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

21st 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.

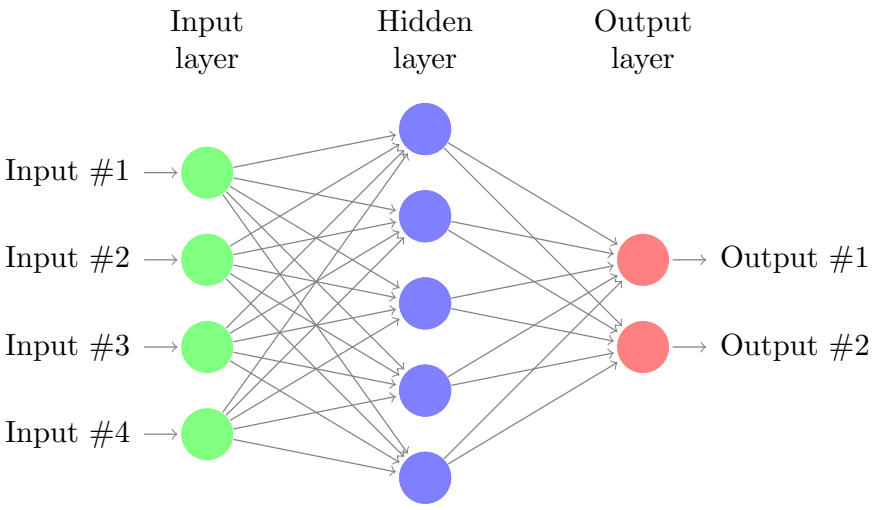

Figure 2.1: A 2 layer feedforward neural network with 4 input nodes, 5 hidden nodes and 1 output node.

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.

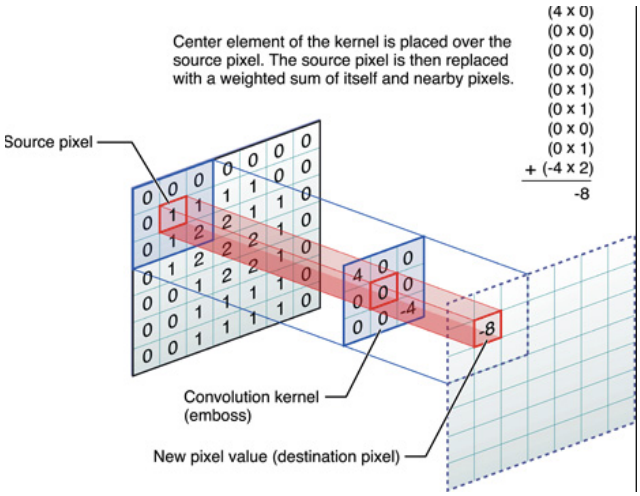

Figure 2.2: The convolution layer. Illustration of the formation of a feature map.

- 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.

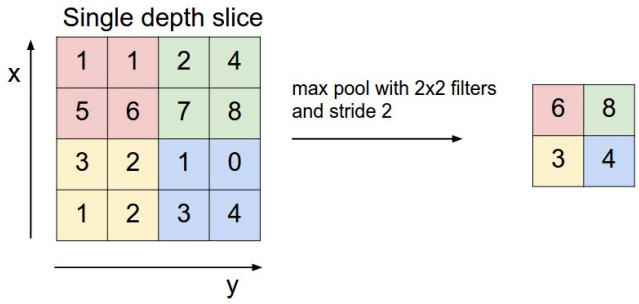

Figure 2.3: Subsampling in action. Here maxpooling is used to halve the size of an feature map using a 2×2 filter

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.

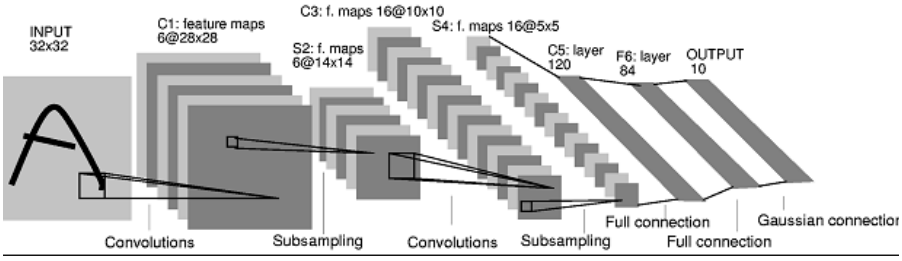

Figure 2.4: A typical convolutional network architecture with 2 convolutional layers, 2 fully connected layer and a Gaussian connection as output layer.

