and our ensemble. Numbers in bold indicate the column maximum for means and column minimum for standard deviations. Class label abbreviations: Oed is Oedema, Enh is Enhancing Tumour, N-Enh is Non-Enhancing Tumour

Model	**Mean**				**Standard Deviation**			
Class	**Oed**	**Enh**	**N-Enh**	**Mean**	**Oed**	**Enh**	**N-Enh**	**Mean**
Deep Medic	0.745	0.633	0.815	0.731	0.193	0.304	0.176	0.179
nnU-Net	0.798	0.695	0.812	0.768	**0.096**	**0.239**	**0.105**	**0.114**
Ensemble	**0.819**	**0.718**	**0.860**	**0.799**	0.121	0.264	0.117	0.118

In the comparison of each solution, i.e. our ensemble, Deep Medic and nnU-Net, Deep Medic showed the poorest performance almost universally in both mean Dice scores and standard deviations with the exception of the Non-Enhancing Tumour class where it marginally outperformed nnU-Net in mean score. Our ensemble showed the highest performance in mean Dice scores for every class, but nnU-Net had the smallest standard deviations for every class. This shows that while our solution had the best overall performance, nnU-Net provides the most consistent predictions.

4.4 Conclusion

In this work, our results confirm that we achieved our goal of building a deep learning solution to brain tumour segmentation that outperforms existing high quality publicly available solutions based on mean Dice score. This shows that through careful adjustment of hyper-parameters and model architecture design around the specific constraints of our data set and available computational hardware resources, we can achieve better segmentation accuracy than with pre-made one-size-fits-all models. Because each solution operates completely differently with different choices in model architecture, training procedure, training data partitioning, hyper-parameters, data pre-processing, etc., it is difficult to ascribe our success to any single engineering decision. Future works to improve our solution further should make individual changes in isolation to better estimate their influence on the overall network performance. Furthermore, each Deep Medic and nnU-Net are capable of multiple different configurations, but only one of each was tested. More thorough testing of the different possible configurations would be needed to fully determine their capacity.

Chapter 5

Conclusion

In this dissertation, we presented two experiments in the domain of multi-modal biomedical image segmentation. Our first experiment in Chapter 3 sought to validate current practices in multi-stream approaches to *unpaired* multi-modal segmentation. Our results supported the idea of multi-stream approaches in general, but contradicted the specific belief present in the literature as to which configuration performs best. Our second experiment in Chapter 4 sought to validate a developed full deep learning segmentation pipeline as applied to *paired* multi-modal images against other existing publicly available pipelines. Our results found that our pipeline had the best performance in mean Dice scores for each class and showed that a pipeline carefully tuned to the specific computational constraints present can perform better than premade solutions. The success of both experiments using our deep learning pipeline with slightly different configurations on different tasks (one for unpaired learning of cardiac images and one for paired learning of brain tumour images) demonstrates an overall robustness in our methods.

5.1 Future Work

There are a number of possible avenues of future work based on our findings, specifically improving the interpretability of our models and addressing the shortcomings of our loss and evaluation function.

5.1.1 Model Interpretability

In Chapter 3, our experimental results showed that certain multi-stream architectures were able to produce better segmentations than others on unpaired multi-modal images, however, this provides us with no indication as to how or why they do. In a follow-up experiment, we showed that the models with the best test-time performance also had the fewest dead intermediate activations on one input image. We suggest that this experiment be repeated on a larger scale so that it carries greater statistical weight. Further, we propose that a new experiment be run that compares the similarities between the resulting network kernels to determine if there is significant deviation between the kernels produced by the different multi-stream networks and the kernels produced by the single-stream networks. As the number of architecture hyper-parameters grows, the process of statistically determining the optimal configuration becomes more and more intractable. For this reason, it is important that we determine why certain configurations had better performance than others and also why this was not the same for both CT and MR images.

