Project for Deep learning in medical imaging: Segmentation - MIC

Task 1: Obtain, Load & Visualize Datasets

Hendrik Schick, Tobias Dorra

December 18, 2019

1 Task

The task was to load and visualize the given datasets.

1.1 The datasets

This project focuses on two datasets, one of them is focused on the liver, the other one on the prostate. Both datasets have the same format.

Each dataset consists out of three folders:

imagesTr The volumetric images that shell be used for training.

labelsTr The segmentation masks for the images in imagesTr.

imagesTs Images for testing.

All images are in the NIfTI-1 Data format (file extension .nii.gz).

2 Implementation

2.1 Loading

We used the function nibabel.load() from the python library nibabel to read the image files.

This gave us the image data in form of three dimensional numpy arrays.

Additionaly to that, we noticed, that the resolution of the images is different allong the different axises. So we needed the correct scaling factors to display the images without distortions. Nibabel returns an affine transformation matrix alongside the pixel data, that transforms pixel-coordinates to world-space coordinates. The elements on the diagonal of this matrix is what we used as scaling factors.

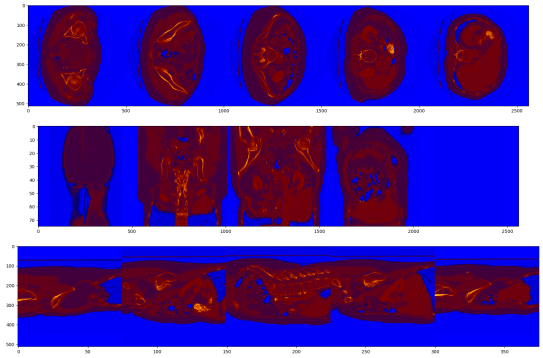

Figure 1: Various image slices in x, y and z direction

2.2 Visualisation

For visualisation, the images were pre-processed using numpy and then shown using matplotlib.

Since the images are volumetric, we decided, to show slices of the images. We extracted slices in uniform intervals, along all three axises.

In terms of colouring, there are two options that both were implemented:

1. Colour only based on the density value inside of the input image. We choose a colour palette, that starts at blue, progresses to red and ends at yellow, to have a better contrast compared to just showing the raw grey scale image. See figures 1 and 2 for examples.

2. Just draw the image in greyscale, but overlay it with colours based on the values from the segmentation mask. See figures 3 and 4 for examples.

We also experimented with rendering the whole volume at once using the emission absorption model, instead of showing individual slices. When using the emission absorption model, each 'layer' of the image can emit some light and also absorb a part of the light that comes from the layers behind it. How much light is emitted or absorbed is controlled by the image data. You can iterate over the layers from back to front and accumulate the amount of light that leaves each layer.

However, the results of this were not as detailed as we hoped for, you can make out way more details in the simple slices described above, so we went with the slices-approach instead. However, you can still see an example image of how it looks in figure 5.

3 Problems and future tasks

There were no major problems with loading or visualizing the dataset. The next step will be to use the datasets for training an actual model. Therefore, we will