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

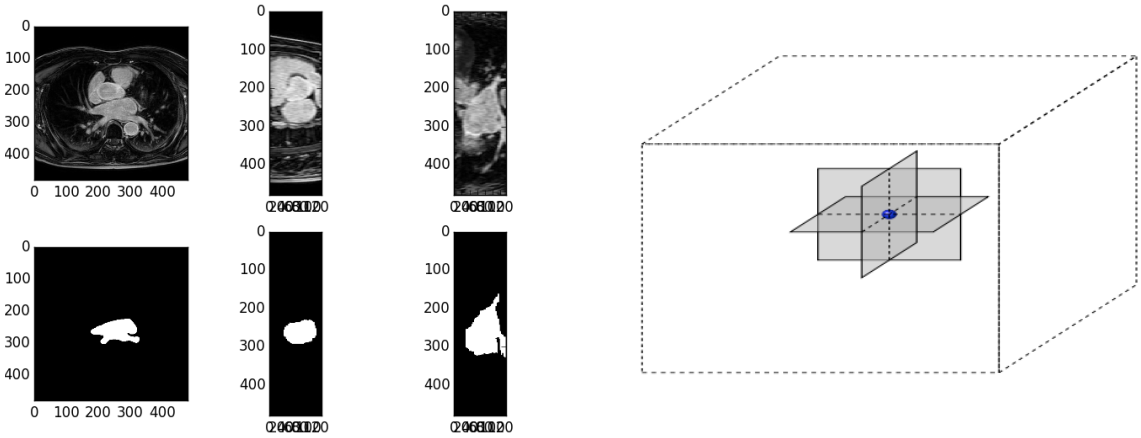

Figure 3.1: Left: grey scale slices from a CT scan taken in the transversal, saggital and coronal planes. Right: illustration of the triplanar

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 Hyper-parameter search

3.4.1 General approach to looking for a good model

How did I generate the training and testing datasets? Describe the base model. Describe how I am undertaking the hyper-parameter search. Describe the dice coefficient.

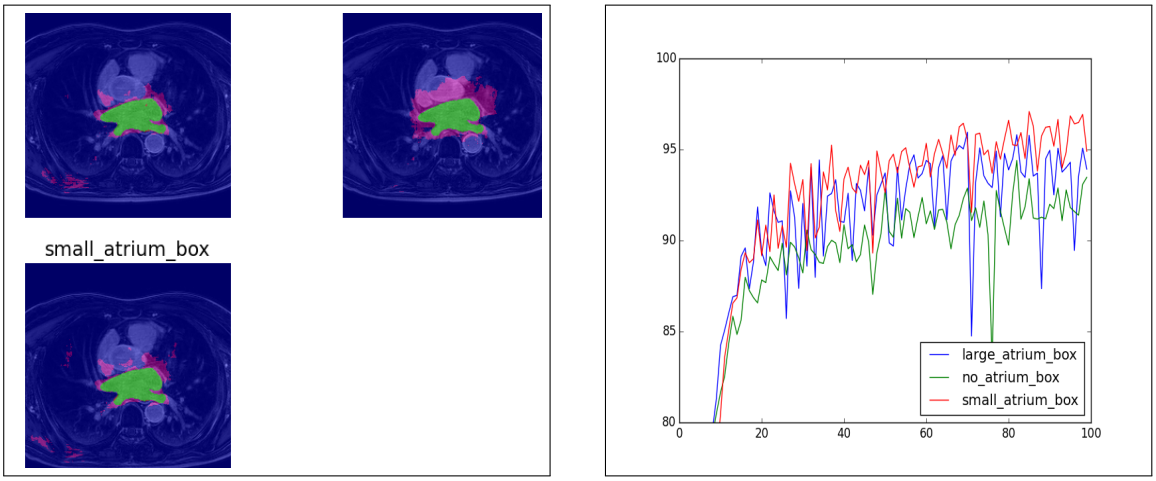

Figure 3.2: Left: grey scale slices from a CT scan taken in the transversal, saggital and coronal planes. Right: illustration of the triplanar

3.4.2 varying datasets type

Describe the atrium box approach. Conjecture that the improved results were because of a higher concentration of training examples in the boundary regions.