5.1.2 Building on the Dice Similarity Coefficient

In our experiments, we chose the Dice Similarity Coefficient (DSC or Dice score) as the primary evaluation metric due to its prevalence in biomedical image segmentation challenges and the literature. We used a modified version of the Dice score for our loss functions to optimize our networks to our chosen evaluation metric as closely as possible. The design of the Dice score is such that it encodes both precision and recall, i.e. it rewards prediction of as much of a class as possible while penalizing over-prediction. In our experiments, we encountered two specific weaknesses in the DSC:

(1) The DSC is calculated based on the number of correct, incorrect, and true voxel classifications, but in no way encodes the spatial distribution of these classifications. In practice, the structures that we seek to segment in biomedical images tend to not be arbitrarily distributed in images, rather they are discrete volumes. In Figure 17 in Chapter 3, we showed that the mis-predicted voxels were largely distributed around the surfaces of these volumes and were only a few voxels thick. This shows that it is easy to locate the general region of interest and the difficulty in segmentation lies in determining the precise structure boundary. Volumes and surfaces grow at different

rates. As a simple example, a sphere's volume grows with its radius (r) in the order of r^3 while its surface area grows only in the order of r^2. As r increases, the ratio of surface area to volume decreases. For segmentation evaluation this means the ratio of erroneous predictions to correct predictions decreases. In this way, the DSC is correlated with this ratio and encodes bias in favour of larger, simpler volumes. In our experiments, we mitigated this effect in our loss score formulations by weighting each class inversely proportionally to its total number of voxels in the training data set. In doing so, we avoided the tendency for the networks to converge on exclusively predicting the background class, however, final test performances still favoured classes with the largest ground-truth volumes. Further work is needed to find modifications or alternatives to the DSC that place more emphasis on the volume surface voxels, for example, weighting voxel contributions to the loss function inversely proportionally to that voxel's distance from the ground-truth class surface.

(2) For cases where a given class is entirely absent from an image, the DSC is overly binary. In this case, correctly predicting zero voxels of this class yields a Dice score of 1.0 while a single mis-predicted voxel yields a near zero Dice score. We saw this effect in the results from Chapter 4 where there was exactly one class that was occasionally absent from the ground-truth segmentations. As a result, that class had by far the worst overall performance at test time. Further work is needed to find modifications or alternatives to the DSC that mitigate the impact of relatively small errors in such cases. For example, such work could look at the impacts of different sizes of ϵ. Given that, per our formulation, the loss resulting from the situation we just described would be $\frac{\epsilon}{1+\epsilon}$, larger values of ϵ would serve to reduce the spread of the distributions and may mitigate the impact of these small errors.

Overall, deep learning has proven to be an incredible tool for image processing. While it has high requirements to achieve maximal performance, this work has taken steps towards more fully taking advantage of what limited resources are available. With practically innumerable possible applications in medical imaging alone, we expect deep learning to be increasingly prominent going forward.

Bibliography

[1] G. N. Hounsfield. Computerized transverse axial scanning (tomography): Part 1. Description of system. *The British Journal of Radiology*, 46(552):1016–1022, December 1973. Publisher: The British Institute of Radiology.

[2] Raymond Damadian. Tumor Detection by Nuclear Magnetic Resonance. *Science*, 171(3976):1151, March 1971.

[3] David E Newman-Toker, Zheyu Wang, Yuxin Zhu, Najlla Nassery, Ali S Saber Tehrani, Adam C Schaffer, Chihwen Winnie Yu-Moe, Gwendolyn D Clemens, Mehdi Fanai, and Dana Siegal. Rate of diagnostic errors and serious misdiagnosis-related harms for major vascular events, infections, and cancers: toward a national incidence estimate using the "big three". *Diagnosis*, 1(ahead-of-print), 2020.

[4] Neumann Anders, Jonsdottir Kristjana, Mouridsen Kim, Hjort Niels, Gyldensted Carsten, Bizzi Alberto, Fiehler Jens, Gasparotti Roberto, Gillard Jonathan H., Hermier Marc, Kucinski Thomas, Larsson Elna-Marie, Sørensen Leif, and Østergaard Leif. Interrater Agreement for Final Infarct MRI Lesion Delineation. *Stroke*, 40(12):3768–3771, December 2009. Publisher: American Heart Association.

