## 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

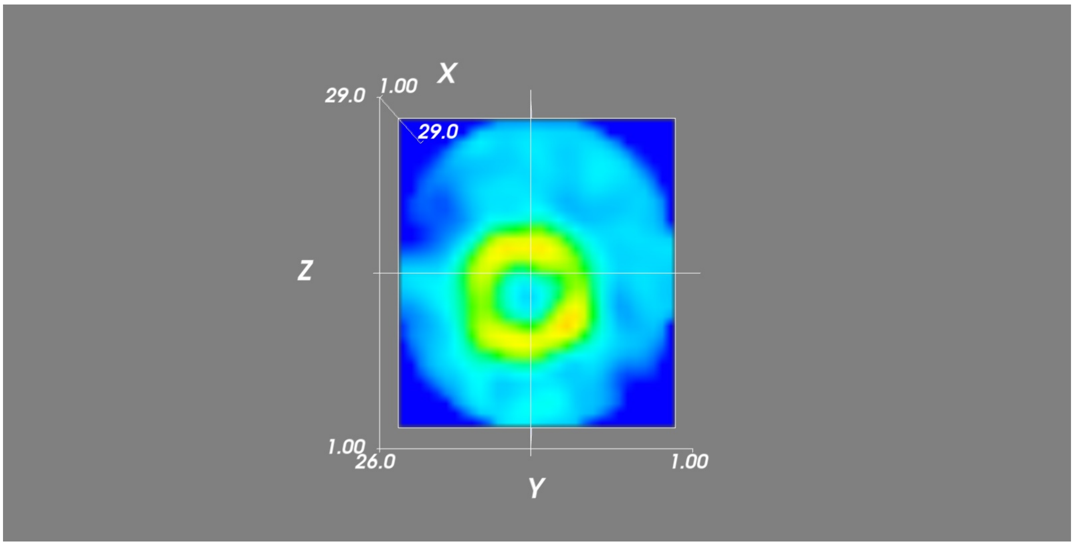
- Wrote a visualisation of the data (`visualisation.py`).
- Still working on repadding the cropped data (It's a bit of a pain).

## 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.

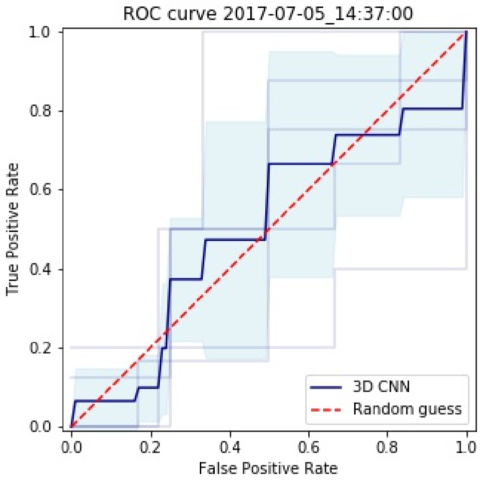
**Got some results using 2D slices:**

- The 2D slices I used look like:

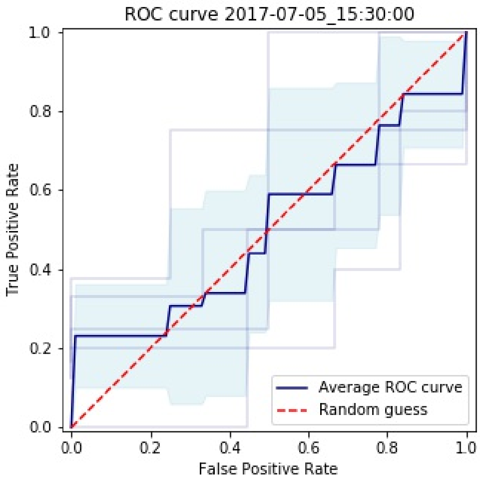



- I have used 2d slices of the data and it works well (halfway through the z-axis). It uses:
    - Slice of rest and slice of stress on z axis. Spacial x and spacial y on x and y axes.
    - LeNet-5 CNN with 3D convolution and subsampling.
    - [2,5,5] filters, pooling 2 with step 2.
    - learning 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:

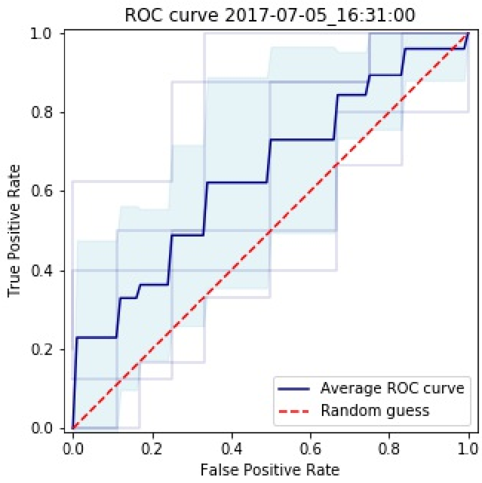


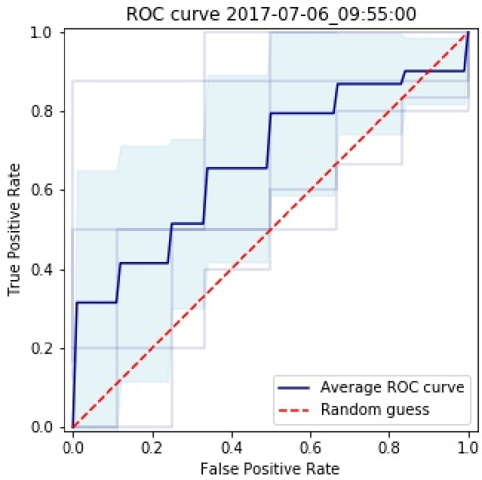

- I think the issue is still the massive amount of blankspace. I should try and scale the arrays so that they are the same size.

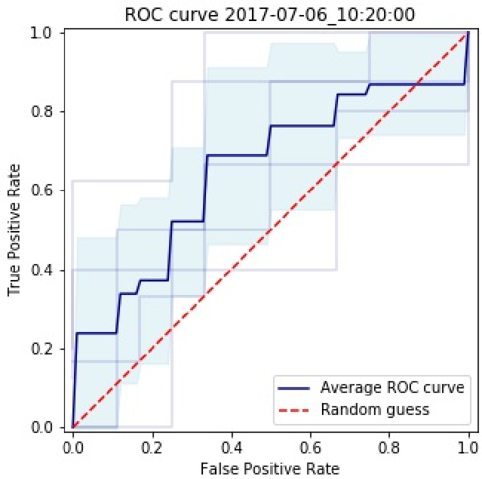
**Have a signal!**

- 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)](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.

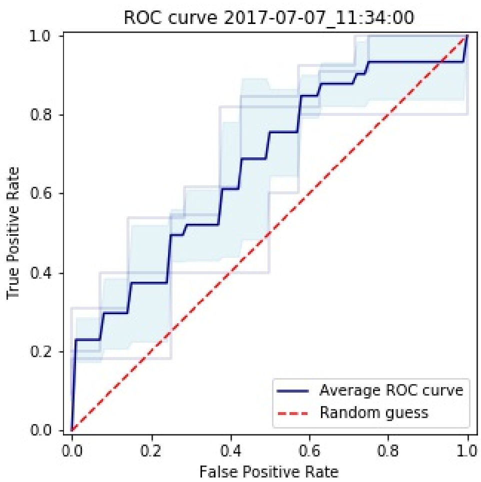
## 2017-07-07

- Because of the overfitting going on when running the CNN, I increased the L2 regularisers' weight decay from 0.001 to 0.01, and added an extra dropout layer between the two FC dense layers.

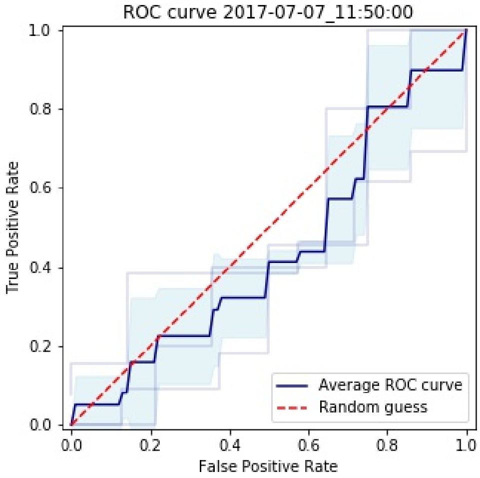
- Can't seem to get any results with a spec/sens over 60%, probably due to the way I'm orgainising the data.

- The CNN appears to train better when using non-scaled data. I can't figure out why. Maybe it's using the image sizes as an aid?

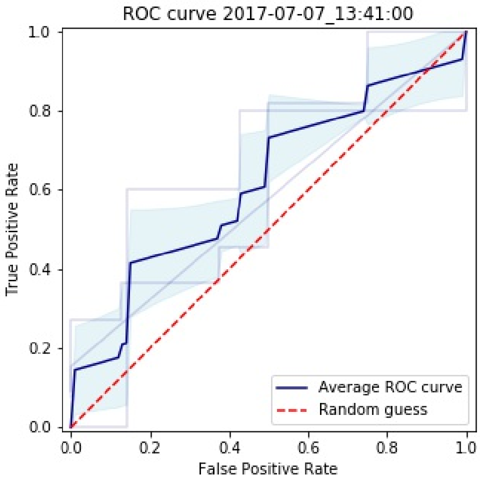
    - Conv filter: [2,15,15]; pool filter: [2,2,2]; 2 FC 1024 neurons, L2 regulisation at weight decay = 0.001, dropout at 0.5 after each FC layer; ADAM optimiser, learning rate = 0.0001, categorical x-entropy loss; batch size = 10 at 38 datum; k = 3 folds.