3.4.3 varying number of convolutional layers

We tried 3 different architectures with 1, 2, 3 convolutional layers while keeping all the other parameters the same. We find that having more than 1 doesn't improve the results much. Of note that I have no max pooling in the second convolutional layer to not decrease the number of features too much.

3.4.4 varying number of connected layers

Having settled on an architecture with 1 convolutional layer, we trained 3 CNNs with 1, 2, and 3 connected layers. Having more than 1 connected layer doesn't make a difference.

3.4.5 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.

3.4.6 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...

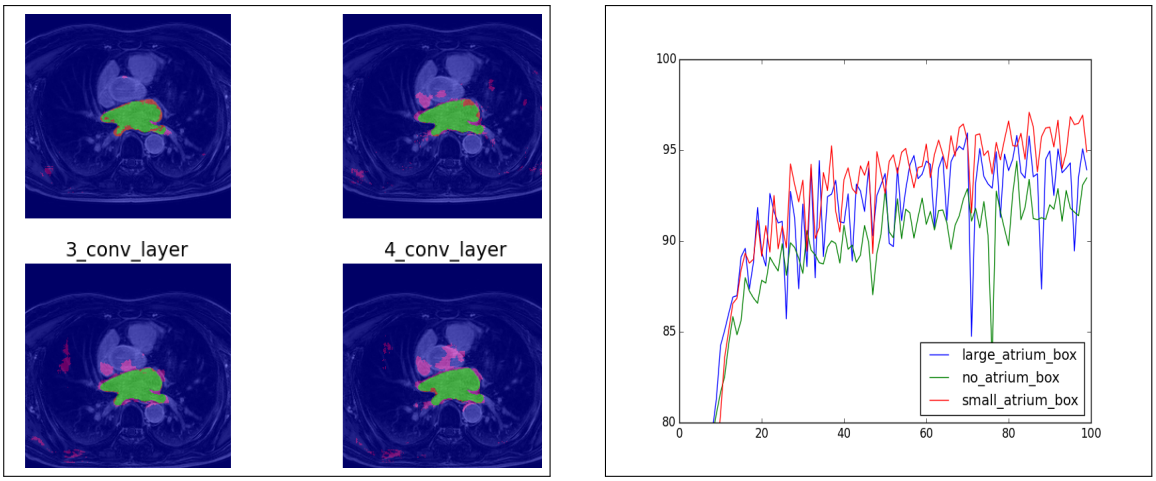

Figure 3.3: Left: grey scale slices from a CT scan taken in the transversal, saggital and coronal planes. Right: illustration of the triplanar

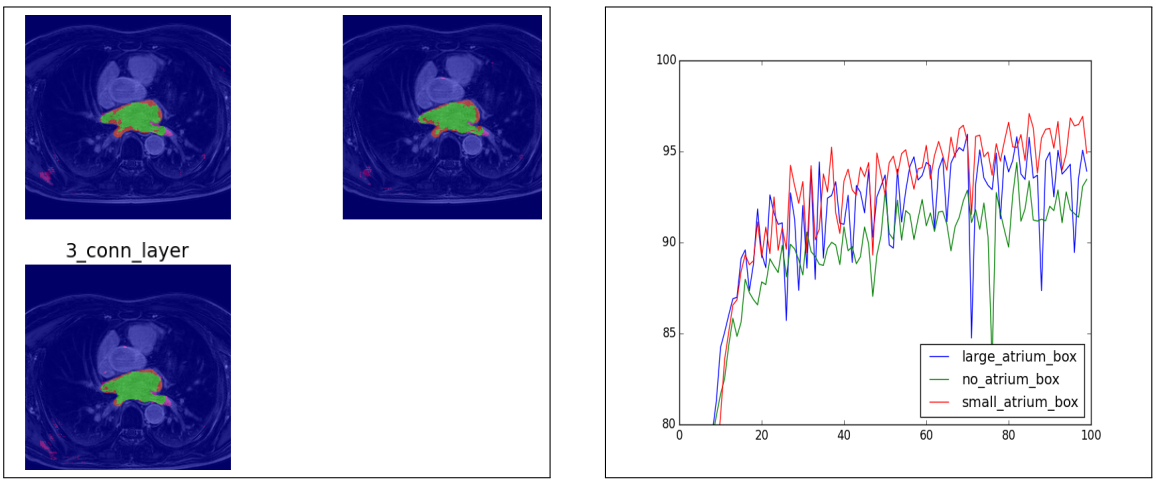

Figure 3.4: Left: grey scale slices from a CT scan taken in the transversal, saggital and coronal planes. Right: illustration of the triplanar

