2017-07-03

- Initial set up of neural net, based on leNet-5, no signal out, most likely due to using unprocessed data.
- The data is very padded, this has been reduced with the function `cropHeart(inp)`, but I will need to make sure all the files are the same size before they get fed into the CNN.
- I could try normalising the data to get a signal, but will need to get the unpadded data working first.

2017-07-04

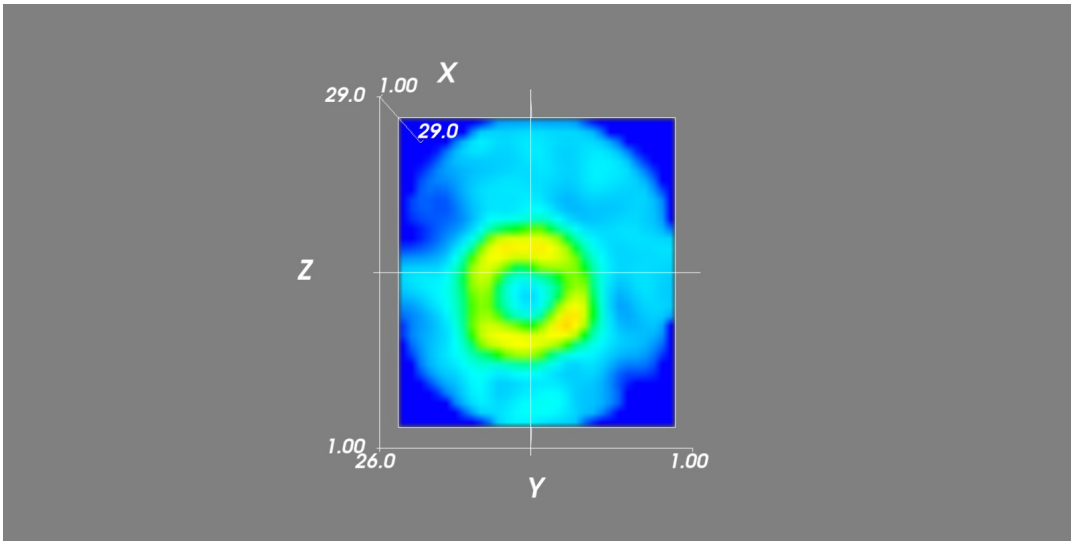
- Wrote a visualisation of the data (`visualisation.py`).
- Still working on repadding the cropped data (It's a bit of a pain).

2017-07-05

- Repadded the cropped data, it is now of size [68,34,34].
- Retrying the CNN with the new data doesn't get a signal. Maybe there isn't enough data to make it work?
- I will fiddle with the hyperparams to see if I can pick something up.
- Maybe normalising the data will help.

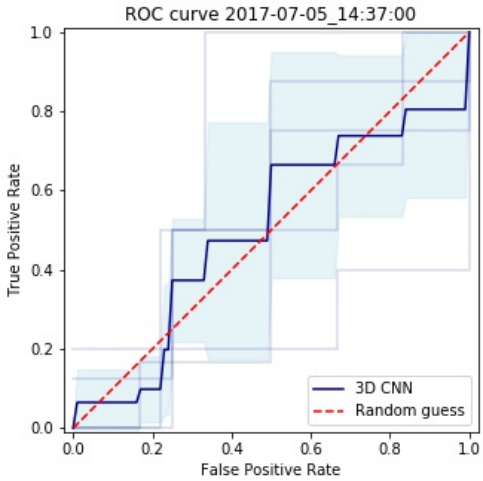
Got some results using 2D slices:

- The 2D slices I used look like:

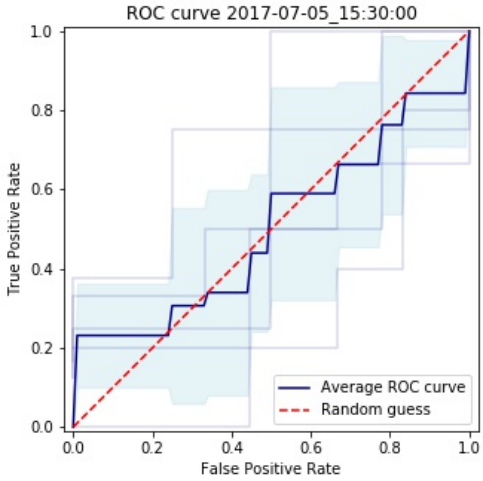


- I have used 2d slices of the data and it works well (halfway through the z-axis). It uses:
    - Slice of rest and slice of stress on z axis. Spacial x and spacial y on x and y axes.
    - LeNet-5 CNN with 3D convolution and subsampling.
    - [2,5,5] filters, pooling 2 with step 2.
    - learning rate of 0.0001, with ADAM optimiser, and batch size of 10.
    - After 50 epochs of 58 images it learns to ~95%.
- I will now apply a k-fold x-validation to it to see if it's not just picking up noise.

