UNIVERSITY OF WARWICK

Luna 16 False Positive Reduction - Algorithm Description

Matt Smith

April 4, 2016

0.1 MODEL

We use a 3D convolutional neural network using Torch 7. The architecture is based on 3 seperate parallel networks, each of which take in a different resolution/scale of the 3D image. The 3 models each output a single node which are then put through a final linear layer and sigmoid to produce a number in $[0,1]$. A torch overview of the first of the three "mini" networks is given below;

```
>> print(model:get(1))
    input
      |`-> (1): nn.Sequential {
      |       [input -> (1) -> (2) -> (3) -> (4) -> (5) -> (6) -> (7) -> (8) -> (9) ->
             (10) -> (11) -> (12) -> (13) -> (14) -> (15) -> (16) -> (17) -> (18) ->
            (19) -> (20) -> (21) -> (22) -> (23) -> (24) -> output]
      |       (1): nn.VolumetricBatchNormalization
      |       (2): nn.VolumetricConvolution
      |       (3): nn.ReLU
      |       (4): nn.VolumetricBatchNormalization
      |       (5): nn.VolumetricConvolution
      |       (6): nn.ReLU
      |       (7): nn.VolumetricMaxPooling
      |       (8): nn.VolumetricBatchNormalization
      |       (9): nn.VolumetricConvolution
      |       (10): nn.ReLU
      |       (11): nn.VolumetricBatchNormalization
```

```
|      (12): nn.VolumetricConvolution
|      (13): nn.ReLU
|      (14): nn.VolumetricMaxPooling
|      (15): nn.VolumetricBatchNormalization
|      (16): nn.VolumetricConvolution
|      (17): nn.ReLU
|      (18): nn.VolumetricBatchNormalization
|      (19): nn.VolumetricConvolution
|      (20): nn.ReLU
|      (21): nn.VolumetricMaxPooling
|      (22): nn.VolumetricBatchNormalization
|      (23): nn.View
|      (24): nn.Linear(8064 -> 1)
|    }
```

Convolutional kernels were of $3 \times 3 \times 3$ with a stride of 1 and max pooling layers were also $3 \times 3 \times 3$ but with a stride of 2.

0.2 DATA

As we have three inputs to our algorithm and one output we create a function which takes a scan/candidate and class combination and outputs four tensors. The first three are our inputs to the network and the last is our target.
Each of the three inputs is a 3D array of dimension $42 \times 42 \times 42$ containing 43^3 voxels, which are typically centered on the candidate location and rotated/scaled using 3D interpolation (for data augmentation purposes). The images are clipped between intensity values of $-1300, 1300$ and are also normalized by their scan's spacing values. The data augmentation proves to be useful more so for the positive class due to its low count in the data set. As previously mentioned the 3 inputs differ in their scaling/resolution to gain different views of the candidate to capture different levels of detail and location.

0.3 TRAINING/TESTING

Due to the highly imbalanced nature of the data we sample between the two classes we equal chance throughout training using one example at a time. We use the binary cross entropy loss function and the Adam optimizer.
To speed up training we use multiple threads to load/prepare the data onto a GPU.
Unfortunately due to time constraints we were only able to train for 100,000 examples. Moreover, we used two fold cross validation for the competition as it took a lengthy amount of time just to train/test the model twice, let alone 10 times. That is for each of the two folds we trained on 50% of the data and tested on the remaining 50%.