FALSE POSITIVE REDUCTION FOR NODULE DETECTION IN CT SCANS USING A CONVOLUTIONAL NEURAL NETWORK: APPLICATION TO THE LUNA16 CHALLENGE

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 pulmonary nodules, extracted from thoracic computed tomography scans, using a convolutional neural network operating on patches extracted from each of the three orthogonal planes. 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 at the patch level was 0.9944 on average per fold in 10-fold cross-validation. The system achieved a competition score of 0.784, detecting 81.5% of all nodules at an average of 1 false positive detections per scan.

1. INTRODUCTION

This report covers our work on the LUNA16 competition (`luna16.grand-challenge.org`), which we entered as part of the course "Computer Aided Diagnosis in Medical Imaging", edition 2016, Radboud University Nijmegen. The LUNA16 challenge asks the contestants to build a computer aided diagnosis (CAD) system for early detection of pulmonary nodules from computed tomography (CT) scans. It is important to detect these nodules because a fraction of them represent lung cancer. We participated in the false positive reduction track of the challenge where teams analyze a set of nodule candidates provided by the challange organizers. In earlier phases of the course we developed a lung segmentation algorithm and a candidate detector, but we did not use these in this work.

To identify the true-positive nodules among the candidates, 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 for the LUNA16 challenge 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. 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 (in z, x or y direction). The resulting patches measure 6×6 centimeters, centered around the candidate location, and are handled and saved as three-dimensional ($3 \times 60 \times 60$) numpy arrays – bearing some structural resemblance to 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 $3 \times 52 \times 52$ (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 is repeated several times. 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 IMAGEDATAGENERATOR() of Keras [Chollet, 2015], a wrapper for Theano [Theano Development Team, 2016] and Tensor Flow [Abadi et al., 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 the same patch to the file system just to load them back into memory for processing, efficiently and effectively speeding up the whole training procedure. This method has the additional benefit of yielding 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, as only during this phase there is a need for balancing the two classes (and as we didn't want our testing method to deviate from the competition standard).

3.3. Patch Sizes

We found that the size of the cropped-out patches was crucial for our network to be able to distinguish true nodules from false positives. When the extracted patches were smaller (40×40 instead of 60×60 with 52×52 crops), it tended to classify all test patches 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. The same procedure was applied during testing.

3.5. CNN Configuration

The network has 3 convolutional layers, with max-pooling layers of size 2×2. Batch normalization was applied after each max-pooling layer to reduce overfitting. 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

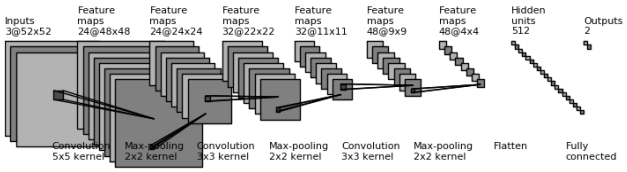

Fig. 1: Architecture of the convolutional neural network.

Table 1: Convolutional layer architecture.

	# of kernels	size ($x \times y \times z$)
C1	24	$5 \times 5 \times 1$
C2	32	$3 \times 3 \times 1$
C3	48	$3 \times 3 \times 1$

512 neurons. We used 2 dense layers in total, where the last layer is connected to a softmax layer for classification. The weights were initialized using He uniform initialization [He et al., 2015]. For the update rule we used Nesterov accelerated gradient descent with a learning rate of 0.01, a decay of $1 \cdot 10^{-3}$ and a momentum of 0.9. Our 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 into 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 testing set. All training procedures ran for 80 epochs, with a random sample of 20,000 negative patches and a mini-batch size of 128. 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 per plane. Subsequently, the average over the three planes for each candidate nodule is computed.

4. EXPERIMENTS AND RESULTS

By feeding the preprocessed and augmented 52×52 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).

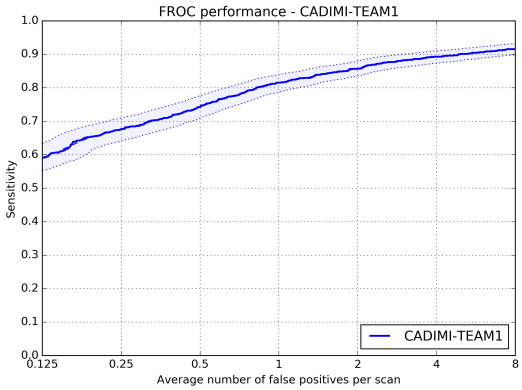

Fig. 2: The FROC-curve

Subset	AUC
0	0.9944
1	0.9976
2	0.9981
3	0.9907
4	0.9936
5	0.9918
6	0.9958
7	0.9907
8	0.9929
9	0.9985

Table 2: AUC results for cross validation.

5. DISCUSSION

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. We did not extract patches for candidates that were close to the image boundary. In hindsight we should have included those candidates in order to decrease the chance of missing pleural nodules. 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.

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. Due to this limited amount of experimentation, we are reasonably confident that our results could be further improved by systematic parameter fine-tuning.

Our resulting framework is capable of performing proper 9 out of 10 cross-validation without facing memory errors. Furthermore, it allows balancing of the classes by data augmentation on the fly. In our opinion, the resulting high AUC scores reflect our CNNs capability of accurately classifying nodules.

6. REFERENCES

[Abadi et al., 2016] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

[Chollet, 2015] Chollet, F. (2015). Keras. `https://github.com/fchollet/keras`.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

[Setio et al., 2016] Setio, A. A. A., Ciompi, F., Litjens, G., Gerke, P., Jacobs, C., van Riel, S. J., Wille, M. M. W., Naqibullah, M., Sánchez, C. I., and van Ginneken, B. (2016). Pulmonary nodule detection in ct images: false positive reduction using multi-view convolutional networks. *IEEE transactions on medical imaging*, 35(5):1160–1169.

[Theano Development Team, 2016] Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.