Artificial Neural Networks for Pectoral Muscle Segmentation in Breast Magnetic Resonance Images

Author:

Samuel Gilmour

Supervisors:

Dr Habib Baluwala

Professor Martyn Nash

Professor Poul Nielsen

Part IV Project Report
Department of Engineering Science

Faculty of Engineering
The University of Auckland

ABSTRACT

The Biomechanics for Breast Imaging Group at the Auckland Bioengineering Institute seeks to develop an automated method for constructing three-dimensional computational models of the female breast from magnetic resonance (MR) image volumes. Explicitly modelling the pectoral muscles has been shown to greatly improve the accuracy of these models – and as such, an automated method for segmenting the pectoral muscles is required. Artificial neural networks (ANNs) were applied to this problem. Initial experiments were conducted to assess how providing networks with the locations of anatomic landmarks influenced the accuracy of predicted segmentations. Experiments progressed to assess the influence of using the three-dimensional voxel intensity information contained in MR images, as well as the performance of deep architectures with either fully connected or convolutional layers of artificial neurons arranged in series. The network capable of producing the most accurate segmentations contained three convolutional layers, was provided with information on the location of three distinct landmarks, and utilised orthogonal planar patches of voxel intensity values extracted from two different spatial scales. The segmentations predicted by this network achieved a median Dice coefficient of 0.69, and median Hausdorff distance of 43.0 mm. The ANNs developed as part of this investigation have the potential to be adapted for segmentation of multiple breast tissue types – which would allow increasingly accurate models that have clinically useful applications in breast cancer imaging to be constructed.

ACKNOWLEDGEMENTS

I would like to thank Dr Habib Baluwala, Professor Martyn Nash and Professor Poul Nielsen for the guidance they have provided throughout the year. Their experience, insights, and enthusiasm have always flowed freely and proven to be invaluable sources of inspiration.

I also wish to acknowledge the contribution of the high performance computing facilities maintained by New Zealand eScience Infrastructure (NeSI) to the results of this investigation. These facilities are jointly funded by institutions collaborating with NeSI and through the Research Infrastructure programme run by the Ministry of Business, Innovation and Employment.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ABI	Auckland Bioengineering Institute
ANN	artificial neural network
CNN	convolutional neural network
CRF	conditional random field
DNN	deep neural network
MR	magnetic resonance
NeSI	New Zealand eScience Infrastructure

LIST OF NOTATION

a	artificial neuron activity
b	artificial neuron bias
$\mathbf{D}$	training dataset
L	loss function
ϵ	training learning rate
μ	training momentum
N	network function
P	predicted segmentation
σ	artificial neuron activation function
T	manually labelled segmentation
$\boldsymbol{\theta}$	network parameters
$\boldsymbol{v}$	training velocity
w_i	artificial neuron weight
x_i	artificial neuron input

1 INTRODUCTION

Every year, 3000 women are diagnosed with breast cancer in New Zealand alone. Another 600 are killed by the disease. Although surgical procedures are becoming more and more sophisticated, the most effective form of defence against breast cancer remains the detection of malignant tumours in their early stages.

A number of imaging systems have been designed for this purpose, and each comes with its own advantages and disadvantages. Mammograms and ultrasounds, for example, are used as initial screening procedures that are relatively inexpensive and easy to conduct. If either of these modalities suggests the presence of a tumorous mass, they are supplemented with a magnetic resonance (MR) scan that provides a more detailed picture of the breast – despite being expensive and difficult to obtain.

Tracking the movement of suspicious features between different imaging modalities and orientations is therefore an important part of the diagnostic process. The ability to do so also helps surgeons to operate on patients in orientations for which data has not been obtained. However, Sharma and Aggarwal (2010) note that it is impractical to manually construct a three-dimensional computational model that fulfils this purpose for every patient – due to the significant amount of time and effort required.

The Biomechanics for Breast Imaging Group at the Auckland Bioengineering Institute (ABI) aims to develop a method for automating this process on breast MR images. Constructing an accurate computational model relies on specific tissue types and structures being identified in these scans, and assigned the correct mechanical properties that allow deformations and displacements under loading to be predicted. The relatively stiff pectoral muscles have been shown by Babarenda Gamage, Boyes, Rajagopal, Nielsen, and Nash (2012) to have a particularly great influence on model accuracy – and therefore represent one of the most important tissue types to identify.

The aim of this project is to develop an automated method for segmenting the pectoral muscles in a breast MR image, which allows them to be explicitly represented in computational models. Ganesan, Acharya, Chua, Min, and Abraham (2013) note that this is not a trivial task. The tissue density and texture information contained in an MR image is not sufficient to distinguish the pectoral muscles from much of the surrounding breast tissue, meaning that approaches based only upon voxel intensity values are ineffective. In addition, the shape and curvature of the pectoral muscles vary greatly between patients – meaning that methods based purely on geometric considerations are unreliable. Artificial neural networks (ANNs) were investigated as a technique with the potential to overcome these difficulties.

Chapter 2 of this report provides background information on the problem, ANNs, and their modifications when being applied to image segmentation. Chapter 3 describes the methods used when developing a technique to automatically segment the pectoral muscles. Chapter 4 presents and discusses the results of the investigation, and Chapter 5 summarises the main conclusions drawn and avenues for future work.

2 BACKGROUND

This chapter provides background information on the problem and the techniques used to approach it. The anatomy of the breast and past techniques that have been used for pectoral muscle segmentation are both described, before ANNs and their modifications when applied to image segmentation are outlined.

2.1 ANATOMY OF THE BREAST

The female breasts lie on the anterior wall of the thoracic cavity. They are composed of skin on the exterior, and a combination of dense fibroglandular tissue and less dense fatty tissue on the interior (Pandya & Moore, 2011).

Behind both breasts lie the pectoralis major and minor muscles, which are composed of tissue far stiffer than the fibroglandular and fatty tissues in the breast. These muscles are referred to collectively as the *pectoral muscles* throughout this report.

Figure 2.1 depicts an axial cross-section through a typical three-dimensional MR image of the breast, and highlights the anatomic structures that will be referred to in this investigation. Tissue density is captured as a voxel intensity value – with darker voxels corresponding to more dense tissue and lighter voxels to less dense tissue.

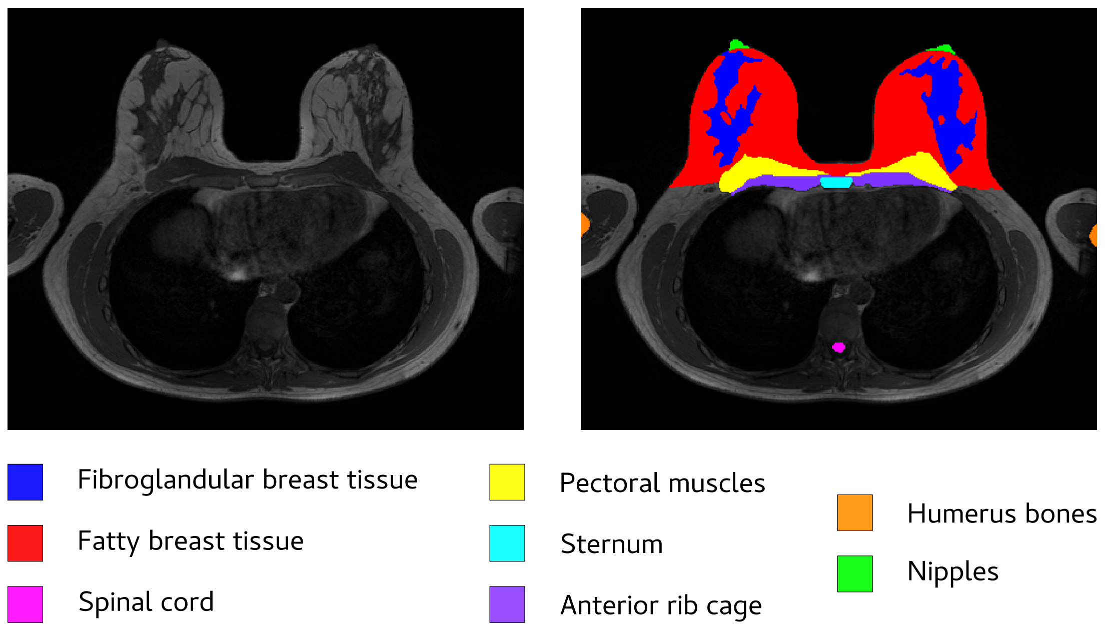


Figure 2.1: Axial cross-section of a MR image showing the anatomic structures relevant to this investigation.

2.2 EXISTING METHODS FOR PECTORAL MUSCLE SEGMENTATION

Image segmentation is the process of partitioning an image into distinct sections – each containing elements that share some problem-specific property. Segmentation of the pectoral muscles within medical images of the breast is a widely investigated application of this process. Although a range of methods have been applied to different imaging modalities, all are affected by one of the two fundamental difficulties identified by Ganesan et al. (2013):

1. Pectoral muscle tissue is often indistinguishable from the fibroglandular and rib cage tissue surrounding it, based solely on voxel intensity values.
2. The shape and size of pectoral muscles vary greatly between patients.

The most basic methods used to approach the problem are based on differences in voxel intensity values between the pectoral muscles and surrounding tissue (Sultana, Ciuc, & Strungaru, 2010; Saltanat, Hossain, & Alam, 2010; Yapa & Harada, 2008). These techniques are computationally inexpensive, but generally produce poor segmentations that bleed through the pectoral muscle boundary and into voxels of fibroglandular and rib cage tissue that share similar intensity values.

Atlas-based approaches seek to address this problem by explicitly considering the shape of the pectoral muscles. These methods perform segmentations of unseen MR images by registering them to match the geometry of an atlas containing manually segmented images (Gubern-Mérida, Kallenberg, Martí, & Karssemeijer, 2012). However, the accuracy of the final segmentation is severely limited by the accuracy of the registration phase – which performs poorly on patients whose pectoral muscle geometry is significantly different to those on which the atlas is based.

Another approach takes into account both intensity and shape information to find a linear approximation of the pectoral muscle boundary using a Hough transform of voxel intensities in a mammogram (Karssemeijer, 1998). While this method has been shown to achieve accurate segmentations in some cases, it struggles when the pectoral muscle boundary is highly curved. Attempts have been made to refine the linear approximation into a curve, but these are affected by variations in the shape and size of pectoral muscles from patient to patient (Yam et al., 2001).

All of these approaches require manual feature selection as a preprocessing step – for example, the information contained in an atlas often includes hand-selected features such as voxel intensity variance and kurtosis (Wu, Weinstein, & Kontos, 2012). The accuracy of these approaches is therefore limited by the relevance of the features selected, as well as the shortcomings of the methods themselves.

ANNs are tools that have been successfully applied to automatic brain MR image segmentation (de Brebisson & Montana, 2015; Lai, 2015), but never to pectoral muscle segmentation. This approach automates the feature selection process and takes into account both low-level features like voxel intensity and high-level features like the shape of surrounding anatomic structures. These characteristics give ANNs advantages over each of the methods previously discussed – and they therefore have the potential to be applied successfully to pectoral muscle segmentation.

2.3 ARTIFICIAL NEURAL NETWORKS

ANNs are a class of machine learning models that iteratively improve the estimation of an unknown function describing a dataset without being explicitly programmed to do so (Kröse & van der Smagt, 1993). In the case of pectoral muscle segmentation, an ANN must classify whether or not each voxel in an MR image is part of the pectoral muscles.

Building an ANN model in this context falls under the paradigm of supervised learning – meaning that the model must be *trained* on a dataset containing both input and output observations. During this process, model parameters are iteratively tuned such that the network can reproduce an output observation from the training set when supplied with the corresponding input. Once trained, the network can be used to generalise and predict outputs for input observations not present in the training set.

A specific instance of an ANN defines a *network function*, which is a mapping from an input onto an output in the dataset being modelled. The exact form of the network function and the way in which it is updated depend upon three components of the ANN. These are identified by MacKay (2003) to be:

ARCHITECTURE The underlying structure of the network. A network architecture describes the topological relationships between its constituent parts and determines the set from which the network function is drawn.

PARAMETERS The numerical values associated with constituent parts of the network. These parameters combine with the architecture to define the network function.

LEARNING RULE The rule by which parameters are adjusted to update the network function and improve its description of the dataset.

2.3.1 Artificial Neurons

The basic building block of any ANN is the artificial neuron. These are processing units that apply a function to an input vector and produce a scalar output, as illustrated in Figure 2.2.

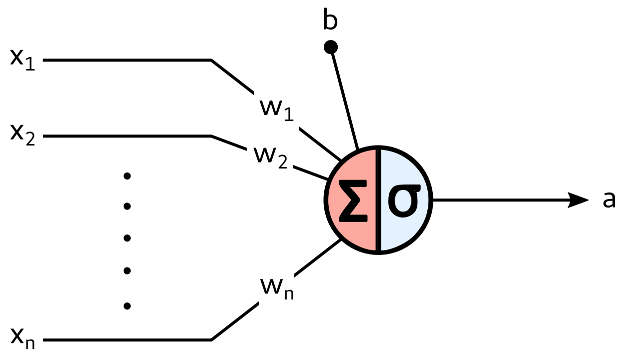


Figure 2.2: Schematic of an artificial neuron – showing inputs, x_i, weights, w_i, and bias, b, being summed, $\sum$, and transformed into activity, a, by an activation function, σ.

Each artificial neuron has some number, n, of inputs, x_i, which each have a weight, w_i, associated with them. An artificial neuron as a whole also has a scalar bias, b, and nonlinear activation function, σ, that is responsible for modelling complex trends in a dataset. These terms are combined according to Equation (1) to produce the output of the artificial neuron, termed its *activity*, a.

$$a(\boldsymbol{x}, \boldsymbol{w}, b) = \sigma\left(\sum_i w_i x_i + b\right) \tag{1}$$

Complete networks are formed by chaining artificial neurons together – meaning that the network function of an ANN is defined by a series of nested function compositions. The weights and biases associated with each artificial neuron are the parameters of the network that are adjusted by the learning rule to update the network function and improve its description of a dataset.

2.3.2 Network Architectures

The topology of artificial neurons in a network is defined by the network architecture. An example of a basic architecture is given in Figure 2.3, which shows the three different types of artificial neuron layers used in an ANN.

The input layer acts as the interface between the domain of the network function and the network itself. Similarly, the output layer acts as the interface between the network and the codomain of the network function.

Hidden layers fall between the input and output layers, and are responsible for modelling the majority of the nonlinearity present in a dataset. *Fully connected* layers are a type of hidden layer in which each artificial neuron receives input from every neuron in the previous layer, and supplies its activity to every neuron in the subsequent layer.

Connecting more hidden layers of artificial neurons in series with each other gives a network more learnable parameters and generally allows more complex patterns in a dataset to be modelled. However, it also increases the time required for training – and may result in networks *overfitting* to noise in the dataset, therefore losing the ability to generalise to unseen input observations.

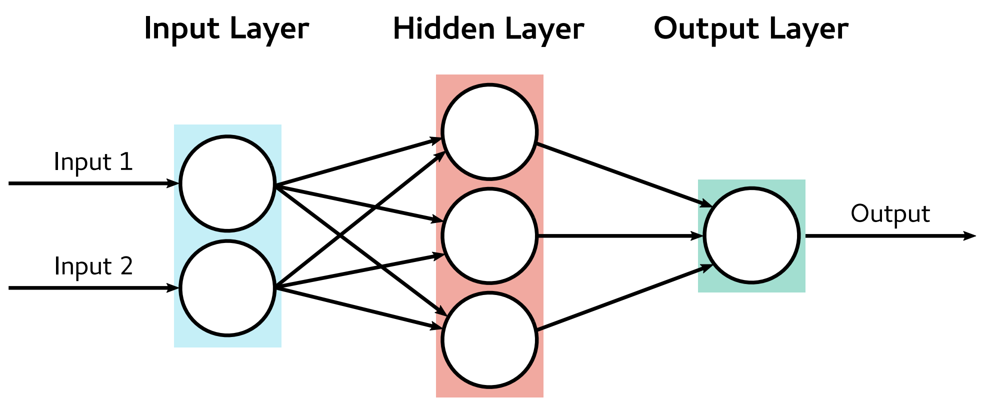


Figure 2.3: A standard ANN architecture, showing the three different types of artificial neuron layers. Adapted from Karpathy (2016).

2.3.3 Learning Rules

The learning rule is used to adjust the parameters associated with a network in order to improve its description of the dataset being modelled.

Solving this problem requires the introduction of a loss function, which evaluates the goodness of fit between a network prediction and an output observation in the dataset. For a training dataset, $\mathbf{D} \subseteq \mathbf{X} \times \mathbf{Y}$, containing pairs of input and output observations, $(x, y) \in \mathbf{D}$, a set of network parameters, $\theta \in \boldsymbol{\Theta}$, a network function, $N : \mathbf{X} \times \boldsymbol{\Theta} \mapsto \mathbf{Y}$, and a loss function, $L : \mathbf{Y} \times \mathbf{Y} \mapsto \mathbb{R}$, the learning rule attempts to find the optimal set of network parameters, θ^*, by solving the nonlinear optimisation problem in Equation (2).

$$\theta^* = \underset{\theta \in \boldsymbol{\Theta}}{\arg\min} \left\{ \sum_{(x,y) \in \mathbf{D}} L\left(N(x, \theta), y\right) \right\} \tag{2}$$

Gradient-based learning rules are the type most commonly employed, and make use of a differentiable loss function to direct their search in the parameter space. At the heart of almost all of these methods is the *backpropagation* algorithm developed by Werbos (1990) – which is an application of the chain rule that exploits the structure of an ANN to minimise the number of function evaluations required when approximating the gradient of the loss surface.

2.4 DEEP NEURAL NETWORKS

A deep neural network (DNN) is an ANN whose architecture contains multiple hidden layers. Configuring artificial neurons in this way means that the network function of a DNN contains more function compositions than an ANN with the same number of artificial neurons. DNNs are therefore able to represent functions that describe complex datasets more precisely than shallow networks – in fact, Leshno, Lin, Pinkus, and Schocken (1993) proved that a DNN with two hidden layers is capable of approximating any function to an arbitrary degree of accuracy, under certain conditions.

There are, however, benefits to increasing the number of layers in a DNN. Håstad and Goldmann (1991) showed that a network with L layers is capable of approximating certain functions that can only be approximated by a network with $L-1$ layers using exponentially as many artificial neurons. Training a network implicitly teaches each hidden layer to detect some *feature* of an input observation – in the context of image segmentation, these may include low-level features like edges or high-level features like shapes – and this information is passed through more processing stages in a deep network than in a shallow network. Figure 2.4 depicts this effect.

2.5 CONVOLUTIONAL NEURAL NETWORKS

Convolutional neural networks (CNNs) are a class of DNN with architectures that enforce a structured connectivity pattern between artificial neurons in successive layers. These architectures also restrict artificial neurons within layers to share weights and biases.