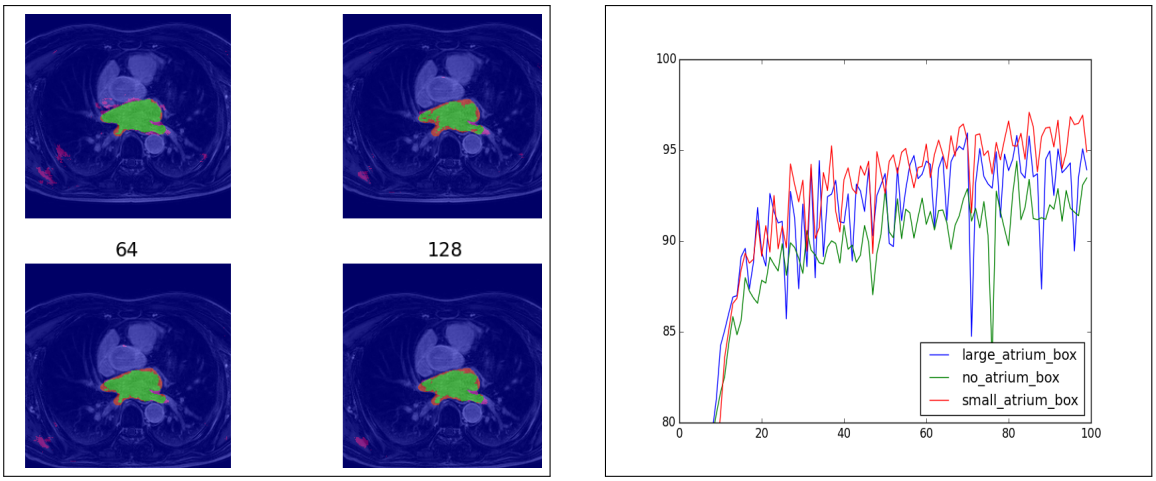

Figure 3.5: Left: grey scale slices from a CT scan taken in the transversal, saggital and coronal planes. Right: illustration of the triplanar

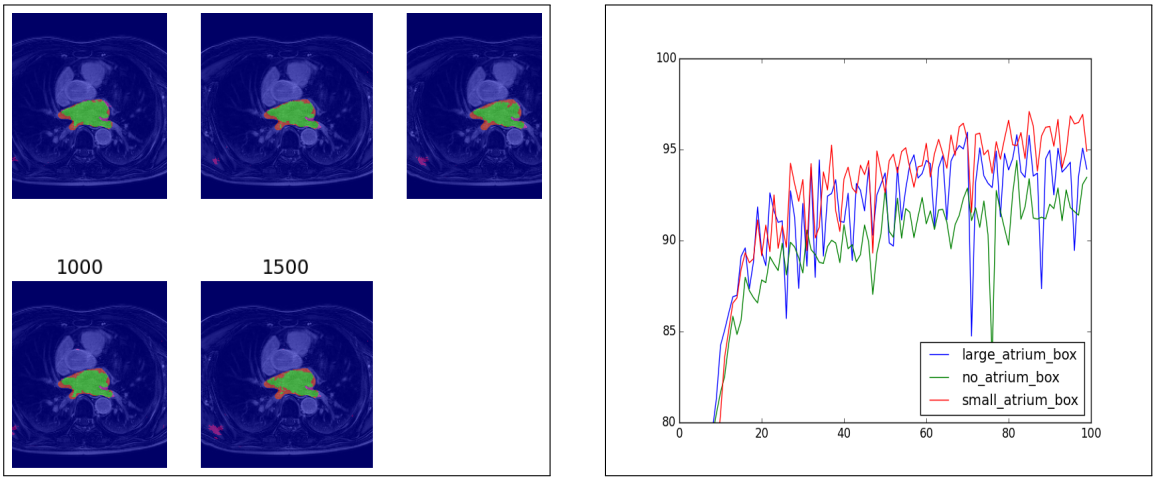

Figure 3.6: Left: grey scale slices from a CT scan taken in the transversal, saggital and coronal planes. Right: illustration of the triplanar

