Imperial College London

Imperial College London
Derpartment of Mathematics

Segmentation of CT scans into Atrium/non Atrium

Thomas Vogel

CID:

Supervised by Prof Giovanni Montana

26th August 2015

Submitted in partial fulfilment of the requirements for the MSc in Statistics of Imperial College London

The work contained in this thesis is my own work unless otherwise stated.

Signed: Date:

Abstract

This is a LaTeX template to be used for project theses by the students of the Imperial College MSc in Statistics. Please have a look at the `*.tex` source and the code comments therein. You may use this template with only minor changes, or make any major style changes you desire (e.g. redesigning the title page, adding headers,...), or you may use a different template, or you may write your own approach from scratch, or you may just use some bits and pieces from the template's LaTeX code. It's up to you. Note that resources on how to use LaTeX are available on Blackboard under 'R&LaTeX intro'.

This template will quote relevant sections from the MSc in Statistics student handbook throughout; e.g.

> " The abstract should be a brief statement of the aims and outcomes of the project, to summarise/advise even for a casual reader!"

Acknowledgements

Thank you supervisor/friends/family/pet.

“ Include an acknowledgement.”

Table of Contents

1 Introduction

Describe some high level background about Machine Learning, deep learning, CNNs, Medical Imaging, literature review style.

2 Background Material: on convolutional neural networks

We first start with a brief review of feed forward Neural Networks before elaborating on Convolutional Neural Networks (CNNs), a specialised kind of neural network which will be relied on for our architecture for our classification problem. CNNs are a special type of Feed-Forward Neural Networks that are particularly effective for input types that have a grid-like structure, such as images, videos or time-series, where the correlation between inputs is highly localised. CNNs have recently been responsible for some of the state-of-the-art results in image recognition [references], speech recognition [references], and natural language processing [reference]. We will discuss CNNs in the context of images.

2.1 Feed Forward Neural Networks

A feedforward Neural networks is a standard model in the machine literature for classification problems. It consists in a number of layers of simple processing units, also called neurons or nodes. Every unit in a layer is connected with all the units in the previous layer, with each connection characterised by a weight encoding the knowledge of the network. The value at each unit is generated by taking a linear combination of the values of the incoming connections weighted by their weights and the result being passed through an activation function. Data enters at the inputs and passes through the network, layer by layer, until it arrives at the output layer. The choice of the activation function is determined by the nature of the data and the assumed distribution of target variables. For classification problems, the softmax function or, in our case, its log, give the output a probabilistic interpretation.

This model can be represented in the form of a network diagram as shown in figure whatever. The process can then be interpreted as a forward propagation of information through the network.

Training the neural networks is done by minimising an error measure of its performance over a training set using gradient-based optimisation routines. Evaluating the error gradients with respect to the parameters can be efficiently done via the Backpropagation algorithm [reference]. Regularisation such as dropout [reference] where at every epoch, only a preset percentage of randomly chosen set of hidden units are active, is generally added to the optimisation procedure to prevent overfitting.

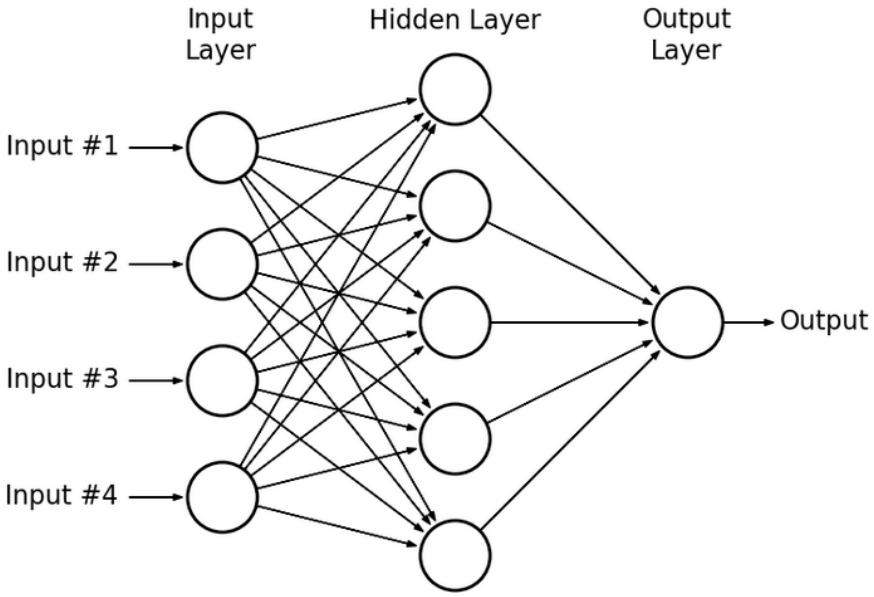

Figure 2.1: A 2 layer feedforward neural network with 4 input nodes, 5 hidden nodes and 1 output node.

