# Solution of the 'grt123' Team

May 5, 2017

## 1 Introduction

Our model is a unified framework of nodule detection and cancer classification based on 3D convolution neural networks. The model receives preprocessed 3D lung scans as input, and outputs both the bounding boxes of suspicious nodules and the probability of getting cancer, the latter of which are used to calculate the loss for submission.

The model provides explainable results in that it explicitly learns the procedure of suspicious nodule detection and full lung information integration. We believe that further investigations into the network behaviors may provide helpful insights to understand the pathological mechanism of cancer detection.

The solution can be divided into three modules, preprocessing, nodule detection and cancer classification. In the following content, we describe them in details.

## 2 Preprocessing

All raw data are first converted into Hounsfield units.

### Extract mask

Next we calculate the mask of space inside a lung. For each slice, the 2D image is filtered with Gaussian filter (size = 1 pixel) and then binarized using -600 as threshold. All 2D connected components that are smaller than 30mm$^2$ or have eccentricity greater than 0.99 are removed. We then find all 3D connected components in the resulting binary 3D matrix, and keep only the ones that does not touch matrix corner, and have a volume between 0.68L and 7.5L.

The remaining components are further analyzed. For each slice of a component, we calculate the area (Area) and the distance (MinDist) to the slice center. If the average MinDist of slices whose Area is greater than 6000mm$^2$ is greater than 62mm, this component is removed. The remaining components are then unioned, representing the lung mask based on image intensity.

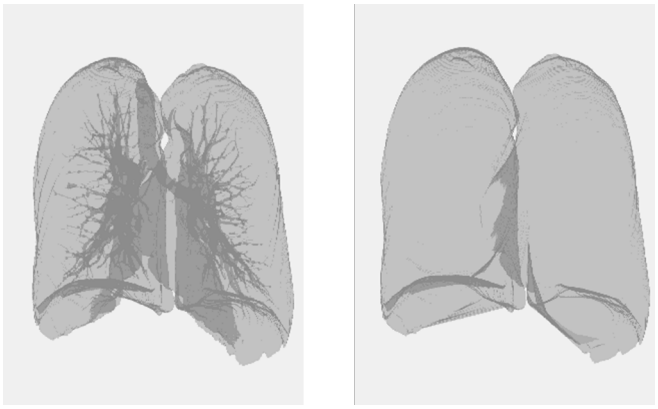

Figure 1: Lung mask calculated in preprocessing. The left panel is raw mask based on intensity. The right panel is the closed and dilated mask to multiply on raw data.

Lung in some cases is connected to the outer space, a few slices from the top need to be removed first to make preprocessing work.

### Convex hull & Dilation

We erode the mask until the biggest two components have similar volume. They are dilated to original size and intersect with raw mask to generate masks for two lungs separately. For each slice of one lung mask, the 2D binary image is replaced with its convex hull if the increased area does not exceed 50%. Masks are further dilated for 10 voxels to include some surrounding space. A full mask is obtained by unioning masks for two lungs.

### Intensity normalization

To prepare the data for deep networks, we transformed data in Hounsfield unit to integers. The raw data matrix is first clipped with a window [-1200, 600], and then transform linearly to [0, 255]. It is then multiplied with the full mask obtained above, and everything outside the mask is padded with 170. In addition, for the space generated by dilation in last step, all values greater than 210 (typically the bones) is replaced with 170 too. The padding value 170 is chosen to resemble the intensity of tissue.

## 3 Dataset

Two lung scans datasets are used to train the model, LUNA16 dataset (abbreviated as LUNA) and the Kaggle stage 1 training set (abbreviated as Kaggle1). LUNA includes the nodule labels annotated by radiologists, while Kaggle1 only includes the per-subject binary cancerous labels.

There are two possible problems with the original datasets. First, LUNA has some very small annotated nodules, which may be less irrelevant to the cancer. Second, some significant differences exist between LUNA nodules and Kaggle1 nodules. There are two kinds of special nodules which exist in Kaggle1 but are rarely found in LUNA, including the very big nodules (larger than 60 mm) and the nodules on main bronchus. See Figure 2 for an example. If the network is trained on LUNA only, it will be difficult to learn to detect these large nodules. To cope with these two problems, we remove the nodules smaller than 6 mm from LUNA annotations and manually labeled the nodules in Kaggle1.

It is worth noting that the authors have no professional knowledge on lung cancer diagnosis, so the nodule selection and manual annotations may raise considerable noises. Nevertheless, these modifications are mainly based only on the appearance of nodules (size) and are demonstrated to be useful in later experiments. More importantly, the model in the next stage (cancer classification) is designed to be robust to wrong detections, which alleviates the demand on highly reliable labels.

ea7373271a2441b5864df2053c0f5c3e

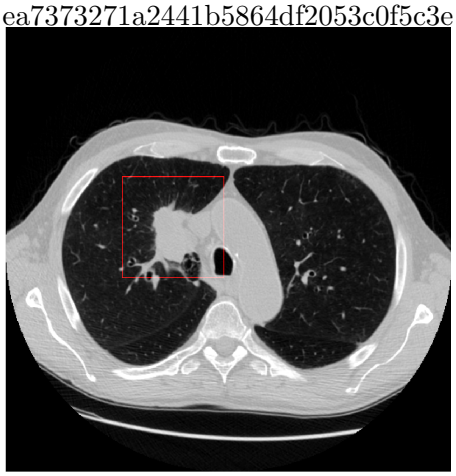

Figure 2: An example of big nodule on main bronchus.