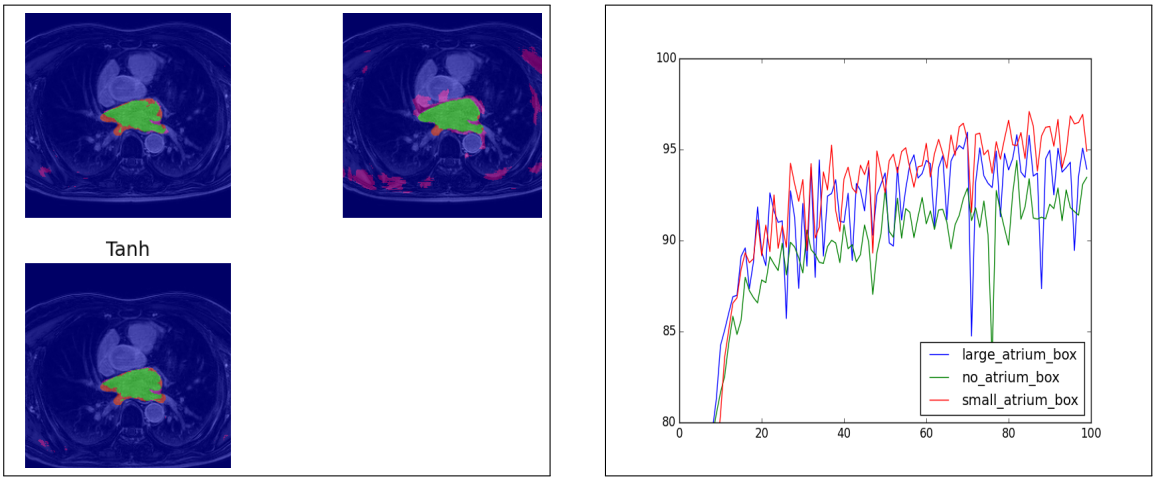

Figure 3.7: Left: grey scale slices from a CT scan taken in the transversal, saggital and coronal planes. Right: illustration of the triplanar

3.4.7 varying the activation function

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

3.4.8 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

3.4.9 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.

3.4.10 varying the momentum

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

3.4.11 varying the data size

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

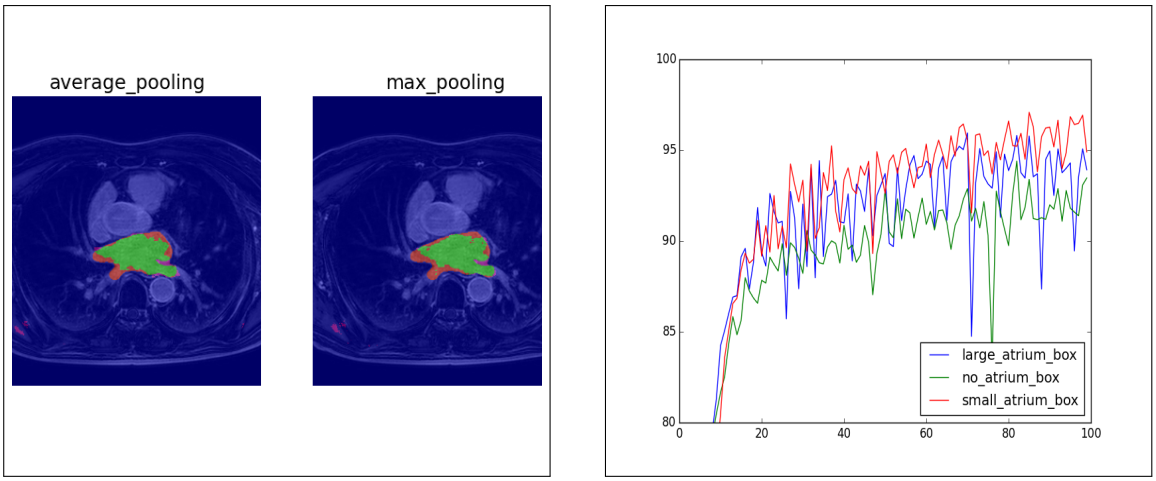

Figure 3.8: Left: grey scale slices from a CT scan taken in the transversal, saggital and coronal planes. Right: illustration of the triplanar

