LUNA16 COMPETITION: FALSE POSITIVE REDUCTION
(PROJECT REPORT: COMPUTER-AIDED DIAGNOSIS IN MEDICAL IMAGING)

Thomas de Bel, Cas van den Bogaard, Valentin Kotov, Luuk Scholten, Nicole Walasek
Radboud University
Data Science, ICIS
Nijmegen

ABSTRACT

We propose a method for automatic false-positive reduction of a list of candidate nodules, extracted from lung CT-scans, using a convolutional neural network. Batch normalisation was applied to reduce overfitting. Data augmentation on the positive set of candidates was used to balance the training set. Training and testing was performed on the LUNA16 competition data set. The resulting AUC score was 0.9944 on average per fold in 10-fold cross-validation.

1. INTRODUCTION

This report covers our work on the third and final phase of the LUNA16 competition, which we entered as part of the course “Computer Aided Diagnosis in Medical Imaging”. The LUNA16 challenge asks the contestants to build a computer aided diagnosis (CAD) system for early detection of lung cancer from CT-scans. The third phase builds on top of and was preceded by the first two phases, in which a segmentation of the CT scans and a set of nodule candidates were generated. The goal of the current phase is false-positive reduction in order to narrow down these sets of candidates and identify the cancerous nodules. To detect the true-positive cases, we built a convolutional neural network (CNN) that takes patches centered on the candidates as input and for each such input predicts the likelihood of belonging to either of the classes, i.e. ”nodule” and ”not-a-nodule”. The performance of the network was measured by calculating the free-response receiver operating characteristic (FROC).

2. DATA

The data set from which the data used in this phase is sourced consists of 888 CT-scans. The slice thickness of the scans ranges between 0.8 and 2.5 mm. The voxel values of the CT-scans are encoded on the Hounsfield Unit (HU) scale. As in the second phase, isotropic resampling is performed on all scans, i.e. the images are transformed such that one pixel uniformly represents one millimeter in all three directions of the scan.

3. METHOD

3.1. Patch Extraction

For this task, the representations we look at characterize each candidate on three geometric planes ((x, y): axial, (y, z): sagittal, and (x, z): coronal plane). From each of those planes, three patches are extracted at three locations on the free axis, with the candidate at the center: the planes with the exact candidate location, as well as 2 pixels/millimeters to both directions on the remaining free axis (i.e. in z, x and y direction). The resulting patches measure 6 × 6 centimeters, centered around the candidate location, and are handled and saved as three-dimensional (3 × 60 × 60) numpy arrays - bearing some structural resemblance of an RGB image. The pixel values are still in Hounsfield units (no loss of fine-grained detail) and the three channels don’t represent different wavelengths but represent spatial differences.

3.2. Data Augmentation

The class frequencies in the data are very imbalanced: there are three orders of magnitude more negative than positive examples. The issue of learning a trivial classifier during the training phase which always assigns the label of the negative class is addressed by applying methods of data augmentation on the positive class, in order to artificially balance the data set. This way, we don’t have to resort to throwing away information on the side of the negative class, and can actually take advantage of the excess of negative training cases. The following augmentations are assumed to preserve patterns that are distinctive of the class labels, i.e. label preserving transformations:

• vertical and horizontal flip
• random cropping to 3 × 52 × 52 (without constraints on the locations of the cropped images within the 60 × 60 source patch, i.e. the candidate center might be located 4, 3, 2, 1 or 0 pixels off the center in the 52 × 52 result)

These augmentations are applied at random to random sequences of the full set of positive instances. This procedure
moreover is repeated several times over. For the negative instances, only random cropping to $3 \times 52 \times 52$ is applied in order to make the patches the same size.

The data augmentation is implemented with the IMAGE-DATAGENERATOR() of Keras [Chollet, 2015], a wrapper for Theano [Theano Development Team, 2016]. This choice comes in handy as it allows for the augmentations to be made on-the-fly, which helps to avoid the need to write many slight variations of a patch to the file system just to load them again into memory for processing, which efficiently and effectively speeds up the whole training procedure. This method has the additional benefit that we can easily yield an equal amount of positive and negative instances in each mini-batch, all sampled from the set of negative instances and the set of augmented positive images.

On a further note, this augmentation pipeline is applied only during training, in order to avoid any chance of the network merely learning to distinguish between artifacts of image transformations instead of actual nodule features. During testing, the average prediction of the four corner $3 \times 52 \times 52$ crops and the center $3 \times 52 \times 52$ crops is taken.

3.3. Patch Sizes

We found that this size of the cropped-out patches was crucial for our network to be able to distinguish true nodules from false positives. We believe that the reason for this lies in the fact that the network exploits the positional relationship of nodules to other (healthy) tissue. When the extracted patches were smaller, all test patches were classified as positive.