- The k=10-fold x-validation shows that the CNN is learning the noise in the data, although this could be due to the small amount of images in each k-fold (only 6!):



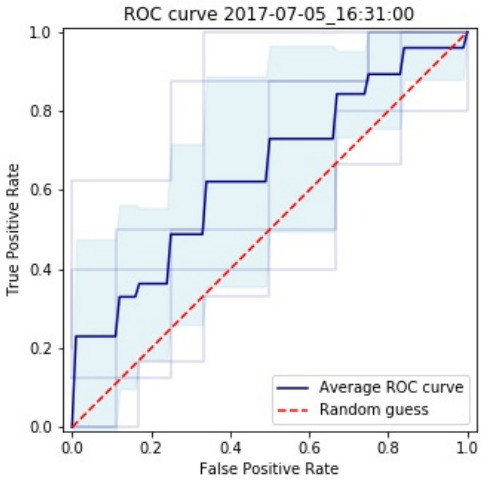
- I tried normalising the arrays, with no luck. It stopped overfitting the data, but still hasn't learnt significantly:

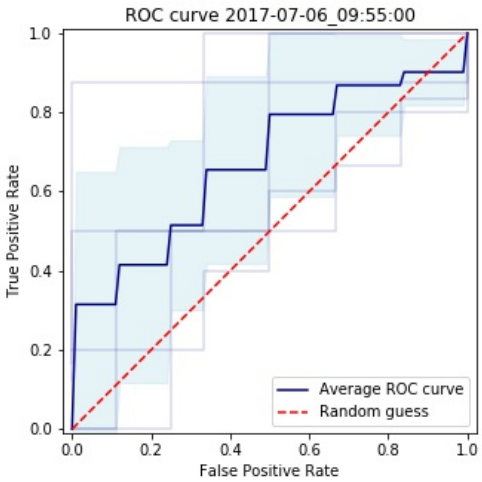


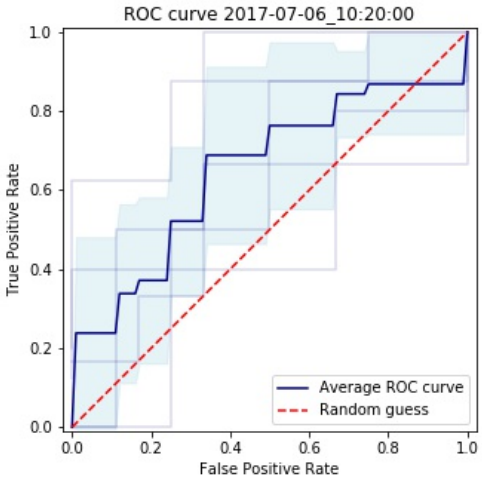
- I think the issue is still the massive amount of blankspace. I should try and scale the arrays so that they are the same size.

Have a signal!

- I have got a signal with the following CNN:
    - Slice of rest and slice of stress on z axis. Spacial x and spacial y on x and y axes.
    - LeNet-5 CNN with 3D convolution and subsampling.
    - [2,10,10] filters, pooling 2 with step 2.
    - learning rate of 0.0001, with ADAM optimiser, and batch size of 10.
    - 5 k-folds.
    - After 50 epochs of 47 images it learns training data to ~95%.
    - Over three repeats:
        - Avg Spec: 0.583, 0.623, 0.663
        - Avg Sens: 0.633, 0.683, 0.700
    - ROC curves:







2017-07-06

- I redid the 2D slice data with three slices along the x, y, and z axes. It will take ~100 mins to finish learning. It's probably time to use some better hardware.
- Found a function [(https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.ndimage.interpolation.zoom.html)](https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.ndimage.interpolation.zoom.html) which should work well for resizing the images.

- Maybe the reason that the slicing works, and the 3D data doesn't is because the CNN filter only sees one 3D image at a time, and sees both the rest and stress images at the same time in the 2D slice data. I could write a 4D CNN to fix this.
    - mhuen seems to have written a 4D convolution by stacking 3D CNN outputs (https://github.com/mhuen/TFScripts/blob/master/py/tfScripts.py). This might work for what I want to do, and stacking can be used for pooling too.
- I wrote a scaling function hat eliminates most of the whitespace. After training the CNN did not learn significantly.
- Added a ROC AUC calculator to the outputs.
- I'm going to try artificially expanding the data.