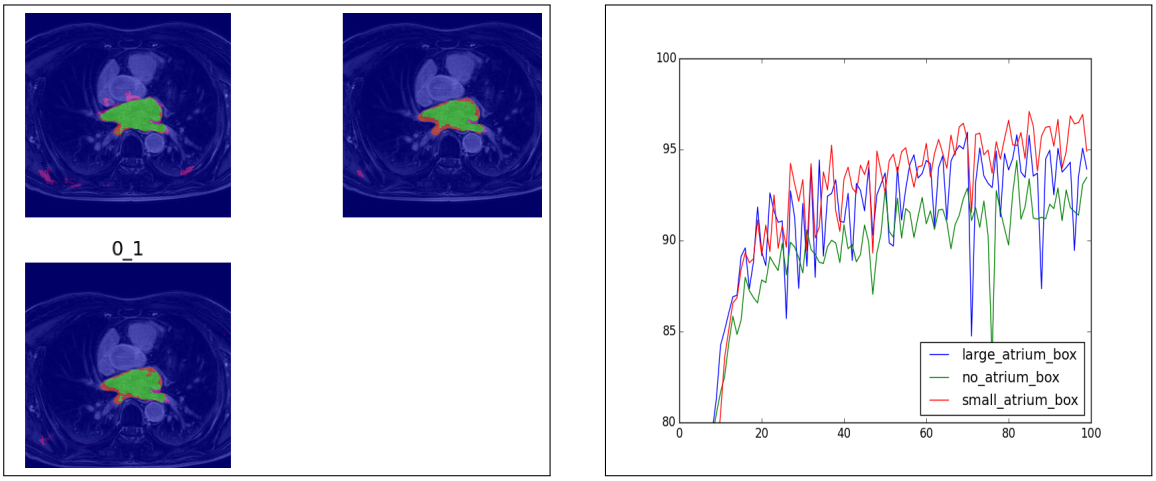

Figure 3.9: Left: grey scale slices from a CT scan taken in the transversal, saggital and coronal planes. Right: illustration of the triplanar

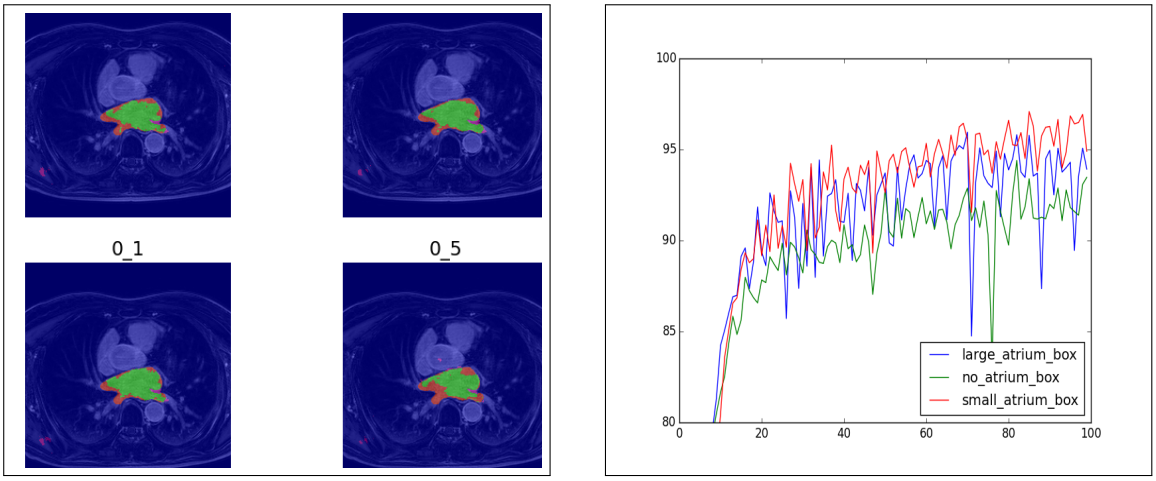

Figure 3.10: Left: grey scale slices from a CT scan taken in the transversal, saggital and coronal planes. Right: illustration of the triplanar