2.2 Convolutional Layers

A CNN is a specialised kind of feedforward neural network where the first few layers of the architecture are convolutional layers. These typically consist of three stages: a convolution stage, a detection stage and a subsampling stage.

2.2.1 Convolution

The convolution stage is responsible for implementing k convolution operation over each channel of the input images. This is accomplished by convolving each channel of the input layer with k kernels of size $s \times s$ in parallel. Unlike fully connected layers, in a convolution, each output node is locally connected to a small contiguous subset of corresponding input nodes determined by the kernel size. Additionally the convolution operation implies weight sharing which greatly reduces the number of trainable parameters. In the case of $m \times m$ input images, the set of output nodes resulting from one kernel is called a feature map of size $m - s + 1$, each detecting the presence of a particular feature in the input image encoded by the kernel. Having multiple kernels run through the input layer in parallel produces a set of feature maps, each responsible for detecting a particular type of feature which might be present in the image.

The benefits of this approach are three folds :

- sparse interaction: every output node is connected to a local subset of inputs.

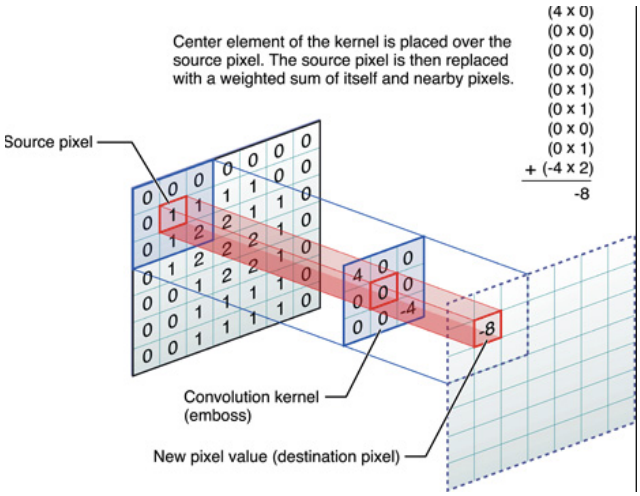

Figure 2.2: The convolution layer. Illustration of the formation of a feature map.

- weight sharing: the same kernel, and thus set of trainable parameters, is used for every output node to produce a given feature map.
- translational equivariance: shifting the input results in an equivalent shift in the output. This is a property of the convolution operation.

In the detector stage, every node in the feature map is then passed to an nonlinear activation unit thereby introducing non-linearity into the system. These activation units usually consist of either Rectified Linear units (ReLU), Tanh units, or Sigmoid units.

Each feature map is then subsampled in the pooling stage by aggregating contiguous nodes using a summary statistic over a rectangular neighbourhood of outputs. Two commonly used ones are the max and average pooling operations which report respectively the maximum and average of a set of inputs.

The benefits of subsampling are that it reduces the number of features by a factor of k for pooling regions spaced k pixels apart, aiding the classification task and improving the computational efficiency of the network. Furthermore subsampling makes the representation become invariant to small translations of the input. Thus translating the input by a small amount results in little to no change in the values of the pooled outputs.