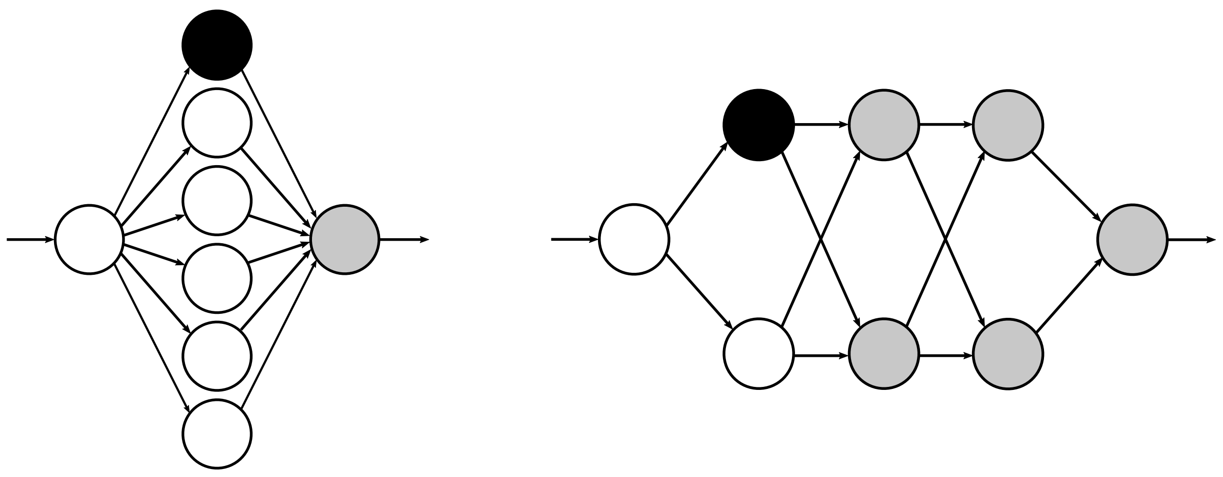


Figure 2.4: Comparison of shallow and deep networks with the same number of hidden artificial neurons. The activity of the black artificial neuron in the deep network (right) reaches more subsequent artificial neurons (shown in grey) than it does in the shallow network (left).

CNNs are particularly well-suited to datasets for which relevant patterns can be identified without regard to their position within an input observation. Because features like edges and shapes can appear anywhere in an image, CNNs are commonly used in the field of image processing. They have become practical for use only in recent years – largely due to the efficient GPU backpropagation algorithm introduced by Ciresan, Meier, Masci, Maria Gambardella, and Schmidhuber (2011) that significantly reduced their typically long training times.

Two types of hidden layers are used in CNNs: convolutional layers and pooling layers.

2.5.1 Convolutional Layers

Artificial neurons in a convolutional layer are arranged in a three-dimensional volume with a width, height and depth. The width and height dimensions correspond to spatial dimensions in a two-dimensional input image, and the depth dimension indexes different *feature maps* – groups of neurons that share weights and aim to identify specific features in the input. Each artificial neuron in a feature map has a *receptive field* of artificial neurons in the previous layer, which contains a subset of artificial neurons in the width and height dimensions but extends through the entire depth dimension.

Within a given feature map, artificial neurons share a weight matrix (or *kernel*) that contains one element for each artificial neuron in their receptive fields. The activity of an artificial neuron is obtained by convolving its kernel with the activities of artificial neurons in its receptive field and applying an activation function to the result. As the learning rule updates kernel weights, convolutional layers become tuned to extract increasingly relevant information. This process is depicted in Figure 2.5.

Artificial neurons in convolutional layers can also be arranged in four-dimensional space to support operations on three-dimensional input images. In this case, the first three dimensions of artificial neurons in the layer index spatial dimensions in the input image, and the final dimension indexes individual feature maps.

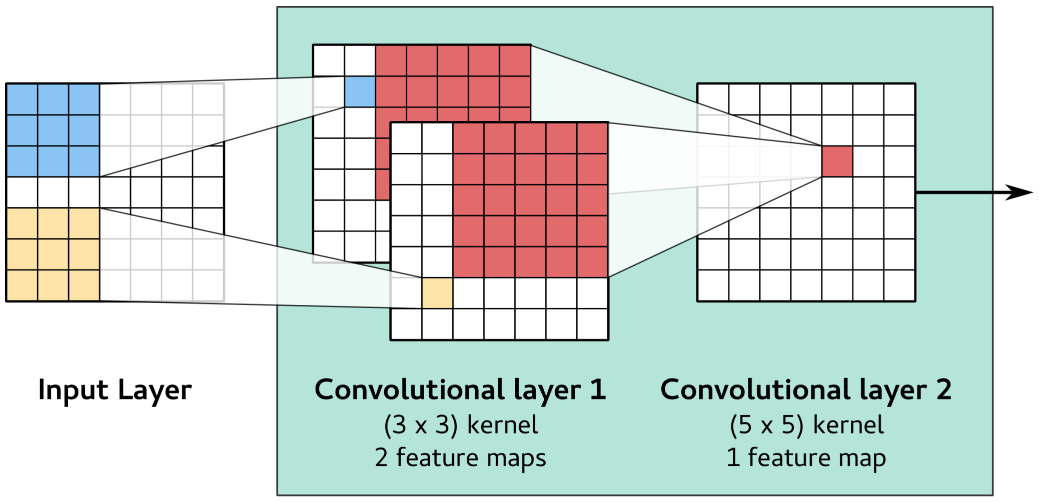


Figure 2.5: Two convolutional layers in series acting on a two-dimensional input image. Each sub-square represents a single artificial neuron, whose activity is calculated by convolving its kernel with artificial neurons in its receptive field and applying an activation function. The receptive fields of selected artificial neurons in each convolutional layer are indicated by their colours.

2.5.2 Pooling Layers

Pooling layers are used to reduce the dimension of a volume of artificial neuron activities generated by a convolutional layer. Max-pooling is the most frequently used pooling method. It partitions each feature map into small groups of artificial neurons with equal width and height, and calculates the output of artificial neurons in the pooling layer by selecting the greatest activity from within each partition. This operation is shown in Figure 2.6.

Pooling layers are positioned between convolutional layers, and act to downsample information while retaining the responses of the most active neurons. As a result, they allow feature maps in successive convolutional layers to gradually operate on spatial scales that increase in size with respect to the original image.

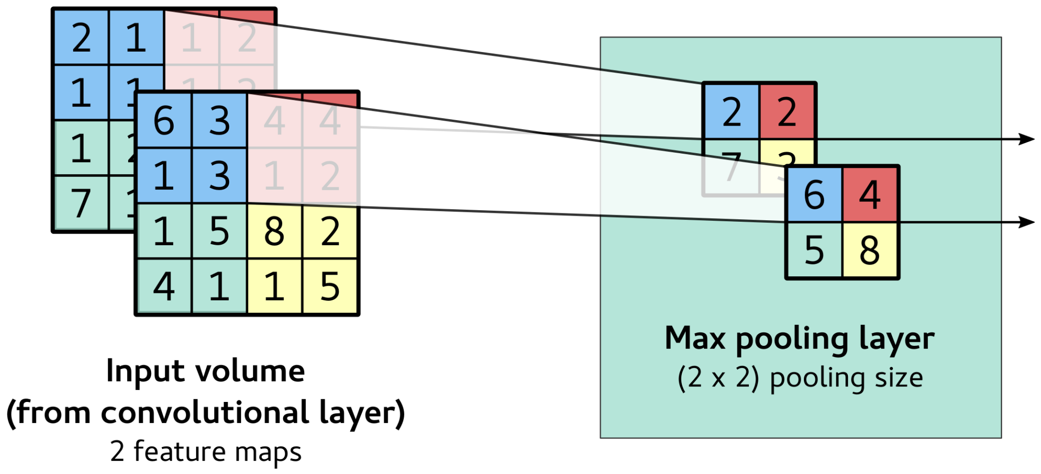


Figure 2.6: A single max-pooling layer acting on a three-dimensional input volume of artificial neuron activities generated by a convolutional layer.

2.6 STRUCTURED PREDICTION

While ANNs (and in particular, CNNs) are recognised for their ability to learn relevant features in a dataset, LeCun (2015) noted that they do not include any form of structured prediction. In the context of image segmentation, this means that the segmentation values of voxels in close proximity are not explicitly considered when a prediction is made.

While this does represent a disadvantage of the approach, the high density nature of data contained in images means that ANNs remain well-suited to image segmentation problems. Neighbouring voxels are naturally more likely to be assigned the same segmentation value – because the information contained in the dataset immediately surrounding them is extremely similar. Nevertheless, approaches to incorporating structured prediction in ANNs have been proposed.

One of these approaches connects whole networks together in series (Pinheiro & Collobert, 2014). The first network takes only voxel intensity values as input, and uses them to produce an initial segmentation. This segmentation is then passed as an additional input (along with the same voxel intensity data fed into the first network) into a secondary network – which produces a refined prediction that identifies and corrects the mistakes made by the first network. Any number of these networks can be connected together to increase the number of stages at which corrections are made.

2.7 SUMMARY

Segmentation of the pectoral muscles is a widely researched problem that has a number of well-recognised difficulties associated with it. ANNs are a class of machine learning models that have been successfully applied to brain segmentation in the past, and offer advantages over many of the methods currently being used to approach pectoral muscle segmentation. DNNs are a class of ANN with the ability to more precisely model complex trends in a dataset – while CNNs are a type of DNN with architectures that are well-suited for image segmentation.

ANNs are extremely flexible mathematical models. They can be used to formulate an image segmentation problem in a number of ways – and the precise nature of this formulation has a particularly large influence on the accuracy with which a network is able to segment the pectoral muscles in a breast MR image volume.

3 METHODS

This chapter provides a description and justification of how the techniques introduced in Chapter 2 were applied to the problem of automatic pectoral muscle segmentation. It begins by outlining the dataset, and proceeds to explain how different ANN models were formulated, trained, tested, and analysed.

3.1 BREAST IMAGE AND SEGMENTATION DATASETS

3.1.1 Volume Information

The complete dataset contained MR images taken from 60 healthy female participants aged between 19 and 75. Each image volume was represented as a volume of voxels with dimensions 448 (sagittal) $\times$ 448 (coronal) $\times$ 256 (axial). Two pieces of information were available for each voxel:

INTENSITY Tissue density was captured as a voxel intensity value, with a higher value corresponding to a higher density. The raw data contained integer values, which were converted to floating point values and standardised to give the dataset a mean of 0 and standard deviation of 1. This was required to avoid networks saturating during training, where the learning rule becomes impeded by vanishing gradients (Krizhevsky, Sutskever, & Hinton, 2012).

SEGMENTATION VALUE Each voxel was manually assigned a boolean value by a medical professional to indicate whether or not it corresponded to a pectoral muscle.

One further piece of information was provided for each image volume:

LANDMARK COORDINATES A tool developed by Baluwala, Malcolm, Jor, Nielsen, and Nash (2015) was used to automatically detect zero-dimensional *landmarks* in each image volume. The coordinates of six landmarks were obtained: the sternal angle, left and right nipples, left and right humerus balls, and spinal cord.

3.1.2 Training and Testing Approach

All ANNs were trained and tested using different sets of image volumes. 15 image volumes were randomly sampled to be used as the testing set, and the remaining 45 formed the training set. This approach was preferred over random resampling or k-fold cross-validation (Refaeilzadeh, Tang, & Liu, 2009) due to the prohibitively long periods of time that would have been required to train and test networks on multiple subsets of image volumes. Despite these techniques producing estimates of network performance that suffer from lower variability, the method chosen to be used is commonly applied in practice (Demšar, 2006) and produced statistically significant results.

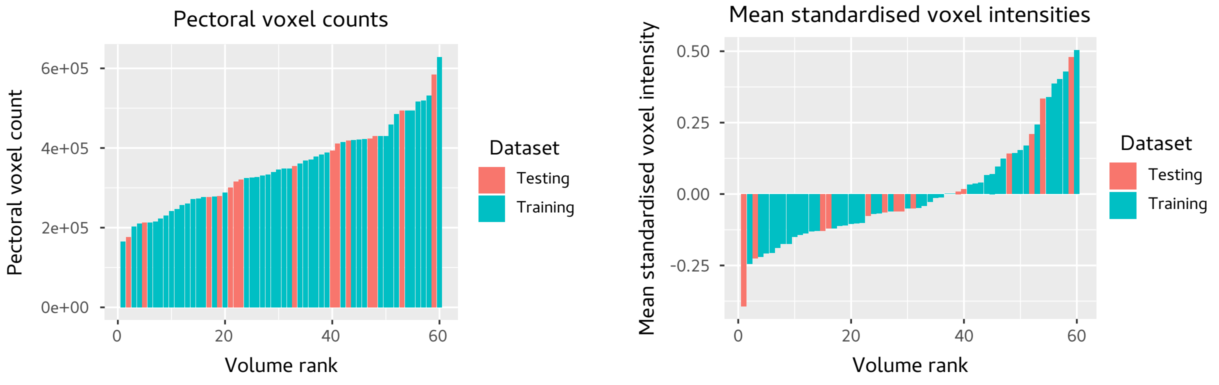


Figure 3.1: Variation in pectoral muscle voxel count and mean standardised voxel intensity of image volumes in the training and testing sets.

It was important that the randomly selected volumes in the training and testing sets fairly represented the entire dataset. Figure 3.1 shows how the volumes selected in both sets varied according to their pectoral muscle voxel counts and mean standardised voxel intensity – a measure of the average density of breast tissue surrounding the pectoral muscles. One volume in the testing set had an abnormally low mean voxel intensity due to a defect in the imaging process, but was included to assess the robustness of network predictions.

3.2 NETWORK INPUTS

A main strength of ANNs is their ability to transform feature selection into an automated process conducted by the learning rule. The information supplied to the network therefore did not need to include any of the manually selected features often used in image processing, such as intensity gradients and edges. Instead, the three main pieces of information that could be supplied to networks were *voxel intensity values*, *voxel position relative to landmarks*, and *previously predicted segmentation values*.

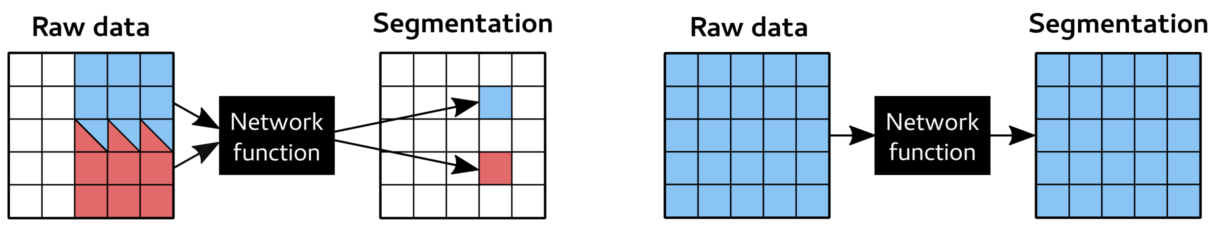
3.2.1 Voxel Intensity Values

Voxel intensity information could be supplied to a network through one of two different approaches, which are illustrated in Figure 3.2.

PATCH-WISE Using this approach, an individual voxel corresponds to a single datapoint – but instead of only passing the intensity value of this voxel into the ANN, a *patch* of neighbouring voxel intensities is also supplied. A network is trained to classify whether or not each of these datapoints individually corresponds to a pectoral muscle.

IMAGE-WISE This approach simultaneously passes an entire image volume into the ANN, and produces an output sharing the same dimension as the input and containing boolean pectoral muscle segmentation values for each voxel.

Networks used for three-dimensional medical image segmentation most commonly use the patch-wise training approach. This is primarily due to the use of convolutional layers in networks applied to this problem. While GPU-optimised implementations of the two-dimensional convolution operation are widely and freely available, similar



(a) Patchwise training. **(b)** Imagewise training.

Figure 3.2: The contrasting methods used by patch-wise and image-wise training approaches. The patch-wise approach was used for training all networks in this investigation.

implementations for a three-dimensional convolution are not. Using an image-wise CNN to segment a volume requires three-dimensional convolutions to be performed – and without a GPU-optimised implementation, the training time and memory requirements can be prohibitively large.

In addition, the image-wise approach to a three-dimensional problem cannot be decomposed into a series of two-dimensional problems without sacrificing the use of three-dimensional information. On the other hand, a patch-wise approach can supply three-dimensional information surrounding a voxel by using a set of orthogonal, two-dimensional patches – which allows established two-dimensional processing techniques to be used while retaining three-dimensional context around a voxel.

Long, Shelhamer, and Darrell (2015) reported that patch-wise and image-wise approaches both achieved comparable accuracy on image segmentation problems – and Liang, Hu, and Zhang (2015) concluded that, in some cases, the image-wise approach could produce a network that suffered from overfitting when compared with a patch-wise network trained on the same dataset. Taking all factors into account, the patch-wise training approach was adopted for this investigation.

3.2.2 Voxel Position Relative to Landmarks

Because the patch-wise approach was used for supplying voxel intensity information to networks, each voxel corresponded to a single datapoint. Landmark information was attached to each of these datapoints by calculating a vector containing the position of that voxel relative to each landmark.

3.2.3 Previously Predicted Segmentation Values

The structured prediction approach described in Section 2.6 connects networks together in series and uses the segmentation values predicted by an initial network as supplementary input for a subsequent network. This technique was investigated in the final experiment. Data was supplied to a secondary network by extracting patches of segmentation values predicted by an initial network, in the same way that voxel intensity patches were extracted.

3.3 TRAINING VOXEL SAMPLING

If every voxel in the set of training image volumes was used as a datapoint, the most complex networks in this investigation would have taken 10 years to train on the hardware available. Training therefore had to be conducted on a subset of all available voxels in order to achieve a training time that was not prohibitively large. For every network, 25,000 voxels were selected from each training image volume to give a total of 1,125,000 datapoints – more than the typical number used in other applications of ANNs to medical image segmentation (de Brebisson & Montana, 2015; Lai, 2015).

The nature of the datapoints chosen for training a network heavily influenced its ability to generalise to unseen examples. For example, a network trained on a high proportion of voxels that were not labelled as pectoral muscles predicted a high proportion of false negatives. It was therefore necessary to develop strategies that obtained training voxels without simply choosing them randomly from the set of all voxels in training image volumes.

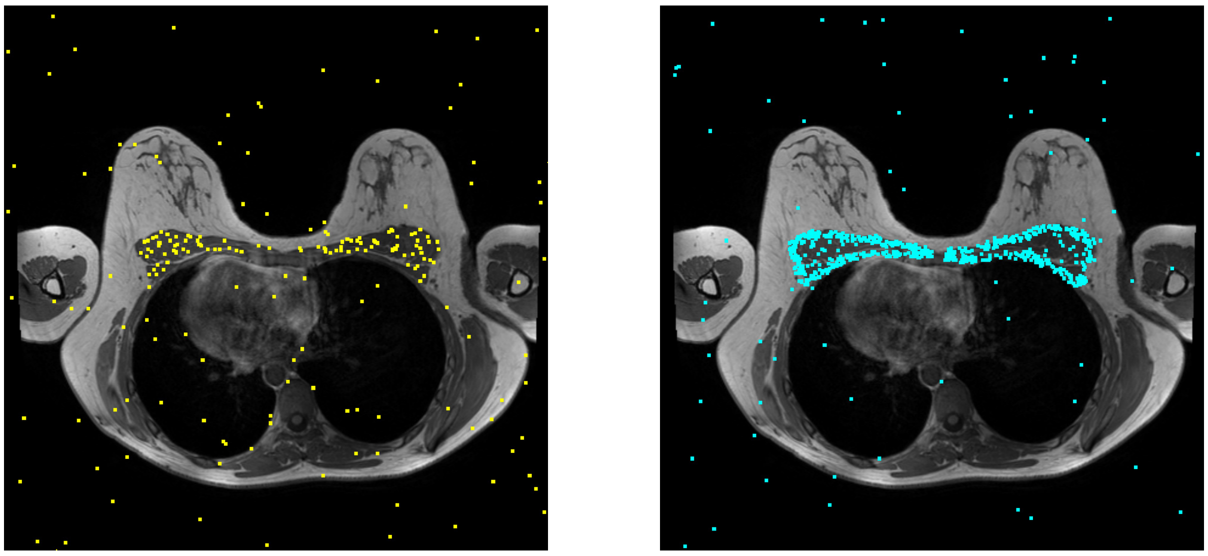
3.3.1 Balanced Sampling

This sampling method randomly selected training datapoints by balancing the number of pectoral muscle voxels with an equal number of voxels that did not correspond to the pectoral muscles.