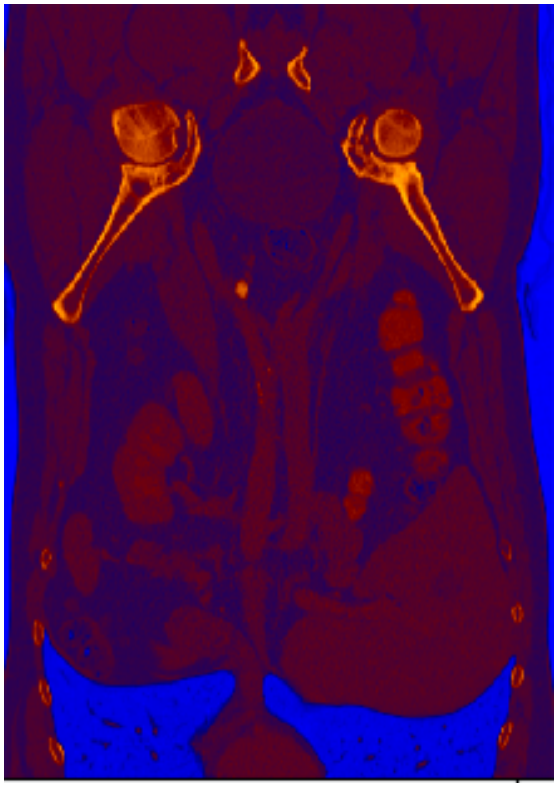

Figure 2: Detail: Colouring of the images

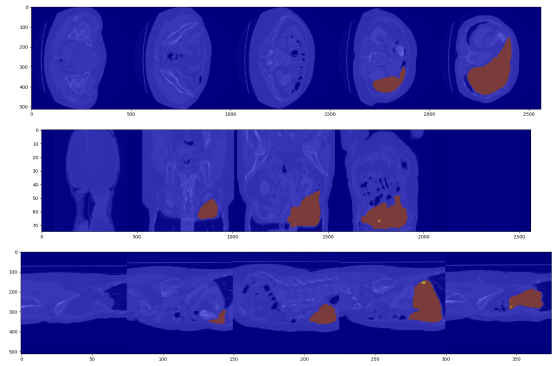

Figure 3: Various image slices in x, y and z direction with a segmentation mask

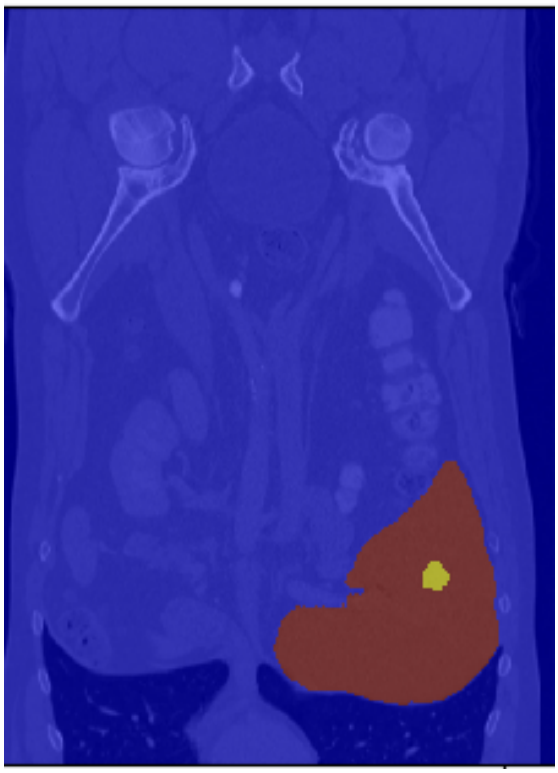

Figure 4: Detail: Colouring of the segmentation mask

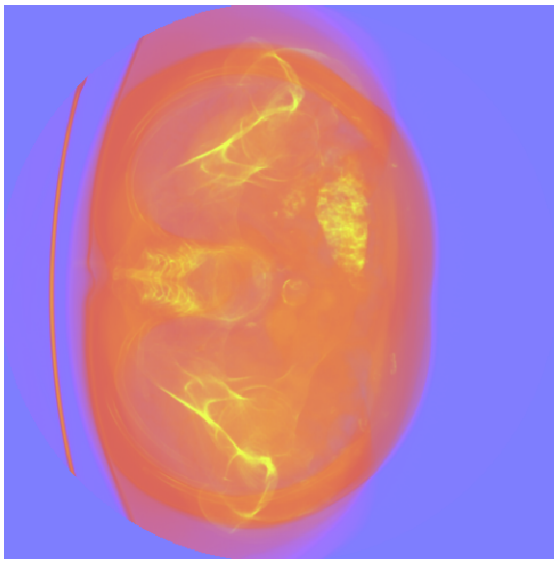

Figure 5: Display of the complete volume at once using the emmission absorption model.

probably need to do some preprocessing. For the sake of visualizing the data, we just used the aspect parameter of the imshow function from matplotlib to account for the different image resolutions per axis mentioned in section 2.1. For training an actual model, we probably will have to actually stretch the image according to those scaling factors. Otherwise, the model would be dependant on the (x/y/z-)resolutiion of the images that were used for training.