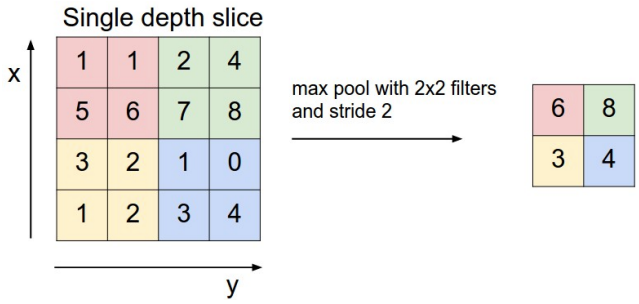

Figure 2.3: Subsampling in action. Here maxpooling is used to halve the size of an feature map using a 2×2 filter

2.2.2 Typical Architecture

A typical convolutional neural network architecture consists of an input layer, a number of convolutional layers, a number of fully connected layers, followed by an output layer. The input layer is an $r \times n \times n$ image, with r being the number of channels and n the image height and width. Each channel is passed through a number convolutional layer in parallel. Then the remaining output nodes from all the resulting feature maps across all channels are "flattened" and passed as input to a number of fully connected layers followed by an output layer. The convolutional layers thus serve as a way to reduce the dimensionality of the image by extracting meaningful local features.

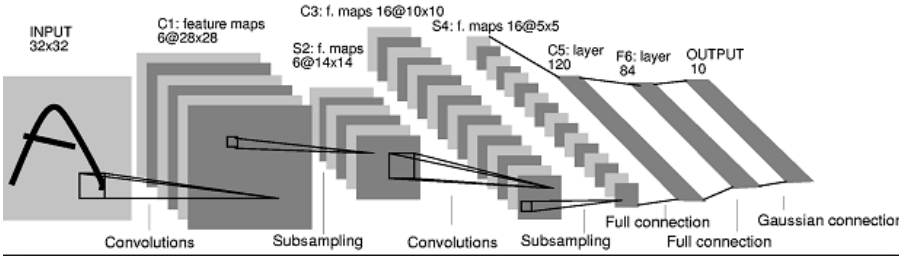

Figure 2.4: A typical convolutional network architecture with 2 convolutional layers, 2 fully connected layer and a Gaussian connection as output layer.

3 Searching for a good model

3.1 The Data

The aim of this thesis is to use deep learning techniques to segment CT scans of the chest and identify the voxels that are in the part of the heart called the atrium. The atrium is composed of two chambers called the right and left atrium which are the two entry points of the blood into the heart, one returning to the heart to complete the circulating cycle through the body and the other coming back from the lungs after being oxygenated.

The data from which the training and testing datasets are generated comprise 27 chest computerised tomography (CT) scans. CT scans are 3 dimensional grey scale images generated via computers combining many X-ray images taken from different angles to produce cross-sectional images of specific body parts. They are stored as DICOM files, DICOM standing for Digital Imaging and Communications in Medicine and which is a standard for handling, storing, printing, and transmitting information in medical imaging. Each has been segmented by trained radiologists at St Thomas's hospital, the results of which are stored in Nearly Raw Raster Data (NRRD) files, a standard format for storing raster data. These are 3D arrays of the same dimension, each voxel taking either or two values which indicates whether the corresponding voxel in the CT scan is in the Atrium (1) or not (0).

3.2 Generating the datasets: The Tri-Planar Method

Classifying the voxels require building an input set for each containing enough local and global information to allow the Neural Network to learn effectively. One way of providing 3 dimensional information is to use the so-called tri-planar method [reference the paper that started it on knee cartilage]. This consists in generating 3 perpendicular square patches of a given dimension in the transversal, saggital and coronal planes centred at the voxel of interest (see the figure). This standard technique has been found to give competitive results compared to using 3D patches while being much more computationally and memory efficient. In addition, we use a multiscale approach where we add 3 more input channels composed of compressed patches that are originally 5 times larger than the above set of patches resized to the same size to provide global information about the surroundings of concerned voxel.

Each patch is fed into a different input channel of the Convolutional Neural Network with one or more convolutional layers. The outputs of those channels are then feed as