3.3.2 Boundary Sampling

This sampling approach specifically targeted voxels located at the boundary between the pectoral muscles and surrounding tissue. 50% of all training datapoints were randomly sampled from voxels that were within two voxels of the boundary, and the remaining 50% were sampled in equal measures from voxels that corresponded to the pectoral muscles and those that did not.

Figure 3.3 shows an axial slice through an MR image that compares the training datapoints obtained by each approach.



(a) Balanced sampling.　　**(b)** Boundary sampling.

Figure 3.3: Comparison of voxel sampling strategies. Coloured voxels indicate those that were selected as training datapoints by each approach.

3.4 TRAINING AND TESTING APPROACHES

3.4.1 Training Data Supply

Batch and *online* training approaches lie at opposite ends of the spectrum governing how training data could be supplied to an ANN in this investigation. When a gradient-based learning rule is used for batch training, estimates of the loss function gradient with respect to the network parameters are calculated across all datapoints. In online training, gradient estimates are calculated for individual datapoints.

By nature, the batch training approach finds a more precise estimate of the true gradient at each update (Bottou, Curtis, & Nocedal, 2016). It has, however, been shown to converge orders of magnitude slower than the online approach while achieving similar accuracy in predictions on unseen examples (Wilson & Martinez, 2003).

A hybrid of batch and online training was used in this investigation. Groups containing 5,000 datapoints were supplied to the networks, and the gradient of the loss function with respect to the parameter set was estimated across all these points. Networks were trained for 100 epochs on each group before the next group was retrieved.

Within each group of datapoints, the proportion classified as pectoral muscles was kept constant. Byeon, Breuel, Raue, and Liwicki (2015) found this approach to improve the ability of a network to generalise when presented with unseen examples.

3.4.2 Loss Function and Learning Rule

Within each group, Nesterov's accelerated gradient descent method was used as the learning rule. This backpropagation-based algorithm estimates the loss function gradient by making use of a prediction/correction approach – and it has been shown to offer quick and stable convergence on a wide variety of problems (Sutskever, Martens, Dahl, & Hinton, 2013). It updates network parameters at each training iteration according to Equations (3) and (4):

$$\boldsymbol{\nu}_t = \mu\boldsymbol{\nu}_{t-1} - \epsilon\frac{\partial L}{\partial \theta}(\theta_{t-1} + \mu\boldsymbol{\nu}_{t-1}) \tag{3}$$

$$\theta_t = \theta_{t-1} + \boldsymbol{\nu}_t \tag{4}$$

where $\boldsymbol{\nu}_t$ is the *velocity* of training at iteration t, θ_t is the set of network parameters, μ is the *momentum* and ϵ is the *learning rate*. The dimensionless momentum and learning rate were set to 0.9 and 0.001 respectively – selected through experimentation based on values used in related studies (Liao, Kodagoda, Wang, Shi, & Liu, 2016). Network parameters were initialised using the Xavier method (Glorot & Bengio, 2010).

The output layer of each network contained a single neuron with the sigmoidal activation function provided in Equation (5). This function forced the network output to fall in the interval $(0, 1)$, and can be interpreted as a layer that added a logistic regression step to provide a measure of the likelihood that a particular voxel corresponded to a pectoral muscle (Dreiseitl & Ohno-Machado, 2002).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

The binary cross-entropy (BCE) function commonly used for boolean classification problems was chosen as the loss function for all networks. It is defined according to Equation (6) – for a dataset, $\mathbf{D} \subseteq \mathbf{X} \times \{0,1\}$, observation, $(x, y) \in \mathbf{D}$, set of network parameters, $\theta \in \Theta$, and network function $N : \mathbf{X} \times \Theta \mapsto (0,1)$.

$$\mathrm{BCE}(x, y, \theta) = -y \ln(N(x, \theta)) - (1-y)\ln(1 - N(x, \theta)) \tag{6}$$

3.4.3 Testing Approach

Segmentations were predicted for image volumes in the testing set by extracting data from every voxel they contained, and passing each individual datapoint through a network. For each datapoint, the network produced a predicted segmentation value that lay in the interval $(0,1)$. The final boolean segmentation was obtained by rounding these segmentation values to the nearest of 0 and 1.

3.5 NETWORK ARCHITECTURES

Six experiments were conducted to assess the effect of different network architectures and training approaches on the accuracy of predicted segmentations. This section details the design decisions made when constructing these architectures, and provides schematics of each.

Developing the architecture of an ANN is a process that suffers greatly from the curse of dimensionality. The vast numbers of design decisions and tunable parameters make it impractical to conduct an exhaustive search over all architectures. This necessitated the use of an approach that gradually built up model complexity by assessing the impact of each new addition to an architecture on the accuracy of its predicted segmentations. As a result – even if a complex addition to a network architecture made a more basic existing feature redundant, the presence of this existing feature did not greatly impact training time or contribute significantly to overfitting.

3.5.1 General Design

Every hidden neuron in all networks was equipped with the *rectifier* activation function introduced by Glorot, Bordes, and Bengio (2011) – which has been shown to reduce the time taken for convergence during training by up to 25% (Krizhevsky et al., 2012). Using the notation for neuron inputs and parameters previously introduced in Section 2.3.1, this function calculates the activity of a neuron using Equation (7).

$$\sigma(x, w, b) = \max(0, \sum_i w_i x_i + b) \tag{7}$$

The size of the voxel intensity patches used as network inputs had a large influence on model complexity – and was chosen to strike a balance between providing enough information and achieving a manageable training time. A study that used ANNs to segment anatomical structures of a similar size to the pectoral muscles found that increasing the patch size above 25 x 25 voxels provided marginal improvement in the accuracy of network predictions – while resulting in training times that grew significantly (Lai, 2015). This size was chosen to be used for patches in all networks.

3.5.2 Without Landmarks vs. With Landmarks

The first experiment made use of an architecture containing as little complexity as possible – with the objective of determining how supplying landmark information to a network influenced the accuracy of its predicted segmentations. Networks were designed based on four different subsets of the six available landmarks:

NO LANDMARKS No landmark information was supplied to a network.

THREE LANDMARKS The information supplied to a network included the distances between a voxel and the left nipple, the right nipple, and the sternal angle. These dimensions were added first due to their close proximity to the breast.

FIVE LANDMARKS These networks used the three landmarks given above, as well as the left and right humerus balls. The humerus balls were the landmarks next most localised to the breast region, and lay on a coronal plane that approximately aligned with the posterior surface of the pectoral muscles.

SIX LANDMARKS The distance between a voxel and the spinal cord (the landmark farthest from the breast) was added to the five dimensions described above.

Landmark information was supplemented with a two-dimensional patch of voxel intensities from a single orientation – either axial, coronal or sagittal. Networks using patches from each of these three orthogonal planes were tested separately. In each network, the landmark information and voxel intensity values were fed into a single fully connected hidden layer, and the networks were trained on voxels sampled using the balanced approach. The complete network architecture is illustrated in Figure 3.4.

For the networks including no landmarks, 1000 artificial neurons were included in the hidden layer to provide sufficient model complexity while complying with the guideline that single-layer models should have fewer hidden artificial neurons than twice their number of inputs (Berry & Linoff, 1997). It should be noted that this was only a general guideline, and changing the internal complexity of the model was investigated in subsequent experiments. Networks that included landmark information were provided with fewer artificial neurons in the hidden layer, so that the total number of learnable parameters in each network remained constant.

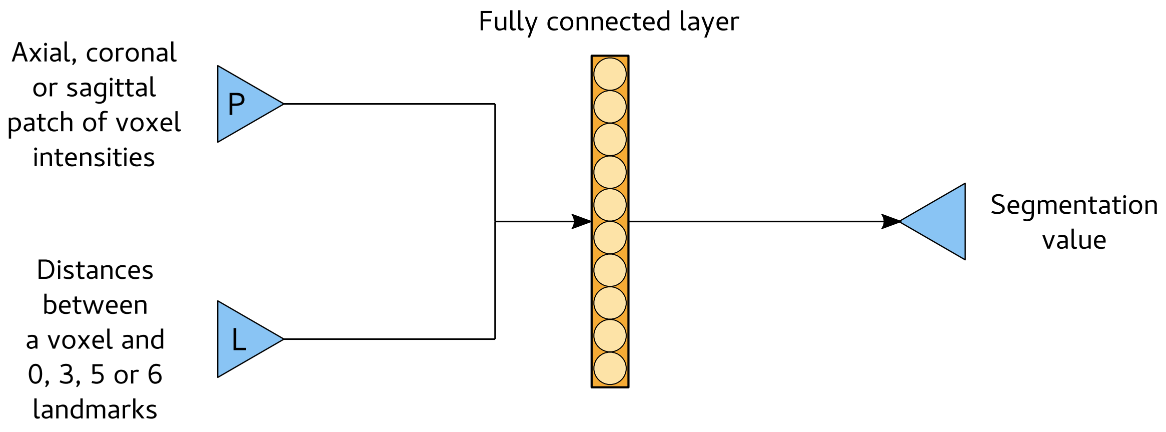

Fully connected layer
Axial, coronal or sagittal patch of voxel intensities
P
Distances between a voxel and 0, 3, 5 or 6 landmarks
L
Segmentation value

Figure 3.4: Schematic of the network architecture used to assess the effect on segmentation accuracy of providing landmark information. Details of all parameters chosen are provided in Table A.1.

In summary – each of the four subsets of landmarks were tested separately with patches taken from the axial, coronal, and sagittal orientations. A total of twelve networks were trained and tested for this experiment.

3.5.3 Balanced Sampling vs. Boundary Sampling

This experiment was conducted to determine the effect of the voxel sampling method (used to obtain training datapoints) on the accuracy of a predicted segmentation. Networks using patches taken from each of the three orthogonal planes were tested separately – and all networks were provided with information about three landmarks. Each network architecture was trained separately on voxels obtained using the balanced and boundary sampling approaches defined in Sections 3.3.1 and 3.3.2 respectively. A total of six networks were tested as part of this experiment.

3.5.4 Single Patch vs. Multiple Patches

This experiment assessed how providing voxel intensity values taken from a three-dimensional context around a voxel influenced the accuracy of segmentations. A new network architecture, which combined all three orthogonal patches and information about three landmarks, was used for this purpose – and is depicted in Figure 3.5. All networks were trained on voxels obtained using the boundary sampling approach.

The number of neurons in the hidden layer of the new architecture was initially set so that its total number of learnable parameters matched the previous architectures that were based on single patches. The second part of this experiment increased this number of artificial neurons to 3000. Networks using three-dimensional patches were also tested, but eventually discarded due to exceedingly long training times.

In total, this experiment required eight networks to be trained and tested – three with patches taken from individual orthogonal planes, and a fourth with all three combined, with these four cases being applied to networks with both the original and increased numbers of artificial neurons in the hidden layer.

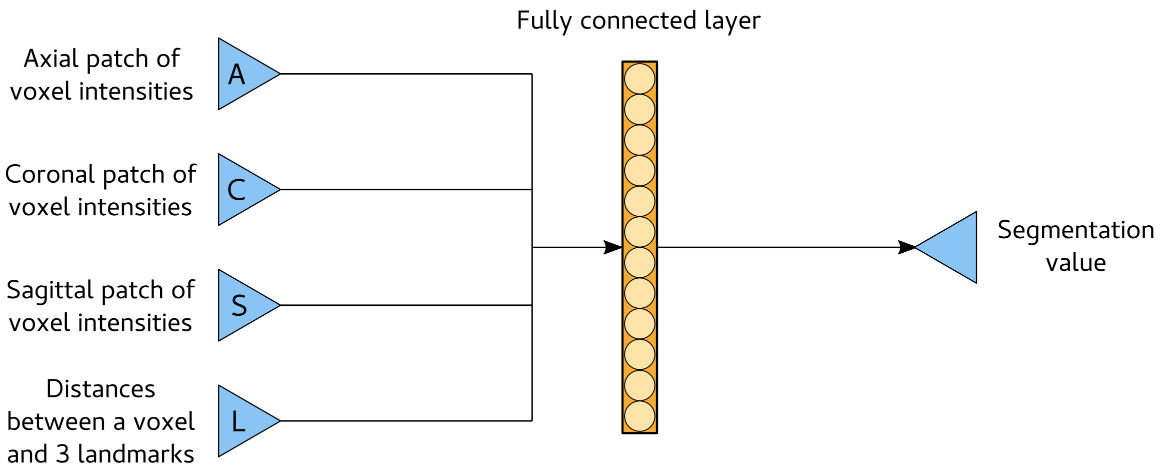

Fully connected layer
Axial patch of voxel intensities
A
Coronal patch of voxel intensities
C
Sagittal patch of voxel intensities
S
Distances between a voxel and 3 landmarks
L
Segmentation value

Figure 3.5: Schematic of the network architecture used to assess the effect of providing three orthogonal patches that captured three-dimensional context around a voxel. Details of the parameters associated with this architecture are included in Table A.2.

3.5.5 Fully Connected Layers vs. Convolutional Layers

This experiment was designed to assess how increasing the depth and internal complexity of a network affected the accuracy of its predicted segmentations. For each voxel, three orthogonal patches were used as inputs for these networks – along with information on three landmarks (the left nipple, the right nipple, and the sternal angle). The training set of voxels was obtained using the boundary sampling approach.

Between one and three fully connected layers or convolutional and max-pooling layers were arranged in series with each of the three orthogonal patches of voxel intensity values. In parallel with these, landmark information was fed through a matching number of fully connected layers arranged in series. This architecture is illustrated in Figure 3.6.

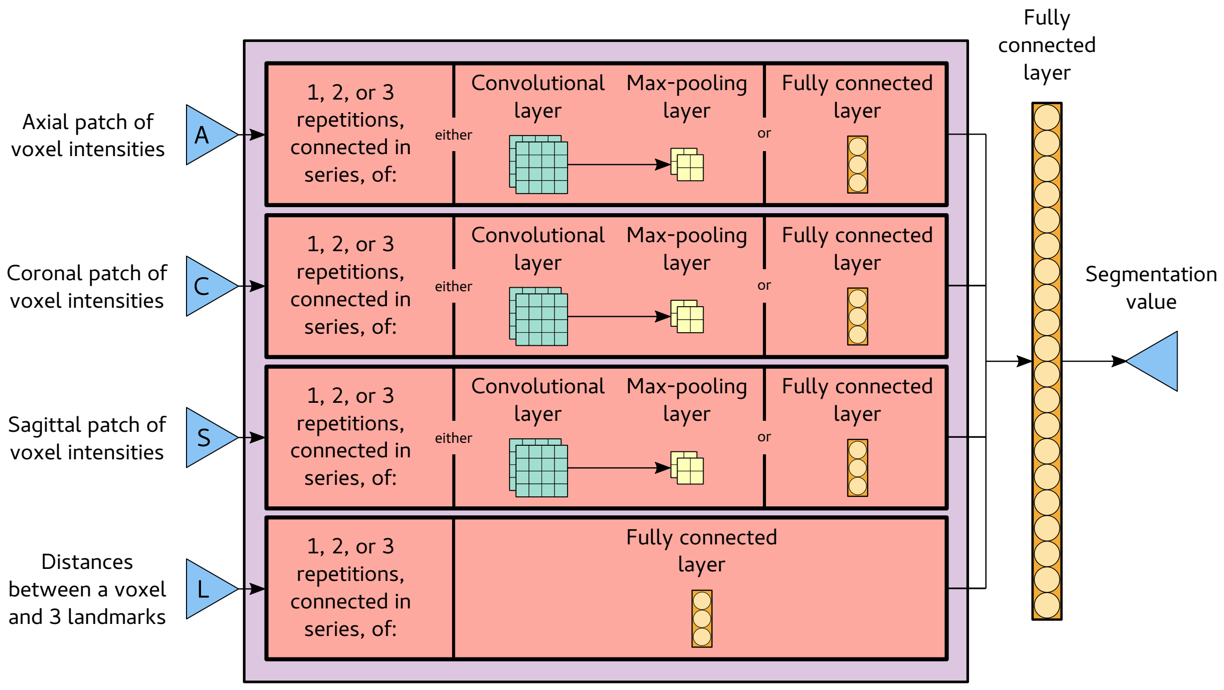

Axial patch of voxel intensities
A
Coronal patch of voxel intensities
C
Sagittal patch of voxel intensities
S
Distances between a voxel and 3 landmarks
L
1, 2, or 3 repetitions, connected in series, of:
either
Convolutional layer
Max-pooling layer
or
Fully connected layer
Fully connected layer
Segmentation value

Figure 3.6: Schematic of the network architecture used to assess the effect on segmentation accuracy of arranging different numbers of fully connected and convolutional layers in series. The full set of parameters in this architecture is provided in Table A.3.

All convolutional layers used 5 x 5 kernels. These were larger than the 3 x 3 kernels used in standard CNNs, as Karpathy (2016) noted that larger kernels often produced superior results in the domain of medical imaging. This choice also agreed with the kernels used by Lai (2015) when applying CNNs to brain segmentation.

Each convolutional layer contained a number of feature maps that doubled at each depth – an approach suggested by Simonyan and Zisserman (2014). 64 feature maps were used in the first layer, 128 in the second, and 256 in the third. Max-pooling layers all used standard 2 x 2 pooling windows (Giusti, Cireşan, Masci, Gambardella, & Schmidhuber, 2013).

The number of hidden neurons contained in each layer of the deep fully connected networks was chosen so that the number of learnable parameters in each of these models matched the number in the CNN with the same number of hidden layers. In all networks, the layers used for processing voxel patches and landmark information were merged with a single fully connected layer containing 1000 artificial neurons.

Six networks were trained and tested as part of this experiment. Each contained between one and three hidden layers that were either fully connected or convolutional and max-pooling layers arranged in series with patches of voxel intensity values.

3.5.6 Context Patches and Dropout

This experiment was conducted to assess how the addition of *context patches* and a *dropout* layer affected the accuracy of predicted segmentations. Patches taken from all three orientations, and information on three landmarks were again used as inputs. Three convolutional and max-pooling layers were connected in series with each patch, and training voxels were obtained using the boundary sampling approach.

The context patches used in this experiment were similar to the existing patches, but drawn from a larger region and downsampled to a size of 25 x 25 voxels using bilinear filtering (Wang, Wang, Wang, & Yin, 2010). These context patches provided information on a larger spatial scale than the existing patches, which were retained to provide detailed information about the immediate surroundings of a voxel.

Two different context patches were tested. One drew information from a 51 x 51 voxel area in each orientation surrounding the voxel being classified, and another from a 75 x 75 voxel area. This part of the experiment therefore only required two networks to be trained and tested, and their architectures are illustrated in Figure 3.7.

The network including 75 x 75 voxel context patches was also tested with a dropout layer. Dropout is a technique developed for DNNs by Hinton, Srivastava, Krizhevsky, Sutskever, and Salakhutdinov (2012) to limit the effects of overfitting. When the network was trained, artificial neurons and their connections were omitted from each iteration with a specific probability, and the network parameters were updated by the learning rule under the assumption that these aritifical neurons did not exist (Baldi & Sadowski, 2013). This technique forced artificial neurons to model patterns in the dataset more independently and without *co-adaptation*, where one artificial neuron relies upon others to correct its own mistakes.

The dropout layer was inserted immediately prior to the final fully connected layer used for merging voxel patches with landmark information. It was assigned a dropout probability of 0.5 – which Hinton et al. (2012) empirically found to provide the greatest improvement in testing performance.