[5] Florent Lalys, Simon Esneault, Miguel Castro, Lucas Royer, Pascal Haigron, Vincent Auffret, and Jacques Tomasi. Automatic aortic root segmentation and anatomical landmarks detection for TAVI procedure planning. *Minimally Invasive Therapy and Allied Technologies*, 28(3):157–164, 2019. Publisher: Taylor & Francis.

[6] Omar Ibrahim Alirr and Ashrani Aizzuddin Abd. Rahni. Survey on Liver Tumour Resection Planning System: Steps, Techniques, and Parameters. *Journal of Digital Imaging*, 33(2):304–323, April 2020.

[7] Xiahai Zhuang, Lei Li, Christian Payer, Darko Štern, Martin Urschler, Mattias P Heinrich, Julien Oster, Chunliang Wang, Örjan Smedby, Cheng Bian, et al. Evaluation of algorithms for multi-modality whole heart segmentation: an open-access grand challenge. *Medical image analysis*, 58:101537, 2019.

[8] S. Shirly and K. Ramesh. Review on 2D and 3D MRI Image Segmentation Techniques. *Current Medical Imaging Formerly Current Medical Imaging Reviews*, 15(2):150–160, January 2019.

[9] Regina Pohle and Klaus D Toennies. Segmentation of medical images using adaptive region growing. In *Medical Imaging 2001: Image Processing*, volume 4322, pages 1337–1346. International Society for Optics and Photonics, 2001.

[10] Juan Eugenio Iglesias and Mert R. Sabuncu. Multi-atlas segmentation of biomedical images: A survey. *Medical Image Analysis*, 24(1):205–219, August 2015.

[11] Yitzchak Pfeffer, Arnaldo Mayer, Adi Zholkover, and Eli Konen. A system for automatic aorta sections measurements on chest ct. In *Medical Imaging 2016: Computer-Aided Diagnosis*, volume 9785, page 978505. International Society for Optics and Photonics, 2016.

[12] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, December 2015.

[13] Spyridon Bakas, Mauricio Reyes, Andras Jakab, Stefan Bauer, Markus Rempfler, Alessandro Crimi, et al. Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the brats challenge, 2019.

[14] Amber L. Simpson, Michela Antonelli, Spyridon Bakas, Michel Bilello, Keyvan Farahani, Bram van Ginneken, Annette Kopp-Schneider, Bennett A. Landman,

et al. A large annotated medical image dataset for the development and evaluation of segmentation algorithms. *arXiv:1902.09063 [cs, eess]*, February 2019. arXiv: 1902.09063.

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[16] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, May 2014. arXiv: 1312.6114.

[17] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]*, June 2014. arXiv: 1406.2661.

[18] Terence Parr and Jeremy Howard. The matrix calculus you need for deep learning. *CoRR*, abs/1802.01528, 2018.

[19] Alexander LeNail. NN-SVG: Publication-Ready Neural Network Architecture Schematics. *Journal of Open Source Software*, 4(33):747, January 2019.

[20] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[22] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

[23] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[24] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for Activation Functions. *arXiv:1710.05941 [cs]*, October 2017. arXiv: 1710.05941.

[25] Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying ReLU and Initialization: Theory and Numerical Examples. *arXiv:1903.06733 [cs, math, stat]*, November 2019. arXiv: 1903.06733.

[26] Djork Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pages 1–14, 2016.

[27] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[28] A Waibel, T Hanazawa, G Hinton, K Shikano, and K J Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, 1989.

[29] Steven B. Damelin and Willard Miller. *The mathematics of signal processing*. Number 48 in Cambridge texts in applied mathematics. Cambridge University Press, Cambridge ; New York, 2012. OCLC: ocn751752415.

[30] Peltarion. 2d convolution block, 2020.

[31] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[32] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. *arXiv:1411.4038 [cs]*, March 2015. arXiv: 1411.4038.

[33] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *arXiv:1609.04836 [cs, math]*, February 2017. arXiv: 1609.04836.

[34] Gerald Friedland, Alfredo Metere, and Mario Krell. A Practical Approach to Sizing Neural Networks. *arXiv:1810.02328 [cs]*, October 2018. arXiv: 1810.02328.