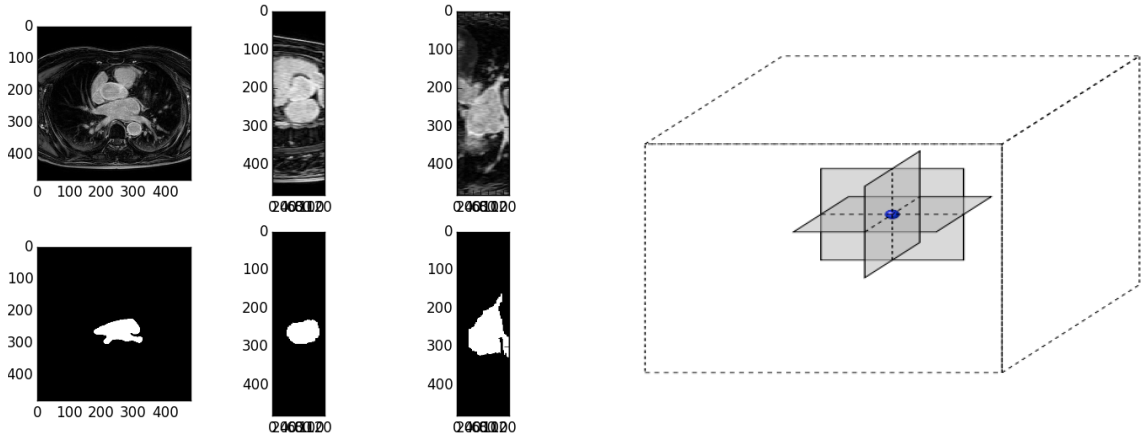

Figure 3.1: Left: grey scale slices from a CT scan taken in the transversal, saggital and coronal planes. Right: illustration of the triplanar

input to a number of connected layers before a classifying layer as output layer.

3.3 Implementation Details

3.3.1 Libraries

We use Torch, an open-source library maintained by Facebook in Lua aiming to provide a Matlab-like environment for scientific computing, along with a number of dependent libraries (cunn, cudnn, fbcunn) to train the convolutional networks on single and multi-GPUs. In addition, we use Python and a number of its libraries to handle all the logistics from generating the datasets to producing plots of segmentation results.

3.3.2 Computer Power

The training of the CNNs were conducted on 2 multi-GPU clusters, named montana-nvidia and montana-k80, kindly provided by Prof. Montana's laboratory. montana-nvidia consists 24 cores with 129 Gb of memory connected to two NVIDIA Tesla K40m and two Tesla K20Xm while montana-k80 on the other hand has 56 cores with 258 Gb of memory supported by 8 of NVIDIA's Testla K80.

3.4 Model Selection

3.4.1 General approach for model selection

To look for a good model, we allocated 20 of the 27 CT scans to the training set, 7 to a testing set. We generated 400000 training examples equally divided among the training CT scans, half being in the atrium and the other half in the non-atrium part. The validation set is composed of 15000 testing examples from each testing CT scans

to a total number of 105000 testing examples. The model selection is done by comparing the dice coefficient on the testing set calculated at the end of end epoch, which is just the classification accuracy in a 2-class problem, segmentation results from a few slices taken from the testing set and a final dice coefficient of the classification accuracy of the segmentation of the entire 7 testing CT scans preformed at the end of training.

We start off our model selection with a CNN composed of 2 convolutional layers with 32 and 64 feature maps respectively, 2 fully connected layers with 1000 and 500 hidden units each respectively, a logsoftmax output layer giving log probabilities. We will be varying in turn the following hyperparameters while keeping the rest constant.

- varying the number of convolutional layers.
- varying the number of connected layers.
- varying the number of feature maps in the chosen number of convolutional layers.
- varying the number of hidden units in the chosen number of connected layers.
- varying the type of activation function (ReLU, Tanh or Sigmoid).
- varying the type of subsampling function (Maxpooling or average pooling).
- varying the learning rate.
- varying the momentum.
- varying the dataset size for the selected model.

3.4.2 varying number of convolutional layers

We start our model selection by the selecting the number of convolutional layers using four architectures with every other hyper parameter remaining the same. Their convolutional layers are given as

- Input (6*32*32) => Conv layer (32*28*28) => 2*2 MaxPooling filter (32*14*14)
- Input (6*32*32) => Conv layer (32*28*28) => 2*2 MaxPooling filter (32*14*14) => Conv layer (64*10*10) => 2*2 MaxPooling filter (64*5*5)
- Input (6*32*32) => Conv layer (32*28*28) => Conv layer (32*24*24) => 2*2 MaxPooling filter (32*12*12) => Conv layer (64*8*8) => 2*2 MaxPooling filter (64*4*4)
- Input (6*32*32) => Conv layer (32*28*28) => Conv layer (32*24*24) => 2*2 MaxPooling filter (32*12*12) => Conv layer (64*8*8) => Conv layer (64*4*4) => 2*2 MaxPooling filter (64*2*2)