3.5.7 Structured Prediction

The final experiment was conducted to assess how the structured prediction approach outlined in Section 2.6 affected the accuracy of predicted segmentations. 25 x 25 voxel patches were extracted from the segmentations predicted by the network with 75 x 75 voxel context patches and a dropout layer, and used as input for a secondary network. This network had an architecture identical to the one used for the previous experiment, but with context patches replaced by the 25 x 25 voxel patches that supplied

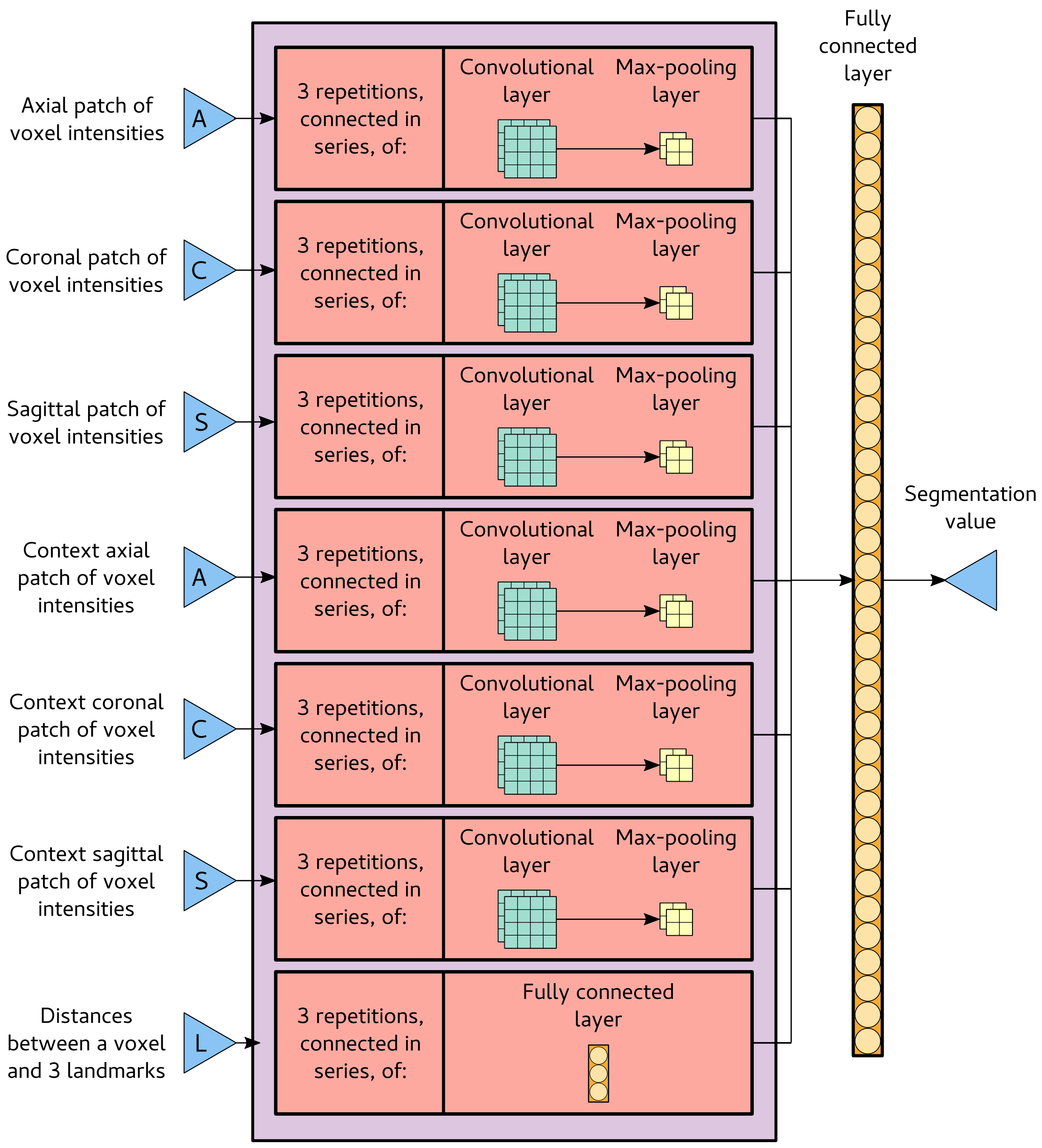

Fully connected layer
Axial patch of voxel intensities
A
3 repetitions, connected in series, of:
Convolutional layer
Max-pooling layer
Coronal patch of voxel intensities
C
3 repetitions, connected in series, of:
Convolutional layer
Max-pooling layer
Sagittal patch of voxel intensities
S
3 repetitions, connected in series, of:
Convolutional layer
Max-pooling layer
Segmentation value
Context axial patch of voxel intensities
A
3 repetitions, connected in series, of:
Convolutional layer
Max-pooling layer
Context coronal patch of voxel intensities
C
3 repetitions, connected in series, of:
Convolutional layer
Max-pooling layer
Context sagittal patch of voxel intensities
S
3 repetitions, connected in series, of:
Convolutional layer
Max-pooling layer
Distances between a voxel and 3 landmarks
L
3 repetitions, connected in series, of:
Fully connected layer

Figure 3.7: Schematic of the network architecture used to assess the effect of providing context voxel intensity patches on segmentation accuracy. Full details of all parameters used in the network are provided in Table A.4.

segmentation information. Connecting more than two networks in series was not tested – because, as well as being extremely time-consuming, Pinheiro and Collobert (2014) found that using more than this number provided no significant benefit in the context of image segmentation problems.

This approach was compared with a simple algorithm that identified connected components of voxels sharing the same segmentation values, and relabelled them to match their surroundings if they contained less than 50,000 voxels.

3.6 ANALYSIS

3.6.1 Segmentation Evaluation Metrics

Each voxel in a predicted segmentation belonged to one of four sets: true positives (TP), true negatives (TN), false positives (FP), or false negatives (FN). The predicted segmentation P was defined by $P = TP \cup FP$, and the manually labelled segmentation T by $T = TP \cup FN$.

Because the pectoral muscles occupied only a small volume in each MR image, an evaluation metric applied to this problem had to place very little weight on the rate of true negative detection. Even a predicted segmentation that classified no voxels as pectoral muscles performed well using metrics that did not satisfy this property.

The accuracy with which a segmentation delineated the boundary of the pectoral muscles was of particular importance. Taha and Hanbury (2015) identified the Hausdorff distance (Hausdorff, 1927), H, as being well-suited for measuring the distance between the surfaces of predicted and manually labelled segmentations. This metric is defined in Equation (8), where the l_2 norm is used as the distance measure.

$$H(P,T) = \max\{\max_{p \in P} \min_{t \in T} l_2(p,t), \max_{t \in T} \min_{p \in P} l_2(t,p)\} \tag{8}$$

Although it provided a good evaluation of accuracy at the pectoral muscle boundary, the Hausdorff distance was very sensitive to spurious artefacts in predicted segmentations. Two additional measures that did not suffer from this problem were used to provide less precise measurements of segmentation accuracy. The first was the Dice coefficient (Dice, 1945), D, which is a dimensionless measure of volume overlap – and by far the metric most commonly used for evaluating medical image segmentations. This metric is defined in Equation (9).

$$D(P,T) = \frac{2|P \cap T|}{|P| + |T|} \tag{9}$$

The Mahalanobis distance (McLachlan, 1999), M, is a dimensionless, distance based measure that accounted for the general shape of a segmentation – but was less precise than the Hausdorff distance (Taha & Hanbury, 2015). This metric is defined in Equation (10):

$$M(P,T) = \sqrt{(\mu_P - \mu_T)^T S^{-1} (\mu_P - \mu_T)} \tag{10}$$

where μ_P and μ_T are the mean position vectors of voxels in the predicted and manual segmentations, and S (defined in Equation (11)) is the common covariance matrix of these voxels in terms of their individual covariance matrices, S_P and S_T.

$$S = \frac{|P|S_P + |T|S_T}{|P| + |T|} \tag{11}$$

3.6.2 Comparison of Network Performance

Each experiment required a set of networks to be compared based on their performance over all MR volumes in the testing set according to each metric. This first required a multiple comparison test to identify any statistically significant differences in network performance. Comparing networks based on their mean performance over all volumes was inadequate, because some volumes were inherently more difficult than others to segment. Performance measures were therefore incommensurable across different volumes – which was not considered by a comparison of means.

The performances of networks on each volume were not normally distributed random variables with constant variance. The non-parametric Friedman test is robust to these conditions, and Demšar (2006) found it to provide more statistical power than the repeated-measures ANOVA test in this situation. The Friedman test was therefore selected for the multiple comparison tests required by each experiment.

If the Friedman test detected a significant difference in network performance, post hoc tests were conducted to determine which pairs of networks caused it. Garcia and Herrera (2008) found the post hoc test of Bergmann and Hommel (1988) to be the most powerful for this application, and it was therefore used when assessing networks.

Statistically significant differences between networks also needed to be quantified. Because the data did not satisfy the assumptions of normality or constant variance, the robust Wilcoxon signed-rank test (Wilcoxon, 1945) was used to compare each pair of networks in an experiment. This test provided an estimate of the median difference in a performance measure achieved by each network on the same testing volume.

3.6.3 Testing Convergence of Network Training

Use of the hybrid training approach described in Section 3.4.1 resulted in a distinct profile to the value of the loss function as it evolved over the training process. Whenever a new group of voxels was presented to the network, the training loss increased sharply. As the network parameters were adjusted while training on this data, the training loss decreased smoothly – before increasing once again when a new group of training voxels was retrieved. This effect is depicted in Figure 3.8.

As a network was trained on more groups of voxels, the value of the loss function prior to the start of a new group decreased. Because these values provided a lower bound on the value of the loss function when the network was presented with a new group of voxels, they were used to test for convergence during training.

A standard convergence test was used, whereby a network was deemed to have converged when the smallest training loss achieved prior to a new group of training voxels being retrieved had not decreased across 20 consecutive groups.

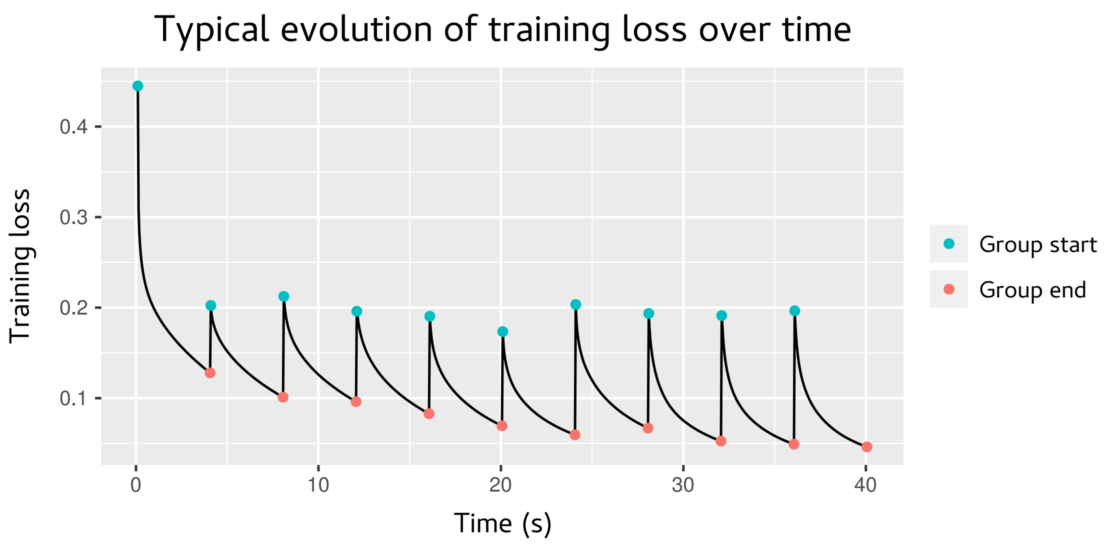


Figure 3.8: Typical behaviour of the network loss function as training proceeded. The *spikes* in training loss correspond to a new group of voxels being retrieved from the set of training datapoints.

3.7 IMPLEMENTATION

3.7.1 Software

All networks were implemented in Python. The bulk of mathematical processing was performed by tools contained in the *Theano* package (Theano Development Team, 2016) – which allowed mathematical expressions involving multi-dimensional arrays to be defined, compiled, and computed using a GPU.

The architectures of each network were built using the *Lasagne* library (Lasagne Development Team, 2016). This tool provided *Theano* interfaces for a selection of ANN layers, loss functions and learning rules. Training was carried out using the *nolearn* library (Nouri, 2014), which provided a layer of abstraction over *Lasagne* for the training process.

Segmentations were evaluated using the *EvaluateSegmentation* tool developed by Taha and Hanbury (2015). This provided an efficient C++ implementation of a variety of metrics commonly used to assess the quality of medical image segmentations. Statistical tests for comparing network performance were carried out in *R* using the Friedman, Bergmann-Hommel, and Wilcoxon tests provided in the *scmamp* package (Calvo & Santafe, 2015).

All code written for processing MR image data, and training and testing networks is available at github.com/sgilm/pecdeeplearn.

3.7.2 Hardware

All experiments were conducted on the high performance computing clusters maintained by New Zealand eScience Infrastructure (NeSI) (nesi.org.nz). Each used an Intel E5-2680 2.70 GHz processor with 64 GB RAM and an NVIDIA K20X GPU with 6 GB memory.

4 RESULTS AND DISCUSSION

Six experiments were conducted over the course of the investigation. This chapter presents the results of each, and discusses how they were used to gradually refine a network architecture into one that was able to accurately segment the pectoral muscles. All results were analysed using the approaches detailed in Section 3.6.2. If one network architecture has been noted to provide an improvement over another, this improvement was found to be statistically significant at the 95% confidence level under the Friedman and Bergmann-Hommel tests. The Wilcoxon test was used to estimate median performance differences between pairs of networks, with these estimates also being valid for image volumes not present in the testing set. Median performance measures across all segmentations predicted by a network have also been included to provide numerical context.

4.1 WITHOUT LANDMARKS VS. WITH LANDMARKS

The first experiment was conducted to assess how providing landmark information as a network input affected the accuracy of predicted segmentations. The architectures of each network are included in Section 3.5.2, and full results in Tables B.1 and B.2.

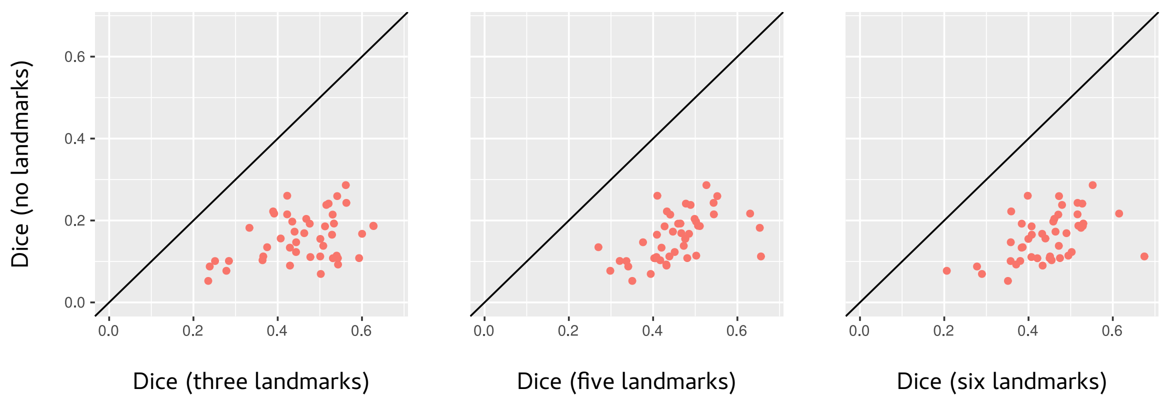
According to every metric, the three networks that included landmark information outperformed the network that did not. This effect is illustrated in Figure 4.1. Aggregated across all patch orientations and landmark sets, the inclusion of landmark information resulted in large improvements to segmentation accuracy:

- A median increase in Dice coefficient of between 0.25 and 0.36, with the median Dice coefficient across all predicted segmentations increasing from 0.16 to 0.46.
- A median decrease in Mahalanobis distance of between 0.34 and 0.59, with the median Mahalanobis distance across all predicted segmentations decreasing from 0.97 to 0.48.

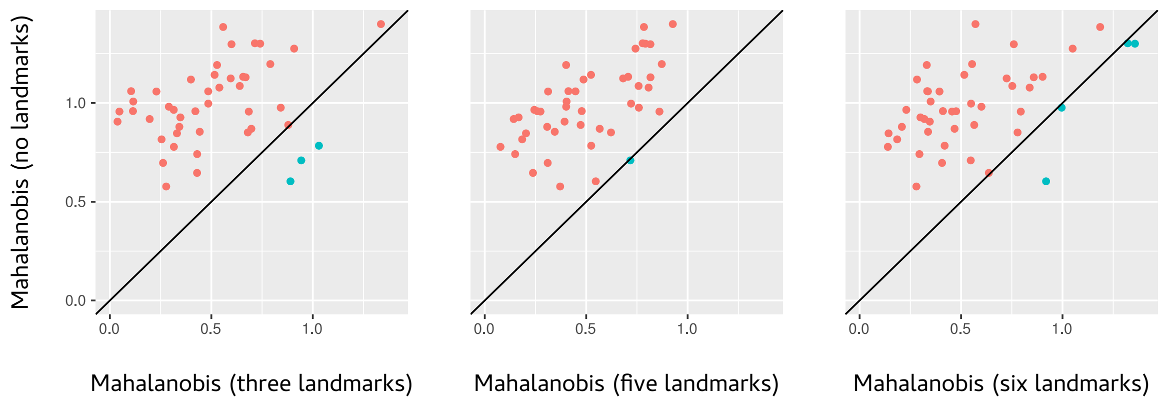
Although the sensitivity of the Hausdorff distance to spurious artefacts in a segmentation meant it was not the most suitable metric to use at this stage, a large median decrease in Hausdorff distance of between 127.9 mm and 140.7 mm was also observed.

Compared to the networks based on three landmarks (the two nipples and the sternal angle), those networks that included additional landmarks (the two humerus balls and the spinal cord) showed no statistically significant differences according to any metric. This is unsurprising – the nipples and the sternal angle represented the landmarks closest to the breast region, and providing any additional information about voxel position relative to the more distant humerus balls or spinal cord was redundant.

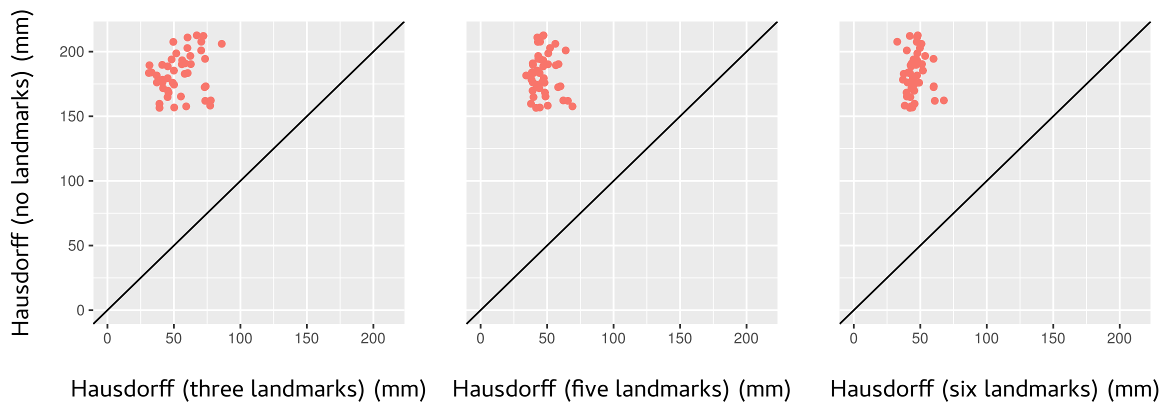
Figure 4.2 provides a representative comparison of segmentations predicted by the network that included no landmark information and the network that included infor-



(a) Comparison based on Dice coefficients.