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[36] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[37] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *arXiv:1707.02968 [cs]*, August 2017. arXiv: 1707.02968.

[38] Xiahai Zhuang and Juan Shen. Multi-scale patch and multi-modality atlases for whole heart segmentation of mri. *Medical image analysis*, 31:77–87, 2016.

[39] Adhish Prasoon, Kersten Petersen, Christian Igel, François Lauze, Erik Dam, and Mads Nielsen. Deep feature learning for knee cartilage segmentation using a triplanar convolutional neural network. In *International conference on medical image computing and computer-assisted intervention*, pages 246–253. Springer, 2013.

[40] Pierrick Coupé, José V Manjón, Vladimir Fonov, Jens Pruessner, Montserrat Robles, and D Louis Collins. Patch-based segmentation using expert priors: Application to hippocampus and ventricle segmentation. *NeuroImage*, 54(2):940–954, 2011.

[41] Le Hou, Dimitris Samaras, Tahsin M Kurc, Yi Gao, James E Davis, and Joel H Saltz. Patch-based Convolutional Neural Network for Whole Slide Tissue Image Classification. *arXiv e-prints*, pages arXiv:1504.07947–arXiv:1504.07947, April 2015.

[42] Kamyar Nazeri, Azad Aminpour, and Mehran Ebrahimi. Two-Stage Convolutional Neural Network for Breast Cancer Histology Image Classification. *arXiv e-prints*, pages arXiv:1803.04054–arXiv:1803.04054, March 2018.

[43] Konstantinos Kamnitsas, Christian Ledig, Virginia F J Newcombe, Joanna P Simpson, Andrew D Kane, David K Menon, Daniel Rueckert, and Ben Glocker.

Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. *Medical Image Analysis*, 36:61–78, 2017.

[44] Lena Maier-Hein, Matthias Eisenmann, Annika Reinke, Sinan Onogur, Marko Stankovic, Patrick Scholz, Tal Arbel, Hrvoje Bogunovic, et al. Why rankings of biomedical image analysis competitions should be interpreted with care. *Nature Communications*, 9(1):5217–5217, 2018.

[45] Daniel P Huttenlocher, Gregory A Klanderman, and William J Rucklidge. Comparing Images Using the Hausdorff Distance. *850 IEEE Transactions On Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993.

[46] Mustafa Elattar, Esther Wiegerinck, Floortje van Kesteren, Lucile Dubois, Nils Planken, Ed Vanbavel, Jan Baan, and Henk Marquering. Automatic aortic root landmark detection in CTA images for preprocedural planning of transcatheter aortic valve implantation. *International Journal of Cardiovascular Imaging*, 32(3):501–511, 2016. Publisher: Springer Netherlands.

[47] Vanya Valindria, Nick Pawlowski, Martin Rajchl, Ioannis Lavdas, Eric Aboagye, Andrea Rockall, Daniel Rueckert, and Ben Glocker. Multi-modal learning from unpaired images: Application to multi-organ segmentation in ct and mri. *Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018*, 2018-January:547–556, 03 2018.

[48] Matt Crane. Questionable answers in question answering research: Reproducibility and variability of published results. *Transactions of the Association for Computational Linguistics*, 6:241–252, 2018.

[49] Pim Moeskops, Jelmer M. Wolterink, Bas H.M. van der Velden, Kenneth G.A. Gilhuijs, Tim Leiner, Max A. Viergever, and Ivana Išgum. Deep learning for multi-task medical image segmentation in multiple modalities. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9901 LNCS(October):478–486, 2016.

[50] Zizhao Zhang, Lin Yang, and Yefeng Zheng. Translating and Segmenting Multimodal Medical Volumes with Cycle- and Shape-Consistency Generative Adversarial Network. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 9242–9251, 2018.

[51] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv:1503.02531 [cs, stat]*, March 2015. arXiv: 1503.02531.

[52] Qi Dou, Quande Liu, Pheng Ann Heng, and Ben Glocker. Unpaired Multimodal Segmentation via Knowledge Distillation. *IEEE Transactions on Medical Imaging*, 1:1–1, 2020.