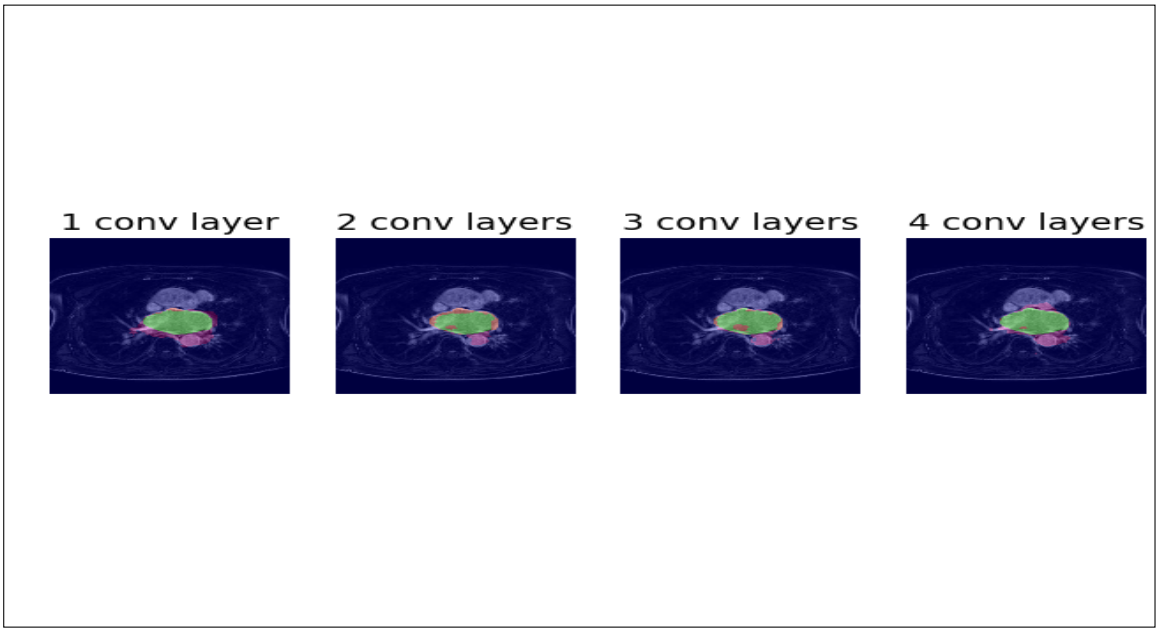

Figure 3.2: Left: grey scale slices from a CT scan taken in the transversal, saggital and coronal planes.

These convolutional layers are then followed by two fully connected layers with 1000 and 500 hidden layers each. The following table gives the results for each of these architectures trained after 100 epochs.

	Dice training set	Dice testing set	Sensitivity	Specificity	Dice test CT scan
1 Conv L	0.959	0.956	0.913	0.970	0.969
2 Conv L	0.958	0.961	0.734	0.989	0.984
3 Conv L	0.956	0.960	0.762	0.987	0.983
4 Conv L	0.959	0.956	0.881	0.973	0.972

The dice coefficient over the training and testing sets are very similar while for the dice coefficient over the test CT scan, there is a significant improvement from 1 convolutional layer on the one hand to 2 and 3 on the other. This is explained by the increase specificity despite a significant drop in the sensitivity as the non-atrium voxels outnumber by a factor of almost 10 the number of atrium voxels.

Figure whatever brings a visual element to this comparison. Each column shows the segmentation of a transversal slice of 3 different CT scan images. The first column corresponding to segmentations by the architecture with 1 convolutional layer as much larger pink patches, corresponding to false positives than the other two columns, corroborating the story told by the dice coefficients. There is not much difference between the masks