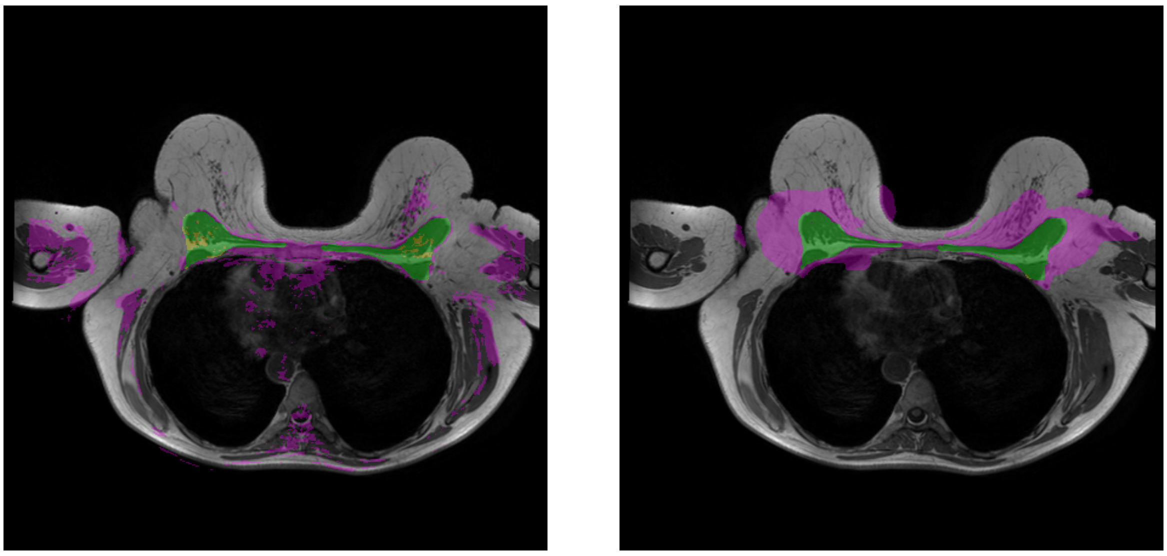


(b) Comparison based on Mahalanobis distances.



(c) Comparison based on Hausdorff distances.

Figure 4.1: Performance comparison of the networks that were provided with landmark information and the network that was not. Network performance in each metric is aggregated across all patch orientations. Red datapoints denote individual image volumes for which the networks that were provided with landmark information outperformed the network that was not, and cyan datapoints represent the converse cases.



(a) No landmarks. **(b)** Three landmarks.

Figure 4.2: Axial slices of segmentations performed by networks with no landmark information, and information about three landmarks. Green voxels denote true positives, purple voxels denote false positives, and yellow voxels denote false negatives.

mation on three landmarks. Both networks correctly classified almost all voxels corresponding to pectoral muscles in the manual segmentation. However, the network provided with no landmark information predicted a segmentation that was heavily dependent on voxel intensity values – and, as a result, bled into the fibroglandular breast tissue, heart, and muscular tissue extending around the posterior of the ribcage.

On the other hand, the network that was provided with information on three landmarks used this data to more effectively identify the general location of the pectoral muscles. It was, however, less precise at the pectoral muscle boundary than the network that was provided with no landmark information.

While the segmentations predicted by networks that were provided with information on landmarks were imprecise at the pectoral muscle boundary, they did more accurately identify the general location of the pectoral muscles. The network provided with information on three landmarks represented the least complex of the landmark networks – and was therefore selected to be refined in the next experiment.

4.2 BALANCED SAMPLING VS. BOUNDARY SAMPLING

Segmentations predicted by the networks that were provided with information on three landmarks contained many false positives on the exterior side of the pectoral muscle boundary. This issue can be explained by the sampling method used to obtain the voxels in training dataset. Under the balanced sampling approach (defined in Section 3.3.1) that was used in the previous experiment, the small volume occupied by the pectoral muscles relative to an entire MR image meant that many of the points sampled from the pectoral muscles were located near the interior of the boundary. However, the points sampled from outside the pectoral muscles were drawn from a far larger set – and almost none were located near the exterior of the boundary.

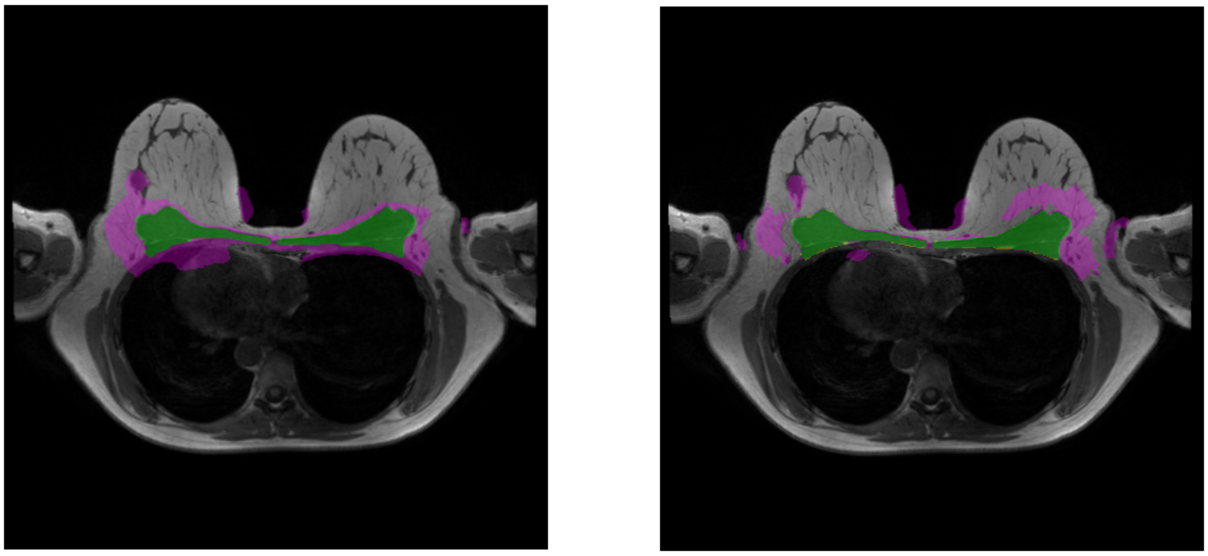

(a) Balanced sampling. (b) Boundary sampling.

Figure 4.3: Axial slices of segmentations performed on the same MR image by networks trained using the balanced or boundary sampling approaches. Green voxels denote true positives, purple voxels false positives, and yellow voxels false negatives.

This experiment was conducted to assess the effect of the boundary sampling approach outlined in Section 3.3.2, which ensured networks were explicitly trained on voxels falling on both sides of the pectoral muscle boundary. The networks used are detailed in Section 3.5.3, and complete results are provided in Tables B.3 and B.4. Because this experiment specifically aimed to assess segmentation overlap at the pectoral muscle boundary, the Dice coefficient was used to compare networks. It was selected in preference to both the shape-sensitive Mahalanobis distance, which was similar between two segmentations with varying degrees of precision at the boundary, and the surface distance-based Hausdorff distance, which was extremely sensitive to the spurious artefacts that remained present in segmentations at this stage.

Across networks using each of the three orthogonal patch orientations, the use of the boundary sampling approach to select voxels for the training dataset resulted in an incremental median increase in Dice coefficient of between 0.0048 and 0.072 – with the median Dice coefficient across all predicted segmentations increasing from 0.48 to 0.53. Figure 4.3 shows a visual comparison of representative segmentations produced by two networks with identical architectures that were trained using the different sampling methods. While using the boundary sampling approach still produced segmentations with large spurious artefacts, they were visibly more precise along large sections of the pectoral muscle boundary and lacked many of the false positive predictions produced by networks using the balanced sampling approach.

Figure 4.4 shows how using the boundary sampling approach increased the training loss observed as the learning rule updated network parameters. This effect was observed because the network was fed with more points drawn from the difficult-to-classify boundary region. Even so, training time did not increase above 15 minutes.

Because the balanced sampling approach reduced the error of predicted segmentations at the pectoral muscle boundary, it was used for all subsequent experiments.

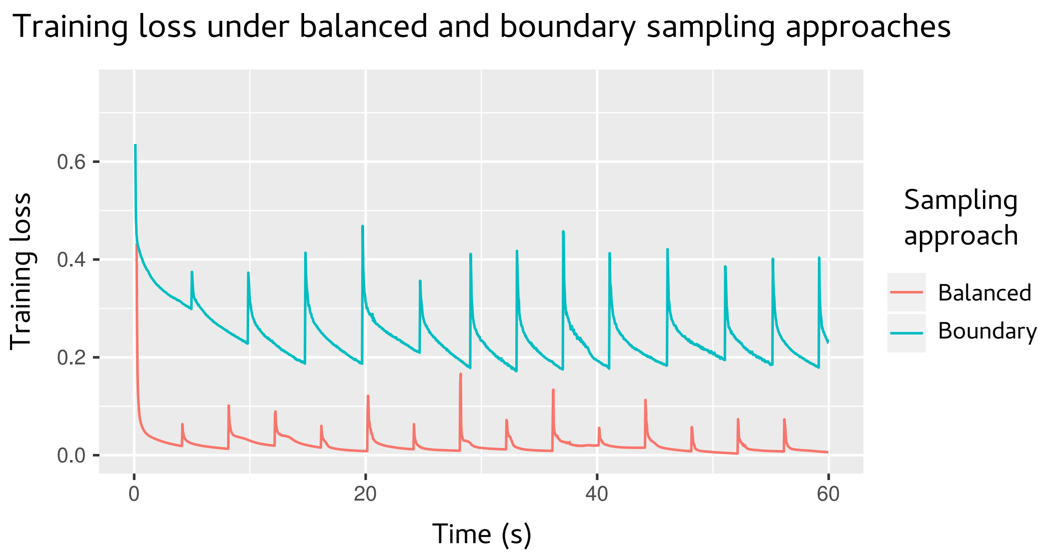


Figure 4.4: Values of the loss function as training proceeded, using either the balanced or boundary sampling approach.

4.3 SINGLE PATCH VS. MULTIPLE PATCHES

This experiment investigated how network performance responded to the inclusion of patches drawn from different orientations, as well as combining three orthogonal patches that provided three-dimensional context around a voxel. The networks used are detailed in Section 3.5.4, and complete results are included in Tables B.5 and B.6.

This experiment was primarily concerned with assessing the overall shape of a predicted segmentation. The Mahalanobis distance was therefore the most appropriate metric for comparing networks, and was selected to be used at this stage.

When fewer artificial neurons were used in the hidden layer of each network (986 for the single patch networks, and 332 for the network with all three orthogonal patches), there were no significant differences in the Mahalanobis distances of the axial, coronal, or sagittal networks – nor the network that included patches from all three orientations. Even though the data supplied to the network based on all three orthogonal patches contained three-dimensional context around a voxel, and was therefore more descriptive than a single patch, the complexity of the network was not sufficient to allow the data to be adequately represented.

When the number of neurons in the hidden layer was increased (to 8854 for the single patch networks, and 3000 for the network with all three orthogonal patches), the network with three orthogonal patches modelled the additional information it was provided with far more effectively. There was a large improvement in accuracy compared to each network that included only a single patch, by the following amounts:

- A median decrease in Mahalanobis distance compared to the network including a single axial patch of 0.13, with the median Mahalanobis distance across all predicted segmentations decreasing from 0.66 to 0.44.
- A median decrease in Mahalanobis distance compared to the network including a single coronal patch of 0.37, with the median Mahalanobis distance across all predicted segmentations decreasing from 0.77 to 0.44.

- A median decrease in Mahalanobis distance compared to the network including a single sagittal patch of 0.17, with the median Mahalanobis distance across all predicted segmentations decreasing from 0.61 to 0.44.

These improvements are shown in Figure 4.5. The network that included three orthogonal patches and more artificial neurons in the hidden layer also offered a significant improvement over the network with three orthogonal patches and fewer neurons in the hidden layer. This improvement was by a median decrease in Mahalanobis distance of 0.12, with the median Mahalanobis distance across all predicted segmentations decreasing from 0.53 to 0.44.

Because the network that included three patches of voxel intensities and more neurons in the hidden layer produced the most accurate predicted segmentations, this network architecture was selected to be refined in the next experiment.

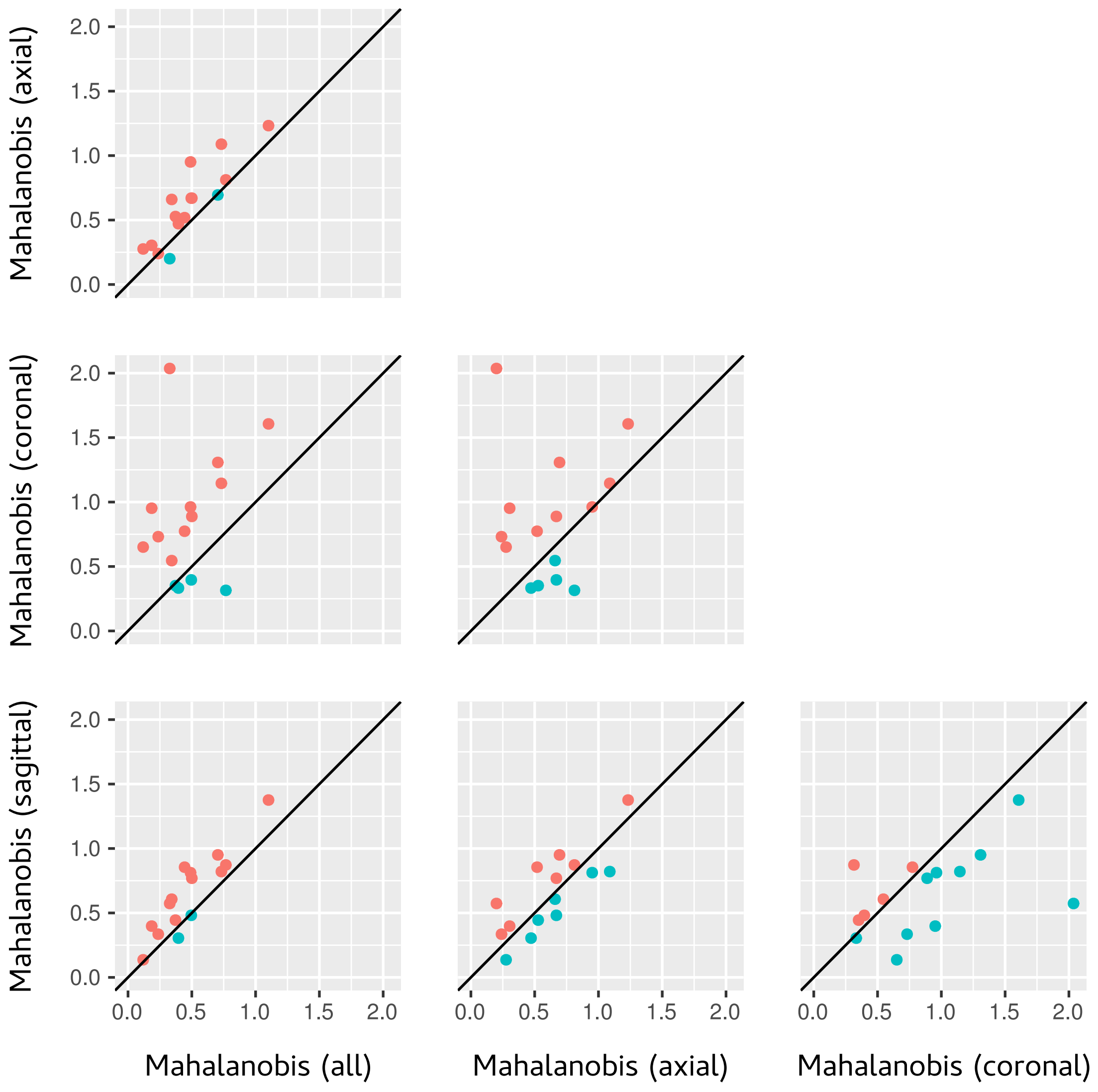


Figure 4.5: Pairwise comparison of network performance across individual patch orientations and all three orthogonal patches, with an increased number of neurons being used in the hidden layer. Red datapoints denote individual image volumes on which the network on the horizontal axis outperformed the network on the vertical axis, and cyan datapoints represent the converse cases.

4.4 FULLY CONNECTED LAYERS VS. CONVOLUTIONAL LAYERS

This experiment focused on investigating two additional aspects of network architecture – the number of hidden layers connected in series, and the use of convolutional layers. The networks used in this experiment are detailed in Section 3.5.5, and full results are presented in Tables B.7 and B.8.

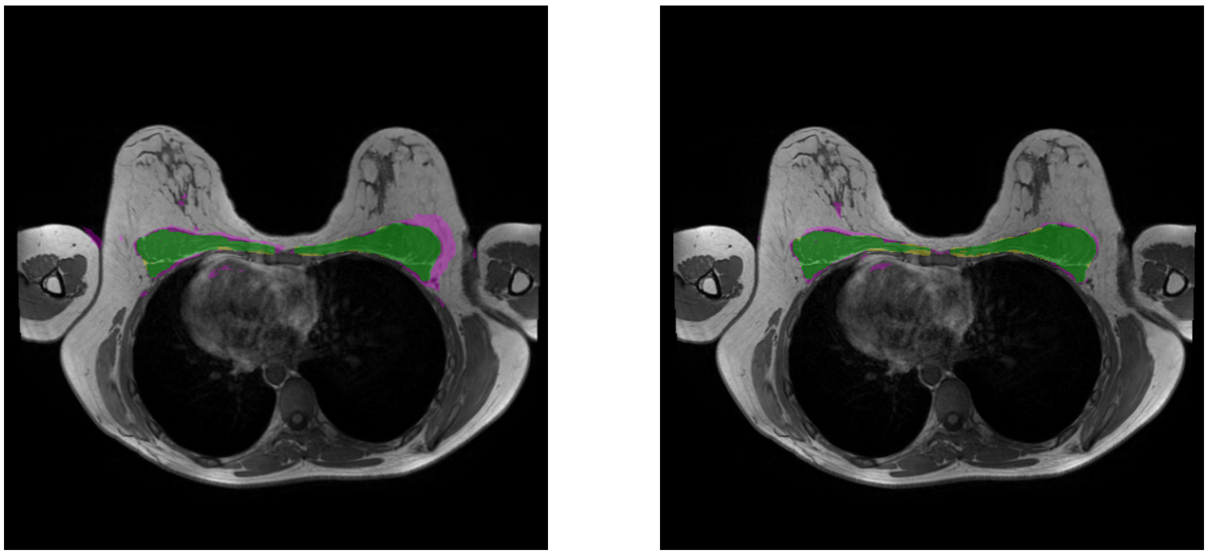
The most accurate segmentations predicted by networks in this experiment contained fewer spurious artefacts. As a result, the Hausdorff distance was considered to be more appropriate than the Mahalanobis distance for comparing networks.

The networks with fully connected hidden layers produced segmentations that remained relatively unchanged in accuracy as the depth of hidden layers was increased. On the other hand, the CNNs showed small-to-moderate performance improvements according to the Hausdorff distance as the number of hidden layers was increased to three. The improvements offered by connecting three convolutional and max-pooling layers in series were as follows:

- A median decrease in Hausdorff distance over the network that included one convolutional and max-pooling layer of 11.3 mm, with the median Hausdorff distance across all predicted segmentations decreasing from 68.7 mm to 55.2 mm.
- A median decrease in Hausdorff distance over the network including two convolutional and max-pooling layers of 6.5 mm, with the median Hausdorff distance across all predicted segmentations decreasing from 59.7 mm to 55.2 mm.
- A median decrease in Hausdorff distance over the network with three fully connected layers of 3.6 mm, with the median Hausdorff distance across all predicted segmentations decreasing from 60.0 mm to 55.2 mm.