[53] Kang Li, Lequan Yu, Shujun Wang, and Pheng-Ann Heng. Towards cross-modality medical image segmentation with online mutual knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 775–783, 2020.

[54] Chen Chen, Chen Qin, Huaqi Qiu, Giacomo Tarroni, Jinming Duan, Wenjia Bai, and Daniel Rueckert. Deep learning for cardiac image segmentation: A review. *arXiv preprint arXiv:1911.03723*, 2019.

[55] N Tajbakhsh, L Jeyaseelan, Q Li, J Chiang, Z Wu, and X Ding. Embracing imperfect datasets: a review of deep learning solutions for medical image segmentation. arxiv. *arXiv preprint arXiv:1908.10454*, 2019.

[56] Christian Payer, Darko Štern, Horst Bischof, and Martin Urschler. Multi-label Whole Heart Segmentation Using CNNs and Anatomical Label Configurations. In Mihaela Pop, Maxime Sermesant, Pierre-Marc Jodoin, Alain Lalande, Xiahai Zhuang, Guang Yang, Alistair Young, and Olivier Bernard, editors, *Statistical Atlases and Computational Models of the Heart. ACDC and MMWHS Challenges*, volume 10663, pages 190–198. Springer International Publishing, Cham, 2018. Series Title: Lecture Notes in Computer Science.

[57] Ryohei Kuga, Asako Kanezaki, Masaki Samejima, Yusuke Sugano, and Yasuyuki Matsushita. Multi-task learning using multi-modal encoder-decoder networks with shared skip connections. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 403–411, 2017.

[58] Fausto Milletari, Nassir Navab, and Seyed Ahmad Ahmadi. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. *Proceedings - 2016 4th International Conference on 3D Vision, 3DV 2016*, pages 565–571, 2016.

[59] Chen Shen, Holger R Roth, Hirohisa Oda, Masahiro Oda, Yuichiro Hayashi, Kazunari Misawa, and Kensaku Mori. On the influence of dice loss function in multi-class organ segmentation of abdominal ct using 3d fully convolutional networks. *arXiv preprint arXiv:1801.05912*, 2018.

[60] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15, 2015.

[61] François Chollet et al. Keras. `https://keras.io`, 2015.

[62] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[63] G. Evren Keles, Brad Anderson, and Mitchel S. Berger. The effect of extent of resection on time to tumor progression and survival in patients with glioblastoma multiforme of the cerebral hemisphere. *Surgical Neurology*, 52(4):371–379, 1999.

[64] Mitchel S. Berger, Anastasia V. Deliganis, Jill Dobbins, and G. Evren Keles. The effect of extent of resection on recurrence in patients with low grade cerebral hemisphere gliomas. *Cancer*, 74(6):1784–1791, 1994.

[65] Jeffrey H. Wisoff, James M. Boyett, Mitchel S. Berger, Catherine Brant, Hao Li, Allan J. Yates, Patricia McGuire-Cullen, Patrick A. Turski, Leslie N. Sutton, Jeffrey C. Allen, Roger J. Packer, and Jonathan L. Finlay. Current neurosurgical management and the impact of the extent of resection in the treatment of malignant gliomas of childhood: a report of the Children's Cancer Group trial no. CCG-945. *Journal of neurosurgery*, 89(1):52–59, 1998.

[66] Konstantinos Kamnitsas, Liang Chen, Christian Ledig, Daniel Rueckert, and Ben Glocker. Multi-scale 3d convolutional neural networks for lesion segmentation in brain mri. *Ischemic stroke lesion segmentation*, 13:46, 2015.

[67] Fabian Isensee, Paul F Jäger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. Automated design of deep learning methods for biomedical image segmentation. *arXiv preprint arXiv:1904.08128*, 2019.

[68] Fabian Isensee, Philipp Kickingereder, Wolfgang Wick, Martin Bendszus, and Klaus H. Maier-Hein. No New-Net. In Alessandro Crimi, Spyridon Bakas, Hugo Kuijf, Farahani Keyvan, Mauricio Reyes, and Theo van Walsum, editors, *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 234–244, Cham, 2019. Springer International Publishing.