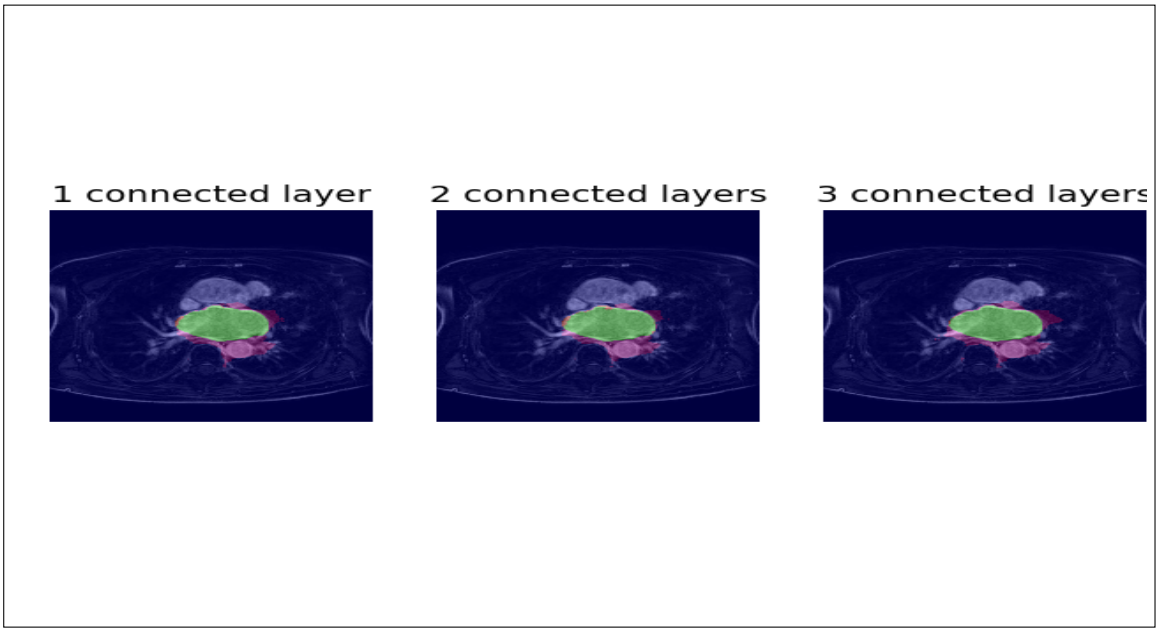

Figure 3.3: Left: grey scale slices from a CT scan taken in the transversal, saggital and coronal planes.

of the second and third layers.

On that basis we choose to go with having 2 convolutional layers.

3.4.3 varying number of connected layers

Having settled on an architecture with 2 convolutional layers, we now train 3 CNNs with 1, 2, and 3 fully connected layers each starting with 1000 hidden units and halving the number of hidden units for each additional layer. Hence the CNN with 3 fully connected layers has 1000, 500, and 250 hidden units in each respectively. Every other hyper parameter remain the same. The results are shown in the following table.

	Dice training set	Dice testing set	Sensitivity	Specificity	Dice test CT scan
1 Connected L	0.957	0.959	0.887	0.972	0.971
2 Connected L	0.958	0.961	0.856	0.974	0.972
3 Connected L	0.953	0.951	0.915	0.967	0.966

The results for 1 and 2 fully connected layer architecture are very similar with very similar sensitivity and specificity and thus Dice coefficient for the test CT scan. However for 3 connected layers, there is a slight decrease in test CT scan dice coefficient from a comparatively lower specificity. Figure whatever shows very similar masks for the 1 and 2 connected layers and larger errors for the third column.