        - non-scaled: 

- scaled:



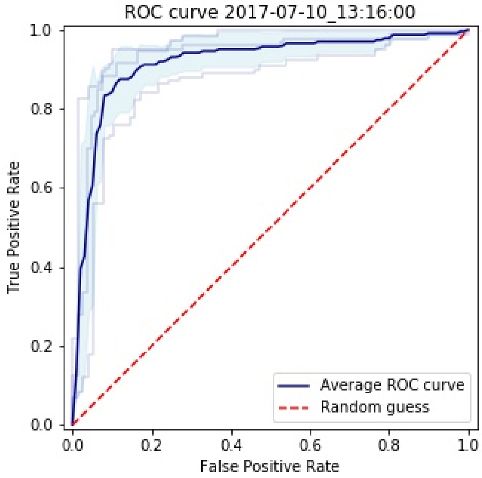

- scaled, not renormalised:

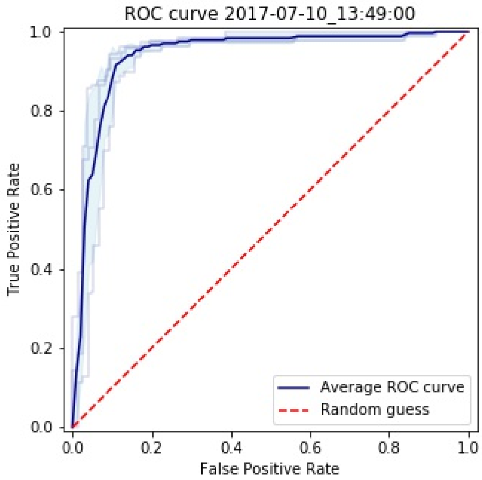



- As shown in the ROC curves, the only data that is causing consistent learning is the non-scaled one. I don't know why.
- Rewrote heart_data.ipynb so that it can resize the input data.

## 2017-07-10

- It might be better to use a Siamese CNN instead of a 4D CNN to compare two 3D images, as training will be faster.
  - I have written CNNs using two-channel, and Siamese architectures, along with the OG 3D convolution architecture. The two-channel and Siamese architectures are described here: [(https://arxiv.org/pdf/1504.03641.pdf)](https://arxiv.org/pdf/1504.03641.pdf).
- The use of a very deep NN architecture would reduce linearity, and may be useful.
- Artificially expanding the data seems to have worked. I am getting after k = 3 folds (100 epochs) at 619 datum (two runs):
  - Spec: 0.864, 0.917
  - Sens: 0.888, 0.883
  - ROC AUC: 0.918, 0.940
  - This is with the two-channel architecture. ROC curves:

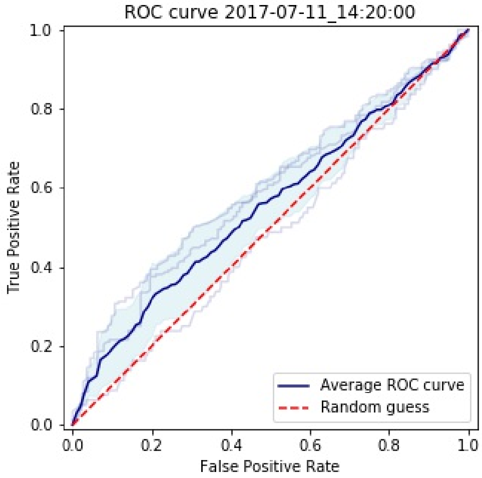
- 


- 


- Haven't got any significant results from the Siamese CNN, but have only trained it to ~30 epochs. It will probably need more training than the two-channel as there are nearly twice as many weights in the Siamese CNN.
- I should try validating the CNN on ppts that it hasn't seen before (like take 10 ppts from the pool before artificial expansion and use these to validate).

# 2017-07-11

- I have separated ppts into different k-folds before expansion, so each k-fold has unique ppts in it now, even after artificial expansion. We'll see how it performs now... (This is in the 2channel ipynb)
    - It doesn't work very well. Getting ~50% accuracy.

- 


- More data would be helpful to reduce overfitting, but using all three dimensions of the heart data may be enough to get "good enough" results.
- I have written a 2 channel CNN for the 3D data. It should be ready to try on the supercomputer.

# 2017-07-12

- Testing the 2dSiameseCNN on the supercomputer:
  `$ qsub -q gpu -l nodes=1:ppn=16 -I -X -l walltime=24:00:00`