Figure 4.6 provides a representative visual comparison of segmentations produced by these two network architectures. Both networks correctly classified almost all voxels that were labelled as pectoral muscles in the manual segmentation. The improved performance offered by the deep network with three convolutional and max-pooling layers was due to its ability to reduce false positive classifications.

The CNN with three hidden layers achieved its improved performance with approximately 2,000,000 learnable parameters – at least 3 times fewer than any other network tested as part of this experiment. This reduction in the required number of parameters was due to the additional max-pooling layers that downsampled information passed between successive convolutional layers – which allowed initial convolutional layers to identify low-level features and subsequent layers to detect progressively higher-level features on larger spatial scales in the original input patches. The increased number of convolution operations performed by the network meant that it was able to effectively model the dataset without requiring a large set of learnable parameters. However, the CNNs all required far greater training times than the fully connected networks. Figure 4.7 compares the training times of each network architecture, and shows that the network that included three convolutional and max-pooling layers required around 10 hours before converging – 40 times greater than for a fully connected network with



(a) Fully connected layers. (b) Convolutional layers.

Figure 4.6: Axial slices of segmentations performed by fully connected and convolutional networks with three hidden layers. Green voxels denote true positives, purple voxels denote false positives, and yellow voxels denote false negatives.

the same number of hidden layers. Although training times of the CNNs increased as more layers were added (because they performed more convolution operations), the training time of the fully connected networks decreased (due to the presence of fewer learnable parameters).

The network with three convolutional and max-pooling layers produced segmentations that were more accurate than any other network tested as part of this experiment. While it required more time to converge during training than the network with three fully connected hidden layers, it was still able to do so in a reasonable period. It was therefore selected as the network to be refined in the next experiment.

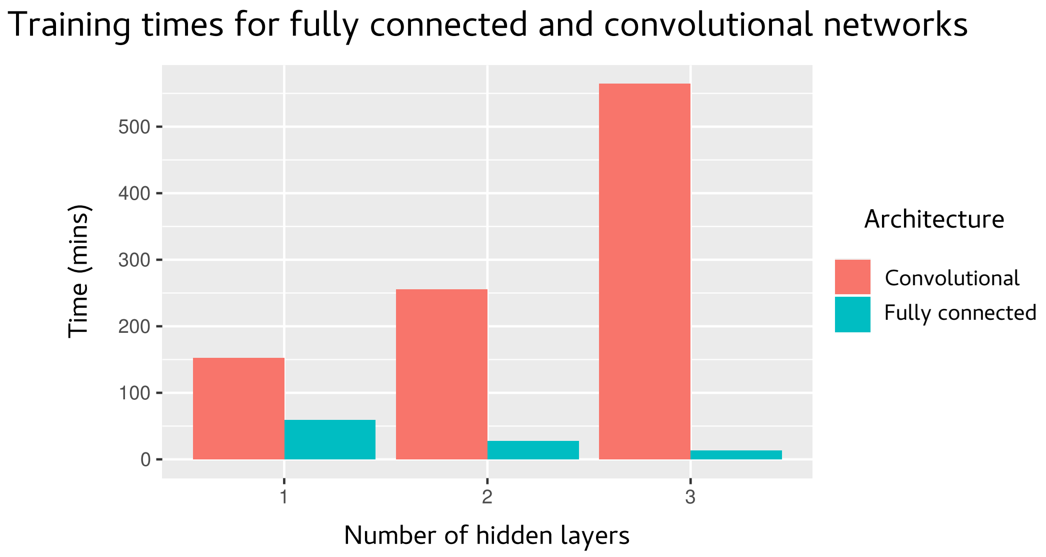


Figure 4.7: Time required for convergence during training of both fully connected and convolutional networks.

4.5 CONTEXT PATCHES AND DROPOUT

This experiment was conducted to assess how including dropout layers and voxel intensity values from a larger spatial scale in a MR image volume influenced the accuracy of predicted segmentations. The network architectures used are described in Section 3.5.6, and complete results are included in Tables B.9 and B.10.

Both networks that included context patches achieved slightly improved performance in Hausdorff distance over the network without them – with a median decrease in Hausdorff distance of between 5.4 mm and 6.5 mm being observed. When compared with each other, the two networks with context patches performed similarly on this metric – as illustrated in Figure 4.8. While providing context patches did improve the accuracy of predicted segmentations, it also resulted in training times that, at 23 hours, were more than twice as large as the network without them.

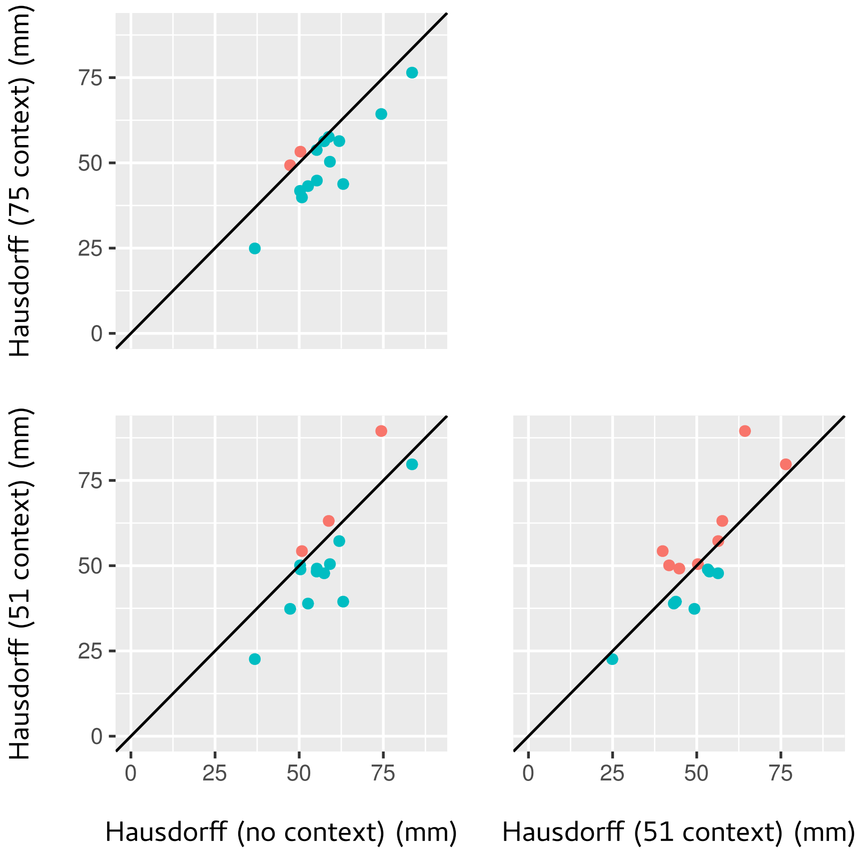


Figure 4.8: Pairwise comparison of network performance according to the Hausdorff distance metric when using different context patches. Red datapoints denote individual image volumes for which the network on the horizontal axis outperformed the network on the vertical axis, and cyan datapoints represent the converse cases.

The next stage of this experiment tested the influence of a dropout layer on network performance – in response to potential overfitting. Because both networks with context patches offered similar performance and had the same complexity (the only difference being the area from which the context patch was drawn), the network including the 75 x 75 voxel context patch was arbitrarily selected for this test.

The network that included a dropout layer showed slightly improved performance over the network that only included context patches, as measured by the Hausdorff distance. A median decrease in Hausdorff distance of 6.6 mm was observed, with the median Hausdorff distance across all segmentations decreasing from 49.2 mm to 45.5 mm. While statistically significant, this improvement was small – and suggested that overfitting was not a prevalent problem in the networks previously tested.

As the best performing network at this stage, the segmentations produced by the network that included 75 x 75 voxel context patches and a dropout layer were selected to be used in the final experiment related to structured prediction.

4.6 STRUCTURED PREDICTION

This experiment was conducted to test the effect of structured approaches that gave explicit consideration to the segmentation values of neighbouring voxels when making a prediction. Both of the approaches detailed in Section 3.5.7 were tested on the segmentations predicted by the network from the previous experiment that included 75 x 75 voxel context patches and a dropout layer.

The connected component approach achieved incremental improvements according to both the Hausdorff distance and Dice coefficient metrics. These improvements were not due to appreciable changes in the shape of segmentations – but rather as a result of small spurious artefacts being removed. The following improvements were observed:

- A median increase in Dice coefficient of 0.012, with the median Dice coefficient across all segmentations increasing from 0.68 to 0.69.
- A median decrease in Hausdorff distance of 4.6 mm, with the median Hausdorff distance across all segmentations decreasing from 45.5 mm to 43.0 mm.

Introducing a secondary network resulted in a large improvement in Hausdorff distance, but a moderate decrease in Dice coefficient by the following amounts:

- A median decrease in Dice coefficient of 0.050, with the median Dice coefficient across all segmentations decreasing from 0.68 to 0.63.
- A median decrease in Hausdorff distance of 9.4 mm, with the median Hausdorff distance across all segmentations decreasing from 45.5 mm to 33.1 mm.

While the secondary network effectively removed spurious segmentation artefacts (resulting in reduced Hausdorff distances), it also led to a loss of precision and more false positives being identified at the pectoral muscle boundary (resulting in reduced Dice coefficients). Given the importance of delineating the surface of the pectoral muscles, the connected component algorithm represented a more suitable approach to structured prediction than the introduction of a secondary network.

Comparing these final results with other studies had limited significance – given that the image volumes used to assess performance were not held constant, and that other studies often sought to segment the pectoral muscles using different imaging modalities. However, the results obtained in this investigation were competitive when compared with the the atlas-based approach to MR image segmentation developed by Gubern-Mérida et al. (2012) – which achieved a mean Dice coefficient of 0.72.

5 CONCLUSIONS AND FUTURE WORK

The objective of this investigation was to develop a method for automatically segmenting the pectoral muscles within a breast MR image. ANNs were selected as the technique to be investigated – due to their success within the domain of brain segmentation and their ability to automate the feature selection process.

The investigation started by building an ANN model with relatively little complexity, and proceeded to refine the network architecture in response to issues observed in the predicted segmentations at each stage. Finally, a structured prediction approach, which explicitly took into account the segmentation values predicted by neighbouring voxels, was tested. The following conclusions were drawn from these experiments:

- Providing landmark information to networks greatly enhanced their ability to localise the pectoral muscles. Information on the position of a voxel relative to landmarks other than the left nipple, right nipple, and sternal angle was redundant.
- Explicitly training a network on voxels obtained from the neighbourhood of the pectoral muscle boundary improved its ability to identify the boundary contour.
- Providing a network with intensity values drawn from three dimensions around a voxel significantly improved the accuracy of network predictions, but only after internal complexity was added to the network.
- A DNN with three convolutional and max-pooling layers connected in series provided the greatest improvement in performance when compared with other deep networks. However, this architecture required a significantly greater period of time to converge during the training process.
- Supplementing standard voxel intensity patches with context patches drawn from a larger area, and then downsampled, incrementally improved the accuracy of network predictions.
- A structured prediction approach that identified connected components in a segmentation removed spurious segmentation artefacts and incrementally improved accuracy. An approach that connected two networks in series also removed spurious segmentation artefacts, but resulted in a significant loss of precision at the pectoral muscle boundary.

Each of these improvements were included in the network architecture that produced the most accurate predicted segmentations across all MR image volumes in the testing set. This network achieved a median Dice coefficient of 0.69 and median Hausdorff distance of 43.0 mm. While not state of the art, these results are competitive when compared with other methods used to approach the same problem (Gubern-Mérida et al., 2012).

There are a number of avenues for future investigation. The use of a patch-wise training approach was necessitated by the lack of a computationally efficient 3D convolution implementation. Ongoing research in this area – especially as interest in three-dimensional datasets such as video and medical image volumes grows – will likely ensure that one becomes available in the near future. This would allow an image-wise training approach to be tested.

Conditional random fields (CRFs) are another approach to structured prediction that have been combined with ANNs and applied to two-dimensional image segmentation. While preliminary tests using CRFs were conducted in this investigation, the approach was found to be computationally intractable when applied to a three-dimensional image volume. Further work may focus on developing more efficient ways of applying this technique, perhaps by reducing the size of the three-dimensional dataset with preliminary supervoxel segmentation approaches.

Even where one network outperformed another in this investigation, the highly variable characteristics of breast MR image volumes meant that networks that performed worse in general occasionally achieved improved performance on individual volumes. An ensemble approach to segmentation – where multiple networks are combined to segment each datapoint (Dietterich, 2000) – may therefore be useful for this application.

The ultimate goal of the Biomechanics for Breast Imaging Group at the ABI is to automatically construct computational models of the breast from MR image volumes. The approach developed as part of this investigation achieves good accuracy when segmenting the pectoral muscles – but perhaps most importantly, has the potential to be adapted for segmentation of multiple tissue types. This will allow different anatomic structures in the breast to be explicitly modelled, and increasingly accurate models to be constructed that have clinically useful applications for breast cancer imaging.

APPENDICES

A NETWORK ARCHITECTURE PARAMETERS

This appendix provides the parameters for each network architecture that was tested as part of this investigation. All patches were 25 x 25 voxels in size, and all artificial neurons were equipped with the rectifier activation function. If network architectures for a particular experiment do not appear in this appendix, it is because they were identical to those used in another experiment. Details are provided in section 3.5.

Table A.1: Parameters selected for network architectures used to assess the effect of providing landmark information on the accuracy of predicted segmentations.

Architecture	Layer	Property	Value
Zero landmarks	Input layer	Number of artifical neurons	625
	Hidden layer	Number of artificial neurons	1000
	Overall	Total learnable parameters	627001
Three landmarks	Input layer	Number of artificial neurons	634
	Hidden layer	Number of artificial neurons	986
	Overall	Total learnable parameters	627097
Five landmarks	Input layer	Number of artificial neurons	640
	Hidden layer	Number of artificial neurons	978
	Overall	Total learnable parameters	627877
Six landmarks	Input layer	Number of artificial neurons	643
	Hidden layer	Number of artificial neurons	972
	Overall	Total learnable parameters	626941

Table A.2: Parameters selected for network architectures used to assess the effect of providing three orthogonal voxel intensity patches from the area surrounding a voxel.

Architecture	Layer	Property	Value
Fewer artificial neurons (single patch)	Input layer	Number of artifical neurons	634
	Hidden layer	Number of artificial neurons	986
	Overall	Total learnable parameters	627097
Fewer artificial neurons (three patches)	Input layer	Number of artificial neurons	1884
	Hidden layer	Number of artificial neurons	332
	Overall	Total learnable parameters	626153
More artificial neurons (single patch)	Input layer	Number of artificial neurons	634
	Hidden layer	Number of artificial neurons	8854
	Overall	Total learnable parameters	5631145
More artificial neurons (three patches)	Input layer	Number of artificial neurons	1884
	Hidden layer	Number of artificial neurons	3000
	Overall	Total learnable parameters	5658001

Table A.3: Parameters selected for network architectures used to assess the effect of deep networks that included between one and three fully connected or convolutional and max-pooling layers, arranged in series with patches of voxel intensity values.

Architecture	Layer	Property	Value
One fully connected layer	Input layer	Number of artifical neurons	1884
	Patch layers	Number of artificial neurons	4762
	Landmark layers	Number of artificial neurons	100
	Merging layer	Number of artificial neurons	1000
	Overall	Total learnable parameters	23532237
Two fully connected layers	Input layer	Number of artificial neurons	1884
	Patch layers	Number of artificial neurons	855
	Landmark layers	Number of artificial neurons	100
	Merging layer	Number of artificial neurons	1000
	Overall	Total learnable parameters	6699831
Three fully connected layers	Input layer	Number of artificial neurons	1884
	Patch layers	Number of artificial neurons	535
	Landmark layers	Number of artificial neurons	100
	Merging layer	Number of artificial neurons	1000
	Overall	Total learnable parameters	2240196
One convolutional/max-pooling layer	Input layer	Number of artificial neurons	1884
	First patch layer	Kernel size	3 x 3
		Number of filters	64
		Max-pooling partition size	2 x 2
	Landmark layers	Number of artificial neurons	100
	Merging layer	Number of artificial neurons	1000
	Overall	Total learnable parameters	23537121
Two convolutional/max-pooling layers	Input layer	Number of artificial neurons	1884
	First patch layer	Kernel size	3 x 3
		Number of feature maps	64
		Max-pooling partition size	2 x 2
	Second patch layer	Kernel size	3 x 3
		Number of feature maps	128
		Max-pooling partition size	2 x 2
	Landmark layers	Number of artificial neurons	100
	Merging layer	Number of artificial neurons	1000
	Overall	Total learnable parameters	6700989
Three convolutional/max-pooling layers	Input layer	Number of artificial neurons	1884
	First patch layer	Kernel size	3 x 3
		Number of feature maps	64
		Max-pooling partition size	2 x 2
	Second patch layer	Kernel size	3 x 3
		Number of feature maps	128
		Max-pooling partition size	2 x 2
	Third patch layer	Kernel size	3 x 3
		Number of feature maps	256
		Max-pooling partition size	2 x 2
	Landmark layers	Number of artificial neurons	100
	Merging layer	Number of artificial neurons	1000
	Overall	Total learnable parameters	2240793

Table A.4: Parameters selected for network architectures used to test the effect of context patches and dropout layers on the accuracy of predicted segmentations.

Architecture	Layer	Property	Value
Context patches	Input layer	Number of artifical neurons	3759
	First patch layer	Kernel size	3 x 3
		Number of feature maps	64
		Max-pooling partition size	2 x 2
	Second patch layer	Kernel size	3 x 3
		Number of feature maps	128
		Max-pooling partition size	2 x 2
	Third patch layer	Kernel size	3 x 3
		Number of feature maps	256
		Max-pooling partition size	2 x 2
	Landmark layers	Number of artificial neurons	100
	Merging layer	Number of artificial neurons	1000
	Overall	Total learnable parameters	4117785
Dropout	Input layer	Number of artificial neurons	3759
	First patch layer	Kernel size	3 x 3
		Number of feature maps	64
		Max-pooling partition size	2 x 2
	Second patch layer	Kernel size	3 x 3
		Number of feature maps	128
		Max-pooling partition size	2 x 2
	Third patch layer	Kernel size	3 x 3
		Number of feature maps	256
		Max-pooling partition size	2 x2
	Landmark layers	Number of artificial neurons	100
	Dropout layer	Dropout probability	0.5
	Merging layer	Number of artificial neurons	1000
	Overall	Total learnable parameters	4117785

B NETWORK PERFORMANCE RESULTS

This appendix provides all data and analysis from the investigation. Tables of median differences in performance (obtained using the Wilcoxon signed-rank test) imply the column architecture being subtracted from the row architecture. Green cells indicate differences that were statistically significant with 95% confidence under the Bergmann-Hommel test. For example – the first result in table B.1 shows median differences in the Dice coefficient between networks using axial patches of voxel intensities and different subsets of landmark information. Statistically significant differences occurred between the network with no landmarks and each of the landmark networks. The network with information on three landmarks predicted segmentations with Dice coefficients a median 0.2552 greater than the network with no landmarks.