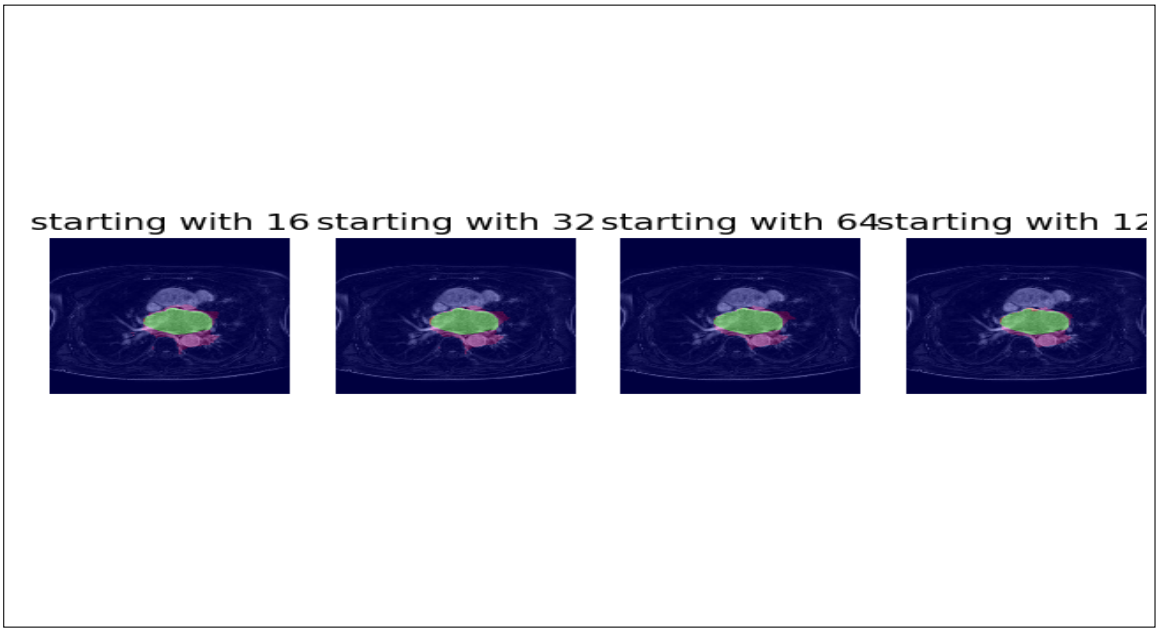

Figure 3.4: Left: grey scale slices from a CT scan taken in the transversal, saggital and coronal planes.

We choose to have 1 fully connected layer for our final architecture.

3.4.4 varying number of feature maps

So now the focus is on finding the right number of feature maps for the convolutional layer. We tried 16/32/64/128 and it seems having 64 feature maps gives the best bang for the buck.

	Dice training set	Dice testing set	Sensitivity	Specificity	Dice test CT scan
With 16 FM	0.956	0.953	0.919	0.967	0.966
With 32 FM	0.957	0.958	0.894	0.972	0.970
With 64 FM	0.959	0.958	0.904	0.971	0.970
With 128 FM	0.962	0.964	0.875	0.976	0.974

3.4.5 varying the number of hidden units

We now vary the number of hidden units in the connected layer, trying 100, 200, 500, 1000 and 1500. 200 is best...

3.4.6 varying the activation function

We experimented with different types of activation function: ReLU, Tanh, Sigmoid. Doing it with ReLU is better...

3.4.7 varying the type of pooling

We also tried different types of pooling: Max pooling and average pooling. No difference, so we stuck with Max pooling

3.4.8 varying the learning rate

Tried different learning rates: 0.01, 0.05, 0.1, 0.5, 1. It seems that 0.1 is best. Having a learning rate of 1 doesn't train the network. We omitted the results.

3.4.9 varying the momentum

Tried different momentums: 0, 0.01, 0.05, 0.1. Having momentum actually slightly improves the test results!

3.4.10 varying the data size

Tried a number of dataset sizes: 400000, 1000000, 3000000: No improved result from the increased data size.

3.4.11 varying datasets type

The first thing we investigate is the sampling method for obtaining the training set. In order to increase the number of non-atrium training examples lying near the boundaries where we expect most of the classification errors to lie, we construct a rectangular area which contains the atrium. The atrium box is constructed by going through all the coordinates of the voxels labeled as being in the atrium, picking the minimum and maximum values in each of the coordinate planes, and possibly adding some padding, this procedure gives us a box containing the atrium.

We train our base CNN on 3 sampling procedures with no atrium box, i.e. all the non-atrium training examples are sampled randomly uniformly, a small atrium box constructed by the procedure above with a padding of 5 pixels in the x and y coordinate directions and of 1 pixel in the z coordinate direction, and finally a large atrium box with a padding of 30 pixels in the x and y coordinates and of 5 pixels in the z coordinate.

The results of the three training runs are shown in Figure whatever. From the testing dice coefficient plot, we get a better classification rate with sampling using an atrium box than no atrium box and particularly with the smaller atrium box. In the segmentation mask, sampling with no atrium box clearly yields more errors in the proximity of the atrium but none away from it whereas there are some errors far away from the atrium from the models trained with the atrium box sampling procedure. This is a consequence of the sampling procedures in both cases. Sampling with an atrium box would naturally yield better segmentation results near the atrium as the proportion of training examples is much higher in those regions than without an atrium box around it.