3.4. Preprocessing of the patches

Before feeding the training data to the network we subtracted the overall training mean from each sample, divided by the overall standard deviation and did the same for testing. As the data set was too large to compute the mean over all training data we decided to use a reasonably-sized subsample as this should yield a reliable approximation of the true mean.

3.5. CNN Configuration

The network has 3 convolutional layers, with max-pooling layers of size $2 \times 2$ and batch normalization layers after each convolutional layer. The network takes $3 \times 52 \times 52$ patches as input. The configurations of the convolutional layers can be viewed in table 1. For the activation functions we used rectified linear units (ReLU). The last convolutional layer is connected to a dense layer using 512 neurons. We used 2 dense layers in total, where the last layer is connected to a softmax layer for classification. We set the learning rate to 0.01 with a decay of $1 \cdot 10^{-6}$. Our whole architecture is displayed in figure 2. Our chosen architecture is based on the one in [Setio et al., 2016].

3.6. Training and testing

The approach used in training and testing the network is 10-fold cross-validation. The data set was divided in 10 sets of approximately equal size. Of these subsets, 9 are used for training the network. The one subset left is then used for testing the results. This procedure is repeated for a total of 10 times, each time with a different subset as testing set. All training procedures ran for 80 epochs, with a random sample of 20000 negative patches and a mini-batch size of 128. During testing we average over the patches belonging to the same nodule. We also implemented a version where we have 3 networks, all set up as the network described in figure 2, one for each axis. As our results were already quite good with one single network, and we did not have much more time left for running experiments, we did not further experiment with the backup configuration.

4. EXPERIMENTS AND RESULTS

By feeding the preprocessed and augmented $52 \times 52$ patches into the network structure described in figure 2, we were able to achieve very good results on the validation folds with respect to the AUC scores. These results can be viewed in table 2.

Figure 2 shows a plot of the free-response receiver operating characteristic (FROC). It can be seen that our results are quite decent, although not as good as potentially indicated by...
the very high AUC scores, on which we will further elaborate on in the following discussion section.

![Fig. 2: The FROC-curve](image)

<table>
<thead>
<tr>
<th>Subset</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.9944</td>
</tr>
<tr>
<td>1</td>
<td>0.9976</td>
</tr>
<tr>
<td>2</td>
<td>0.9981</td>
</tr>
<tr>
<td>3</td>
<td>0.9907</td>
</tr>
<tr>
<td>4</td>
<td>0.9936</td>
</tr>
<tr>
<td>5</td>
<td>0.9918</td>
</tr>
<tr>
<td>6</td>
<td>0.9958</td>
</tr>
<tr>
<td>7</td>
<td>0.9907</td>
</tr>
<tr>
<td>8</td>
<td>0.9929</td>
</tr>
<tr>
<td>9</td>
<td>0.9985</td>
</tr>
</tbody>
</table>

Table 2: AUC results for cross validation.

Part of the data augmentation we applied was flipping the images horizontally and vertically. This augmentation stems from the realization that the nodules should be fairly point-symmetrical. One could further exploit this symmetry by not only flipping the images, but applying random rotations as well. To avoid the need to apply padding to fill the periphery of rotated patches, creating artifacts that would only be present in the training data, this additional augmentation would require the extraction of larger initial patches.

Furthermore, the deficits in our FROC-curve might be partially explained by two factors: first of all our candidates did not contain all true positives to begin with, secondly we might have missed a few true positives as we only extracted patches for those candidates around which we were able to extract 60×60 patches. Although it seems highly unlikely that candidates close to the image boundary might be true positives it might be safer to also include those. If we had more time we would think about ways to include those lost true positives in order to improve our FROC score.

Another important issue to consider is the way the three different 2D-planes are being distinguished. Feeding patches from all directions into the same network allows the training of the network on a larger number of samples, which is important considering the low number of true positives and the limits of data augmentation. There are, however, conceptual advantages to training three different networks, one for each of the directions, and to creating an ensemble of those networks: This ensemble might be better at learning features which are direction-specific, since not all directions are fed to the same network. It is hard to say out of hand which of these approaches works best. Unfortunately, we did not have the time to also test our setup using one network for each of the directions that we had already implemented. However, we believe that this might have improved our FROC score in the end.

Also impacting our performance were timing constraints that are due to the scope of this project. We were therefore not able to experiment a lot with the hyper-parameters, such as learning rate, momentum and learning rate decay. We were also not able to test and contrast different network architectures, thus having to rely on intuition to create a network that reaches a reasonable performance on this problem. We spent a lot of our time on designing and implementing a framework that allows us to actually train on all 9 subsets at once during each iteration of the cross-validation without getting memory problems and also balances the data by augmentation on the fly, of which we are quite proud.

Due to this limited amount of experimentation, we are reasonably confident that our results could be further improved by systematic parameter fine-tuning. Nevertheless, we are quite happy with the results that we achieved. Also, we believe that given our high AUC scores, we actually managed to build and train a convolutional neural network which can be used for accurate nodule classification.

6. REFERENCES