Table B.1: Median differences in performance between pairs of architectures with information about different landmark sets, grouped by metric and patch orientation.

axial (dice)	0_landmarks	3_landmarks	5_landmarks	6_landmarks
0_landmarks		-0.2552	-0.2920	-0.2654
3_landmarks			-0.0269	0.0026
5_landmarks				0.0366
6_landmarks				

coronal (dice)	0_landmarks	3_landmarks	5_landmarks	6_landmarks
0_landmarks		-0.3598	-0.3263	-0.3318
3_landmarks			0.0365	0.0448
5_landmarks				-0.0005
6_landmarks				

sagittal (dice)	0_landmarks	3_landmarks	5_landmarks	6_landmarks
0_landmarks		-0.2952	-0.2459	-0.2609
3_landmarks			0.0518	0.0366
5_landmarks				-0.0043
6_landmarks				

axial (mahalanobis)	0_landmarks	3_landmarks	5_landmarks	6_landmarks
0_landmarks		0.5866	0.5600	0.5668
3_landmarks			-0.0041	-0.0229
5_landmarks				0.0056
6_landmarks				

coronal (mahalanobis)	0_landmarks	3_landmarks	5_landmarks	6_landmarks
0_landmarks		0.4345	0.4324	0.3435
3_landmarks			0.0002	-0.0805
5_landmarks				-0.0634
6_landmarks				

sagittal (mahalanobis)	0_landmarks	3_landmarks	5_landmarks	6_landmarks
0_landmarks		0.4706	0.4698	0.4019
3_landmarks			0.0038	-0.0487
5_landmarks				-0.0652
6_landmarks				

axial (hausdorff)	0_landmarks	3_landmarks	5_landmarks	6_landmarks
0_landmarks		132.2138	138.4552	138.9433
3_landmarks			4.4825	8.1210
5_landmarks				2.0210
6_landmarks				

coronal (hausdorff)	0_landmarks	3_landmarks	5_landmarks	6_landmarks
0_landmarks		132.5387	140.7250	136.4050
3_landmarks			10.4869	4.3710
5_landmarks				-4.6444
6_landmarks				

sagittal (hausdorff)	0_landmarks	3_landmarks	5_landmarks	6_landmarks
0_landmarks		127.8980	132.2838	135.3787
3_landmarks			5.5514	8.7703
5_landmarks				1.9909
6_landmarks				

Table B.2: Raw performance data achieved by architectures that were provided with information on different sets of landmarks.

metric	architecture	orientation	test_1	test_2	test_3	test_4	test_5	test_6	test_7	test_8	test_9	test_10	test_11	test_12	test_13	test_14	test_15	training_time
dice	0_landmark	axial	0.16	0.09	0.20	0.19	0.18	0.16	0.19	0.20	0.19	0.17	0.22	0.21	0.10	0.13	0.26	14.93
dice	0_landmark	coronal	0.09	0.05	0.11	0.11	0.11	0.09	0.11	0.11	0.11	0.11	0.10	0.12	0.08	0.07	0.14	6.87
dice	0_landmark	sagittal	0.17	0.10	0.24	0.19	0.22	0.17	0.21	0.24	0.19	0.17	0.26	0.24	0.13	0.15	0.29	7.93
dice	3_landmark	axial	0.50	0.24	0.47	0.63	0.33	0.41	0.53	0.43	0.48	0.44	0.39	0.42	0.28	0.43	0.54	4.14
dice	3_landmark	coronal	0.54	0.24	0.50	0.59	0.37	0.43	0.53	0.54	0.48	0.54	0.36	0.44	0.28	0.50	0.51	3.34
dice	3_landmark	sagittal	0.53	0.25	0.52	0.63	0.39	0.46	0.53	0.52	0.51	0.60	0.42	0.56	0.37	0.44	0.56	4.67
dice	5_landmark	axial	0.48	0.34	0.50	0.51	0.65	0.48	0.46	0.50	0.46	0.45	0.43	0.54	0.32	0.42	0.55	3.01
dice	5_landmark	coronal	0.43	0.35	0.44	0.48	0.66	0.43	0.41	0.50	0.41	0.40	0.42	0.45	0.30	0.39	0.47	2.28
dice	5_landmark	sagittal	0.41	0.34	0.48	0.51	0.63	0.47	0.44	0.49	0.43	0.49	0.41	0.54	0.27	0.38	0.53	5.27
dice	6_landmark	axial	0.40	0.28	0.46	0.53	0.52	0.44	0.53	0.46	0.38	0.46	0.36	0.52	0.38	0.38	0.47	5.92
dice	6_landmark	coronal	0.37	0.35	0.45	0.47	0.67	0.43	0.42	0.49	0.41	0.45	0.45	0.50	0.21	0.29	0.47	3.34
dice	6_landmark	sagittal	0.41	0.36	0.53	0.52	0.61	0.49	0.47	0.48	0.41	0.43	0.40	0.52	0.39	0.36	0.55	4.42
hausdorff	0_landmark	axial	165.30	173.19	183.43	168.32	206.04	212.61	188.62	207.53	198.67	185.34	193.97	179.47	157.64	183.88	175.96	14.93
hausdorff	0_landmark	coronal	159.69	172.49	190.94	169.78	207.65	194.43	189.76	210.96	193.31	189.48	196.66	176.24	162.25	181.56	174.48	6.87
hausdorff	0_landmark	sagittal	156.51	161.95	182.83	164.83	200.92	212.15	178.28	202.80	190.22	183.49	190.34	171.70	158.24	176.14	156.78	7.93
hausdorff	3_landmark	axial	55.26	73.90	60.35	45.98	85.97	67.18	45.35	49.54	51.75	49.99	48.25	45.29	59.24	33.18	49.08	4.14
hausdorff	3_landmark	coronal	39.10	73.11	58.40	45.45	70.58	73.42	41.21	60.24	56.19	31.59	62.32	41.41	77.72	37.07	50.09	3.34
hausdorff	3_landmark	sagittal	39.15	73.56	58.11	45.23	70.49	72.15	41.39	60.11	55.94	31.16	62.71	41.86	77.26	37.16	50.20	4.67
hausdorff	5_landmark	axial	48.76	60.02	44.34	48.38	56.17	47.25	47.10	43.05	50.84	42.50	45.35	47.53	69.07	38.55	48.03	3.01
hausdorff	5_landmark	coronal	37.85	58.28	39.22	39.05	44.70	43.43	39.32	42.64	47.10	56.60	43.31	46.24	62.17	34.25	41.88	2.28
hausdorff	5_landmark	sagittal	41.73	65.60	38.65	39.76	63.98	46.75	38.29	52.17	50.69	39.75	58.56	44.11	50.46	39.24	44.52	5.27
hausdorff	6_landmark	axial	39.90	60.14	42.20	39.67	50.84	48.09	43.15	47.34	47.76	51.97	44.96	44.42	42.97	40.93	49.10	5.92
hausdorff	6_landmark	coronal	45.66	60.02	43.57	45.44	32.65	59.94	46.85	47.60	47.55	42.72	53.64	45.72	67.72	47.55	45.32	3.34
hausdorff	6_landmark	sagittal	42.21	60.90	37.54	42.50	39.86	42.12	36.95	49.70	51.03	41.39	51.24	43.36	38.28	40.41	44.01	4.42
mahalanobis	0_landmark	axial	0.96	0.78	1.06	1.01	1.12	1.19	1.09	1.30	0.85	1.12	0.85	0.93	1.40	1.38	1.06	14.93
mahalanobis	0_landmark	coronal	0.97	0.60	0.91	0.96	0.65	1.06	0.96	1.08	0.89	1.00	0.58	0.74	1.28	1.30	0.82	6.87
mahalanobis	0_landmark	sagittal	0.92	0.71	0.96	0.98	0.85	1.14	0.98	1.13	0.87	1.13	0.70	0.78	1.20	1.30	0.88	7.93
mahalanobis	3_landmark	axial	0.42	1.03	0.10	0.12	0.40	0.53	0.64	0.60	0.68	0.60	0.33	0.35	1.34	0.56	0.23	4.14
mahalanobis	3_landmark	coronal	0.31	0.89	0.04	0.11	0.43	0.48	0.68	0.54	0.88	0.48	0.28	0.43	0.91	0.74	0.25	3.34
mahalanobis	3_landmark	sagittal	0.20	0.94	0.05	0.29	0.44	0.52	0.84	0.66	0.70	0.67	0.26	0.32	0.79	0.71	0.34	4.67
mahalanobis	5_landmark	axial	0.26	0.52	0.41	0.40	0.49	0.40	0.76	0.82	0.62	0.68	0.20	0.17	0.93	0.78	0.31	3.01
mahalanobis	5_landmark	coronal	0.24	0.55	0.39	0.48	0.24	0.45	0.86	0.81	0.47	0.72	0.37	0.15	0.74	0.79	0.18	2.28
mahalanobis	5_landmark	sagittal	0.14	0.72	0.27	0.40	0.34	0.52	0.76	0.71	0.57	0.82	0.31	0.08	0.87	0.78	0.31	5.27
mahalanobis	6_landmark	axial	0.48	0.42	0.33	0.35	0.28	0.33	0.75	0.76	0.78	0.72	0.14	0.30	0.57	1.19	0.39	5.92
mahalanobis	6_landmark	coronal	0.23	0.92	0.35	0.41	0.64	0.34	0.79	0.84	0.56	0.55	0.28	0.29	1.05	1.36	0.19	3.34
mahalanobis	6_landmark	sagittal	0.32	0.55	0.46	0.60	0.34	0.52	1.00	0.90	0.47	0.86	0.41	0.14	0.55	1.32	0.21	4.42

Table B.3: Median differences in performance between pairs of networks that were trained on voxels obtained using the balanced and boundary sampling approaches, grouped by metric and patch orientation.

axial (dice)	balanced	boundary
balanced		-0.0718
boundary		

coronal (dice)	balanced	boundary
balanced		-0.0048
boundary		

sagittal (dice)	balanced	boundary
balanced		-0.0492
boundary		

axial (mahalanobis)	balanced	boundary
balanced		-0.1543
boundary		

coronal (mahalanobis)	balanced	boundary
balanced		-0.2627
boundary		

sagittal (mahalanobis)	balanced	boundary
balanced		-0.1105
boundary		

axial (hausdorff)	balanced	boundary
balanced		0.9395
boundary		

coronal (hausdorff)	balanced	boundary
balanced		-3.2012
boundary		

sagittal (hausdorff)	balanced	boundary
balanced		0.8070
boundary		

Table B.4: Raw performance data achieved by networks that were trained on voxels obtained using the balanced and boundary sampling approaches.

metric	sampling	orientation	test_1	test_2	test_3	test_4	test_5	test_6	test_7	test_8	test_9	test_10	test_11	test_12	test_13	test_14	test_15	training_time
dice	balanced	axial	0.50	0.24	0.47	0.63	0.33	0.41	0.53	0.43	0.48	0.44	0.39	0.42	0.28	0.43	0.54	4.14
dice	balanced	coronal	0.54	0.24	0.50	0.59	0.37	0.43	0.53	0.54	0.48	0.54	0.36	0.44	0.28	0.50	0.51	3.34
dice	balanced	sagittal	0.53	0.25	0.52	0.63	0.39	0.46	0.53	0.52	0.51	0.60	0.42	0.56	0.37	0.44	0.56	4.67
dice	boundary	axial	0.53	0.28	0.57	0.62	0.49	0.52	0.59	0.55	0.54	0.71	0.42	0.52	0.32	0.52	0.57	3.09
dice	boundary	coronal	0.53	0.25	0.49	0.56	0.60	0.47	0.57	0.60	0.44	0.62	0.34	0.41	0.21	0.48	0.53	2.89
dice	boundary	sagittal	0.55	0.33	0.58	0.59	0.49	0.53	0.62	0.61	0.52	0.63	0.45	0.51	0.41	0.48	0.70	1.93
hausdorff	balanced	axial	55.26	73.90	60.35	45.98	85.97	67.18	45.35	49.54	51.75	49.99	48.25	45.29	59.24	33.18	49.08	4.14
hausdorff	balanced	coronal	39.10	73.11	58.40	45.45	70.58	73.42	41.21	60.24	56.19	31.59	62.32	41.41	77.72	37.07	50.09	3.34
hausdorff	balanced	sagittal	39.15	73.56	58.11	45.23	70.49	72.15	41.39	60.11	55.94	31.16	62.71	41.86	77.26	37.16	50.20	4.67
hausdorff	boundary	axial	42.21	73.74	54.06	47.01	63.55	59.40	49.37	58.63	48.72	40.11	54.42	43.93	60.50	39.76	57.70	3.09
hausdorff	boundary	coronal	45.10	68.45	50.63	46.91	73.67	63.65	53.23	64.13	51.47	64.40	65.89	51.02	56.75	42.24	64.47	2.89
hausdorff	boundary	sagittal	39.23	63.78	53.29	43.37	73.78	61.81	50.02	61.34	49.86	53.48	71.04	45.40	45.00	43.06	41.05	1.93
mahalanobis	balanced	axial	0.42	1.03	0.10	0.12	0.40	0.53	0.64	0.60	0.68	0.60	0.33	0.35	1.34	0.56	0.23	4.14
mahalanobis	balanced	coronal	0.31	0.89	0.04	0.11	0.43	0.48	0.68	0.54	0.88	0.48	0.28	0.43	0.91	0.74	0.25	3.34
mahalanobis	balanced	sagittal	0.20	0.94	0.05	0.29	0.44	0.52	0.84	0.66	0.70	0.67	0.26	0.32	0.79	0.71	0.34	4.67
mahalanobis	boundary	axial	0.61	0.49	0.29	0.53	0.67	0.21	0.87	0.58	0.91	0.32	0.69	0.69	1.23	0.97	0.52	3.09
mahalanobis	boundary	coronal	0.76	0.44	0.59	0.75	0.51	0.36	0.76	0.44	1.08	0.11	0.92	1.43	1.37	1.00	0.70	2.89
mahalanobis	boundary	sagittal	0.75	0.71	0.22	0.78	0.45	0.25	0.72	0.62	0.89	0.45	0.67	0.55	1.01	1.01	0.34	1.93

Table B.5: Median differences in performance between pairs of networks that included either single patches or three orthogonal patches, grouped by metric and number of artificial neurons included in the hidden layer.

less_neurons (dice)	axial	coronal	sagittal	multiple
multiple		-0.02606275	0.0154	-0.0457
axial			0.0499	-0.0158
coronal				-0.0595
sagittal				

more_neurons (dice)	multiple	axial	coronal	sagittal
multiple		0.0126	0.0700	-0.0371
axial			0.0741	-0.0508
coronal				-0.1124
sagittal				

multiple (dice)	less_neurons	more_neurons
less_neurons		-0.0272
more_neurons		

less_neurons (mahalanobis)	multiple	axial	coronal	sagittal
multiple		-0.0318	-0.1373	-0.0472
axial			-0.0905	0.0136
coronal				0.0960
sagittal				

more_neurons (mahalanobis)	multiple	axial	coronal	sagittal
multiple		-0.1251	-0.3725	-0.1716
axial			-0.1880	-0.0209
coronal				0.1754
sagittal				

multiple (mahalanobis)	less_neurons	more_neurons
less_neurons		0.1182
more_neurons		

less_neurons (hausdorff)	multiple	axial	coronal	sagittal
multiple		6.5485	2.9000	6.2730
axial			-3.923	-0.4539
coronal				3.4597
sagittal				

more_neurons (hausdorff)	multiple	axial	coronal	sagittal
multiple		1.2604	-0.0822	7.6208
axial			-1.8086	6.0604
coronal				7.1802
sagittal				

multiple (hausdorff)	less_neurons	more_neurons
less_neurons		1.0826
more_neurons		

Table B.6: Raw performance data achieved by networks that included either a single patch, or three orthogonal patches.

metric	architecture	orientation	test_1	test_2	test_3	test_4	test_5	test_6	test_7	test_8	test_9	test_10	test_11	test_12	test_13	test_14	test_15	training_time
dice	less_neurons	axial	0.53	0.28	0.57	0.62	0.49	0.52	0.59	0.55	0.54	0.71	0.42	0.52	0.32	0.52	0.57	3.09
dice	less_neurons	coronal	0.53	0.25	0.49	0.56	0.60	0.47	0.57	0.60	0.44	0.62	0.34	0.41	0.21	0.48	0.53	2.89
dice	less_neurons	sagittal	0.55	0.33	0.58	0.59	0.49	0.53	0.62	0.61	0.52	0.63	0.45	0.51	0.41	0.48	0.70	1.93
dice	less_neurons	multiple	0.50	0.27	0.48	0.66	0.52	0.44	0.62	0.46	0.45	0.60	0.46	0.51	0.26	0.45	0.63	3.41
dice	more_neurons	axial	0.56	0.29	0.56	0.64	0.46	0.48	0.56	0.51	0.51	0.61	0.45	0.48	0.43	0.41	0.68	6.77
dice	more_neurons	coronal	0.46	0.27	0.54	0.55	0.45	0.55	0.50	0.54	0.40	0.52	0.33	0.21	0.16	0.48	0.50	13.45
dice	more_neurons	sagittal	0.53	0.36	0.67	0.59	0.57	0.57	0.60	0.62	0.52	0.65	0.53	0.53	0.28	0.54	0.71	11.35
dice	more_neurons	multiple	0.49	0.26	0.55	0.58	0.52	0.49	0.59	0.53	0.51	0.67	0.48	0.60	0.22	0.53	0.69	6.53
hausdorff	less_neurons	axial	42.21	73.74	54.06	47.01	63.55	59.40	49.37	58.63	48.72	40.11	54.42	43.93	60.50	39.76	57.70	3.09
hausdorff	less_neurons	coronal	45.10	68.45	50.63	46.91	73.67	63.65	53.23	64.13	51.47	64.40	65.89	51.02	56.75	42.24	64.47	2.89
hausdorff	less_neurons	sagittal	39.23	63.78	53.29	43.37	73.78	61.81	50.02	61.34	49.86	53.48	71.04	45.40	45.00	43.06	41.05	1.93
hausdorff	less_neurons	multiple	49.56	80.21	56.44	49.26	77.45	70.38	59.56	66.65	62.46	47.58	52.08	53.76	41.09	49.35	52.86	3.41
hausdorff	more_neurons	axial	48.26	73.99	53.16	52.18	78.68	64.82	60.73	71.04	44.88	56.15	57.87	48.89	53.12	44.06	48.34	6.77
hausdorff	more_neurons	coronal	50.73	75.59	62.33	56.92	80.06	72.94	52.78	65.01	51.05	62.50	59.11	47.04	51.90	38.96	54.18	13.45
hausdorff	more_neurons	sagittal	39.13	61.95	53.60	50.34	65.46	53.27	46.66	62.66	51.52	51.18	55.84	41.79	69.58	39.87	40.36	11.35
hausdorff	more_neurons	multiple	51.33	79.66	58.55	65.69	71.81	72.67	58.45	66.13	50.46	43.19	55.67	45.92	92.23	39.75	47.33	6.53
mahalanobis	less_neurons	axial	0.61	0.49	0.29	0.53	0.67	0.21	0.87	0.58	0.91	0.32	0.69	0.69	1.23	0.97	0.52	3.09
mahalanobis	less_neurons	coronal	0.76	0.44	0.59	0.75	0.51	0.36	0.76	0.44	1.08	0.11	0.92	1.43	1.37	1.00	0.70	2.89
mahalanobis	less_neurons	sagittal	0.75	0.71	0.22	0.78	0.45	0.25	0.72	0.62	0.89	0.45	0.67	0.55	1.01	1.01	0.34	1.93
mahalanobis	less_neurons	multiple	0.70	0.67	0.26	0.53	0.42	0.31	0.53	0.64	0.81	0.53	0.46	0.60	1.67	0.85	0.21	3.41
mahalanobis	more_neurons	axial	0.52	0.81	0.28	0.67	0.66	0.47	0.95	0.67	0.69	0.53	0.30	0.20	1.23	1.09	0.24	6.77
mahalanobis	more_neurons	coronal	0.77	0.31	0.65	0.89	0.55	0.33	0.96	0.40	1.31	0.35	0.95	2.04	1.61	1.15	0.73	13.45
mahalanobis	more_neurons	sagittal	0.86	0.87	0.14	0.77	0.61	0.30	0.81	0.48	0.95	0.44	0.40	0.57	1.38	0.82	0.34	11.35
mahalanobis	more_neurons	multiple	0.44	0.77	0.12	0.50	0.34	0.40	0.49	0.50	0.70	0.37	0.19	0.33	1.10	0.73	0.24	6.53

Table B.7: Median differences in performance between pairs of networks that included between one and three fully connected or convolutional and max-pooling layers, grouped by metric, layer type, and number of layers.

fully_connected (dice)	0_layers	1_layer	2_layers	3_layers
0_layers		-0.0099	0.0420	0.0093
1_layer			0.0557	0.0227
2_layers				-0.0317
3_layers				

convolutional (dice)	1_layer	2_layers	3_layers
1_layer		0.0021	-0.0512
2_layers			-0.0512
3_layers			

1_layer (dice)	convolutional	fully_connected
convolutional		0.0388
fully_connected		

2_layers (dice)	convolutional	fully_connected
convolutional		0.0868
fully_connected		

3_layers (dice)	convolutional	fully_connected
convolutional		0.1048
fully_connected		

fully_connected (mahalanobis)	0_layers	1_layer	2_layers	3_layers
0_layers		-0.0851	-0.1148	-0.0396
1_layer			-0.0166	0.0282
2_layers				0.0664
3_layers				

convolutional (mahalanobis)	1_layer	2_layers	3_layers
1_layer		0.0051	-0.0281
2_layers			-0.0316
3_layers			

1_layer (mahalanobis)	convolutional	fully_connected
convolutional		-0.0618
fully_connected		

2_layers (mahalanobis)	convolutional	fully_connected
convolutional		-0.1045
fully_connected		

3_layers (mahalanobis)	convolutional	fully_connected
convolutional		-0.0226
fully_connected		

fully_connected (hausdorff)	0_layers	1_layer	2_layers	3_layers
0_layers		-6.0407	-4.3983	-2.5075
1_layer			1.9543	2.8827
2_layers				-0.2367
3_layers				

convolutional (hausdorff)	1_layer	2_layers	3_layers
1_layer		3.8486	11.3297
2_layers			6.4532
3_layers			

1_layer (hausdorff)	convolutional	fully_connected
convolutional		4.2671
fully_connected		

2_layers (hausdorff)	convolutional	fully_connected
convolutional		1.8401
fully_connected		

3_layers (hausdorff)	convolutional	fully_connected
convolutional		-3.5554
fully_connected		

Table B.8: Raw performance data achieved by networks that included between one and three fully connected or convolutional and max-pooling layers.

metric	architecture	num_layers	test_1	test_2	test_3	test_4	test_5	test_6	test_7	test_8	test_9	test_10	test_11	test_12	test_13	test_14	test_15	training_time
dice	fully_connected	0	0.49	0.26	0.55	0.58	0.52	0.49	0.59	0.53	0.51	0.67	0.48	0.60	0.22	0.53	0.69	6.53
dice	fully_connected	1	0.49	0.28	0.67	0.57	0.51	0.53	0.52	0.60	0.49	0.63	0.54	0.56	0.41	0.51	0.66	59.18
dice	fully_connected	2	0.45	0.22	0.56	0.52	0.43	0.50	0.53	0.49	0.47	0.56	0.48	0.50	0.38	0.50	0.58	27.38
dice	fully_connected	3	0.46	0.24	0.63	0.51	0.48	0.50	0.55	0.56	0.48	0.56	0.59	0.56	0.35	0.54	0.68	13.35
dice	convolutional	1	0.48	0.35	0.71	0.55	0.61	0.58	0.59	0.66	0.51	0.63	0.63	0.63	0.35	0.53	0.74	152.35
dice	convolutional	2	0.46	0.34	0.71	0.52	0.58	0.57	0.61	0.66	0.51	0.61	0.66	0.61	0.35	0.56	0.74	255.27
dice	convolutional	3	0.52	0.42	0.77	0.54	0.68	0.64	0.65	0.70	0.53	0.71	0.69	0.71	0.35	0.60	0.76	564.09
hausdorff	fully_connected	0	51.33	79.66	58.55	65.69	71.81	72.67	58.45	66.13	50.46	43.19	55.67	45.92	92.23	39.75	47.33	6.53
hausdorff	fully_connected	1	65.67	82.86	56.17	55.19	93.28	69.17	62.07	73.16	63.07	57.77	67.68	55.51	66.53	49.30	51.24	59.18
hausdorff	fully_connected	2	64.16	85.99	54.43	58.05	90.97	59.57	66.61	62.62	57.71	58.79	67.96	55.62	47.11	38.86	66.67	27.38
hausdorff	fully_connected	3	67.20	82.26	51.20	62.21	90.01	63.83	60.02	71.18	58.43	61.93	52.11	56.67	51.63	40.83	49.56	13.35
hausdorff	convolutional	1	49.84	92.03	59.30	69.31	86.75	60.22	69.06	79.22	58.15	68.67	68.13	70.78	77.31	52.95	59.13	152.35
hausdorff	convolutional	2	59.54	81.82	55.23	62.27	94.92	59.67	57.02	73.80	58.29	77.27	65.02	58.69	67.01	42.46	56.74	255.27
hausdorff	convolutional	3	63.07	74.39	52.62	61.90	83.54	57.42	58.77	50.26	59.13	50.81	55.19	47.31	55.25	36.81	50.37	564.09
mahalanobis	fully_connected	0	0.44	0.77	0.12	0.50	0.34	0.40	0.49	0.50	0.70	0.37	0.19	0.33	1.10	0.73	0.24	6.53
mahalanobis	fully_connected	1	0.65	0.63	0.12	0.49	0.58	0.26	0.70	0.53	0.85	0.40	0.35	0.44	1.16	0.91	0.27	59.18
mahalanobis	fully_connected	2	0.62	0.49	0.33	0.50	0.61	0.06	0.55	0.63	0.88	0.39	0.68	0.50	1.10	0.89	0.50	27.38
mahalanobis	fully_connected	3	0.51	0.91	0.14	0.43	0.63	0.33	0.41	0.64	0.78	0.46	0.25	0.32	1.25	0.73	0.12	13.35
mahalanobis	convolutional	1	0.74	0.66	0.12	0.45	0.34	0.17	0.31	0.49	0.83	0.36	0.35	0.34	1.18	0.73	0.13	152.35
mahalanobis	convolutional	2	0.70	0.40	0.12	0.55	0.26	0.17	0.46	0.44	0.82	0.34	0.27	0.35	1.34	0.79	0.11	255.27
mahalanobis	convolutional	3	0.63	0.71	0.12	0.59	0.33	0.29	0.43	0.38	0.99	0.18	0.49	0.48	1.15	0.83	0.04	564.09

Table B.9: Median differences in performance between pairs of networks that included context patches or a dropout layer, grouped by metric.

context (dice)	context_51	context_75	local
context_51		0.0203	0.0178
context_75			-0.0040
local			

dropout (dice)	context_51	dropout
context_51		-0.0172
dropout		

context (mahalanobis)	context_51	context_75	local
context_51		0.0450	-0.0313
context_75			-0.0719
local			

dropout (mahalanobis)	context_51	dropout
context_51		-0.0328
dropout		

context (hausdorff)	context_51	context_75	local
context_51		-0.3023	-6.5160
context_75			-5.3796
local			

dropout (hausdorff)	context_51	dropout
context_51		6.5620
dropout		

Table B.10: Raw performance data achieved by networks that included context patches or a dropout layer.

metric	architecture	test_1	test_2	test_3	test_4	test_5	test_6	test_7	test_8	test_9	test_10	test_11	test_12	test_13	test_14	test_15	training_time
dice	local	0.52	0.42	0.77	0.54	0.68	0.64	0.65	0.70	0.53	0.71	0.69	0.71	0.35	0.60	0.76	564.09
dice	context_51	0.55	0.40	0.77	0.58	0.67	0.67	0.68	0.72	0.54	0.72	0.73	0.77	0.38	0.62	0.73	1345.56
dice	context_75	0.55	0.37	0.76	0.59	0.68	0.63	0.64	0.68	0.61	0.66	0.72	0.70	0.37	0.57	0.75	1389.12
dice	dropout	0.57	0.34	0.78	0.61	0.67	0.68	0.73	0.74	0.58	0.74	0.75	0.77	0.39	0.67	0.71	1155.18
hausdorff	local	63.07	74.39	52.62	61.90	83.54	57.42	58.77	50.26	59.13	50.81	55.19	47.31	55.25	36.81	50.37	564.09
hausdorff	context_51	43.79	64.33	43.19	56.38	76.46	56.34	57.59	41.79	50.34	39.87	53.77	49.30	44.81	24.91	53.28	1345.56
hausdorff	context_75	39.46	89.49	38.90	57.22	79.72	47.77	63.13	50.10	50.46	54.28	48.28	37.35	49.15	22.60	48.91	1389.12
hausdorff	dropout	23.49	62.56	29.79	49.74	57.86	48.26	44.31	52.96	43.65	38.34	42.84	49.52	45.54	20.36	47.82	1155.18
mahalanobis	local	0.63	0.71	0.12	0.59	0.33	0.29	0.43	0.38	0.99	0.18	0.49	0.48	1.15	0.83	0.04	564.09
mahalanobis	context_51	0.68	0.54	0.16	0.60	0.37	0.16	0.30	0.40	0.98	0.07	0.60	0.34	1.15	0.86	0.06	1345.56
mahalanobis	context_75	0.59	0.55	0.15	0.40	0.26	0.12	0.26	0.37	0.93	0.14	0.43	0.49	0.98	0.83	0.05	1389.12
mahalanobis	dropout	0.69	0.76	0.11	0.62	0.50	0.04	0.22	0.55	0.79	0.22	0.56	0.15	1.33	1.13	0.02	1155.18

Table B.11: Median differences in performance between the network with a dropout layer that did not include structured prediction, and either the connected component algorithm or secondary network approach.

components (dice)	dropout	components
dropout		-0.0119
components		
secondary (dice)	dropout	secondary
dropout		0.0501
secondary		

components (mahalanobis)	dropout	components
dropout		-0.0225
components		
secondary (mahalanobis)	dropout	secondary
dropout		0.0670
secondary		

components (hausdorff)	dropout	components
dropout		4.6248
components		
secondary (hausdorff)	dropout	secondary
dropout		9.4397
secondary		

Table B.12: Raw performance data achieved by the network with a dropout layer, when combined with approaches to structured prediction.

metric	architecture	test_1	test_2	test_3	test_4	test_5	test_6	test_7	test_8	test_9	test_10	test_11	test_12	test_13	test_14	test_15	training_time
dice	dropout	0.57	0.34	0.78	0.61	0.67	0.68	0.73	0.74	0.58	0.74	0.75	0.77	0.39	0.67	0.71	1155.18
dice	secondary	0.67	0.24	0.64	0.61	0.57	0.64	0.73	0.62	0.65	0.64	0.70	0.62	0.45	0.63	0.61	1070.12
dice	components	0.58	0.36	0.79	0.62	0.69	0.69	0.74	0.75	0.58	0.77	0.76	0.79	0.41	0.68	0.72	0.00
hausdorff	dropout	23.49	62.56	29.79	49.74	57.86	48.26	44.31	52.96	43.65	38.34	42.84	49.52	45.54	20.36	47.82	1155.18
hausdorff	secondary	15.42	51.71	15.86	45.72	34.32	33.14	26.11	49.24	31.52	20.78	27.31	47.43	54.88	26.48	44.33	1070.12
hausdorff	components	9.95	59.68	30.27	46.79	38.14	47.57	35.21	43.00	43.19	36.39	34.28	45.39	44.86	20.17	47.02	0.00
mahalanobis	dropout	0.69	0.76	0.11	0.62	0.50	0.04	0.22	0.55	0.79	0.22	0.56	0.15	1.33	1.13	0.02	1155.18
mahalanobis	secondary	0.29	0.87	0.31	0.37	0.41	0.09	0.32	0.66	0.20	0.57	0.31	0.12	1.16	0.66	0.18	1070.12
mahalanobis	components	0.72	0.94	0.11	0.68	0.52	0.04	0.27	0.54	0.80	0.21	0.60	0.13	1.48	1.14	0.06	0.00

REFERENCES

Babarenda Gamage, T. P. B., Boyes, R., Rajagopal, V., Nielsen, P. M., & Nash, M. P. (2012). Modelling prone to supine breast deformation under gravity loading using heterogeneous finite element models. In *Computational biomechanics for medicine* (pp. 29–38). New York: Springer.

Baldi, P. & Sadowski, P. J. (2013). Understanding dropout. In *Advances in neural information processing systems* (pp. 2814–2822). New York: Curran Associates, Inc.

Baluwala, H. Y., Malcolm, D. T., Jor, J. W., Nielsen, P. M., & Nash, M. P. (2015). Automatic landmark detection using statistical shape modelling and template matching. In *Computational biomechanics for medicine* (pp. 75–82). New York: Springer.

Bergmann, B. & Hommel, G. (1988). Improvements of general multiple test procedures for redundant systems of hypotheses. In *Multiple hypothesenprüfung/multiple hypotheses testing* (pp. 100–115). New York: Springer.

Berry, M. J. & Linoff, G. (1997). *Data mining techniques: for marketing, sales, and customer support*. hoboken: Wiley.

Bottou, L., Curtis, F. E., & Nocedal, J. (2016, June). Optimization methods for large-scale machine learning. *ArXiv e-prints*. arXiv: 1606.04838

Byeon, W., Breuel, T. M., Raue, F., & Liwicki, M. (2015). Scene labeling with lstm recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3547–3555).

Calvo, B. & Santafe, G. (2015). Scmamp: statistical comparison of multiple algorithms in multiple problems. *The R Journal*.

Ciresan, D. C., Meier, U., Masci, J., Maria Gambardella, L., & Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. In *Ijcai proceedings-international joint conference on artificial intelligence* (Vol. 22, *1*, pp. 1237–1242).

de Brebisson, A. & Montana, G. (2015). Deep neural networks for anatomical brain segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 20–28).

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*(Jan), 1–30.

Dice, L. R. (1945). Measures of the amount of ecologic association between species. *Ecology*, *26*(3), 297–302.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems* (pp. 1–15). New York.

Dreiseitl, S. & Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics*, *35*(5), 352–359.

Ganesan, K., Acharya, U. R., Chua, K. C., Min, L. C., & Abraham, K. T. (2013). Pectoral muscle segmentation: a review. *Computer Methods and Programs in Biomedicine*, *110*(1), 48–57.

Garcia, S. & Herrera, F. (2008). An extension on"statistical comparisons of classifiers over multiple data sets"for all pairwise comparisons. *Journal of Machine Learning Research*, *9*(Dec), 2677–2694.

Giusti, A., Cireşan, D. C., Masci, J., Gambardella, L. M., & Schmidhuber, J. (2013). Fast image scanning with deep max-pooling convolutional neural networks. *ArXiv e-prints*. arXiv: 1302.1700

Glorot, X. & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *AISTATS* (Vol. 9, pp. 249–256). JMLR Proceedings. JMLR.org.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *AISTATS* (Vol. 15, pp. 315–323). JMLR Proceedings. JMLR.org.

Gubern-Mérida, A., Kallenberg, M., Martí, R., & Karssemeijer, N. (2012). Segmentation of the pectoral muscle in breast mri using atlas-based approaches. In *International conference on medical image computing and computer-assisted intervention* (pp. 371–378). Springer. New York.

Håstad, J. & Goldmann, M. (1991). On the power of small-depth threshold circuits. *Computational Complexity*, *1*(2), 113–129.

Hausdorff, F. (1927). *Mengenlehre*. Berlin: Walter de Gruyter.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv e-prints*. arXiv: 1207.0580

Karpathy, A. (2016). Convolutional neural networks for visual recognition. Stanford University. Retrieved from cs231n.github.io/convolutional-networks/

Karssemeijer, N. (1998). Automated classification of parenchymal patterns in mammograms. *Physics in Medicine and Biology*, *43*(2), 365–378.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 25* (pp. 1097–1105). New York: Curran Associates, Inc.

Kröse, B. & van der Smagt, P. (1993). An introduction to neural networks.

Lai, M. (2015). Deep learning for medical image segmentation. *ArXiv e-prints*. arXiv: 1505.02000

Lasagne Development Team. (2016). Lasagne. https://github.com/Lasagne/Lasagne/. GitHub.

LeCun, Y. (2015). What's wrong with deep learning? Presentation delivered at CVPR 2015.

Leshno, M., Lin, V. Y., Pinkus, A., & Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, *6*(6), 861–867.

Liang, M., Hu, X., & Zhang, B. (2015). Convolutional neural networks with intralayer recurrent connections for scene labeling. In *Advances in neural information processing systems* (pp. 937–945). New York: Curran Associates, Inc.

Liao, Y., Kodagoda, S., Wang, Y., Shi, L., & Liu, Y. (2016). Understanding scene categories by objects: a semantic regularized scene classifier using convolutional neural networks. In *Ieee international conference on robotics and automation* (pp. 2318–2325). Piscataway: IEEE.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431–3440).

MacKay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge: Cambridge University Press.

McLachlan, G. J. (1999). Mahalanobis distance. *Resonance*, *4*(6), 20–26.

Nouri, D. (2014). Nolearn: scikit-learn compatible neural network. https://github.com/dnouri/nolearn. GitHub.

Pandya, S. & Moore, R. G. (2011). Breast development and anatomy. *Clinical Obstetrics and Gynecology*, *54*(1), 91–95.

Pinheiro, P. H. & Collobert, R. (2014). Recurrent convolutional neural networks for scene labeling. In *International conference on machine learning* (pp. 82–90).

Refaeilzadeh, P., Tang, L., & Liu, H. (2009). Cross-validation. In *Encyclopedia of database systems* (pp. 532–538). New York: Springer.

Saltanat, N., Hossain, M. A., & Alam, M. S. (2010). An efficient pixel value based mapping scheme to delineate pectoral muscle from mammograms. In *IEEE fifth international conference on bio-inspired computing: theories and applications* (pp. 1510–1517). Piscataway: IEEE.

Sharma, N., Aggarwal, L. M. et al. (2010). Automated medical image segmentation techniques. *Journal of Medical Physics*, *35*(1), 3.

Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *ArXiv e-prints*. arXiv: 1409.1556

Sultana, A., Ciuc, M., & Strungaru, R. (2010). Detection of pectoral muscle in mammograms using a mean-shift segmentation approach. In *IEEE eighth international conference on communications* (pp. 165–168). Piscataway: IEEE.

Sutskever, I., Martens, J., Dahl, G. E., & Hinton, G. E. (2013). On the importance of initialization and momentum in deep learning, 1139–1147.

Taha, A. A. & Hanbury, A. (2015). Metrics for evaluating 3d medical image segmentation: analysis, selection, and tools. *BMC Medical Imaging*, *15*(1), 15–29.

Theano Development Team. (2016). Theano: A Python framework for fast computation of mathematical expressions. *ArXiv e-prints*. arXiv: 1605.02688

Wang, H.-J., Wang, J.-L., Wang, M.-H., & Yin, Y.-M. (2010). Efficient image inpainting based on bilinear interpolation downscaling. *Optics and Precision Engineering*, *18*(5), 1234–1241.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550–1560.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, *1*(6), 80–83.

Wilson, D. R. & Martinez, T. R. (2003). The general inefficiency of batch training for gradient descent learning. *Neural Networks*, *16*(10), 1429–1451.

Wu, S., Weinstein, S., & Kontos, D. (2012). Atlas-based probabilistic fibroglandular tissue segmentation in breast mri. In *International conference on medical image computing and computer-assisted intervention* (pp. 437–445). Springer. New York.

Yam, M., Brady, M., Highnam, R., Behrenbruch, C., English, R., & Kita, Y. (2001). Three-dimensional reconstruction of microcalcification clusters from two mammographic views. *IEEE Transactions on Medical Imaging*, *20*(6), 479–489.

Yapa, R. D. & Harada, K. (2008). Connected component labeling algorithms for grey-scale images and evaluation of performance using digital mammograms. *International Journal of Computer Science and Network Security*, *8*(6), 33–41.