## 4 Nodule detection

We designed a 3D convolutional neural network to detect suspicious nodules. Basically it is a 3D version proposal net but equipped with some task-specific modifications to the training procedure and network structure. Unlike the generic object detection in computer vision which involves with tens or hundreds of classes, there are only two classes in this task. So the predicted proposals are directly used as detection results, and a second-stage classifier used in RCNN-stype detectors is not adopted.

## Patch-based input for training

Object detection models usually adopt the image-based training. In the training phase, the entire image is used as input to the network. However, this is not feasible for our 3D ConvNet model due to the GPU memory constraint. When the resolution of lung scans is kept at a fine level, even a single sample consumes more than the maximum memory of mainstream GPUs.

To overcome this problem, patch-based inputs are used for training. The input is a $128 \times 128 \times 128$ cube cropped from the lung scans. Two kinds of patches are randomly selected. First, 70% of the inputs are selected so that they contain at least one nodule target. Second, 30% of the inputs are cropped randomly from lung scans, and may contain no nodules. The later kind of inputs ensure a coverage of enough negative samples.

If a patch goes beyond the range of lung scans, it is padded with value 170, same as in preprocessing. The nodule targets are not necessarily located at the center of patch, but had a margin larger than 12 pixels from the boundary of the patch (except for a few nodules that are too big).

Data augmentation is used to alleviate the over-fitting problem. The patches are randomly left-right flipped and resized with a ratio bewteen 0.8 and 1.15. Other augmentations are also tried, such as axes swapping and rotation. However, they yield no significant improvement and thus are not adopted.

## Positive sample balancing

Even though we have removed some very small nodules from LUNA, the distribution of nodule size is still highly unbalanced. The number of small nodules is much greater than that of big nodules. If uniform sampling is used, the network will learn to be biased to small nodules. while sacrificing the accuracy of big nodules. This is unwanted because big nodules are usually stronger indicators of cancer than smaller ones. To prevent this result, we increase the sampling frequencies of big nodules in the training set. Specifically, the sampling frequency of nodules larger than 30, 40 mm were 2,6 times larger than other nodules, respectively.

## Hard negative mining

Negative samples are much more than positive samples, and their distribution is highly unbalanced. Most of of them are easy to be discriminated by the network, but the others have a similar appearance with suspicious nodules. As a common practice in object detection, hard negative mining is usually used to deal with this problem. We use an simple online version of hard negative mining in the training process.

First, we process the patches using the network, and obtain the output map, on each pixel of which are the classification confidence scores and regression terms. Second, $N$ negative samples are randomly selected as a candidate pool. Third, the negative samples in this pool are sorted in descending order based on their classification confidence scores, and the top $n$ samples are selected as the hard negatives. Other negative samples are

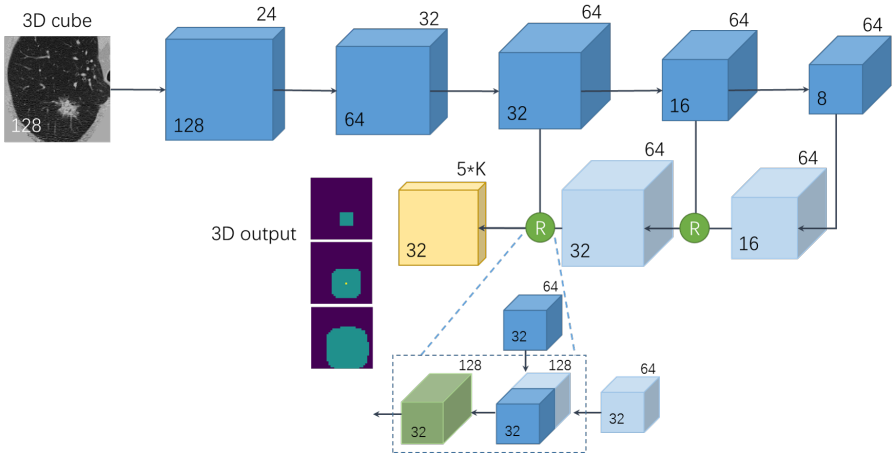


Figure 3: Structure of nodule dection net.

discarded and not included in the computation of loss. The use of a randomly selected candidate pool can reduce the correlation between negative samples. By adjusting the size of the candidate pool and the value of $n$, the strength of hard negative mining can be controlled.

## Network structure

The detector network is inspired by U-Net, and its structure is shown in Figure 4. It is composed of five groups of 3d residual blocks interleaved with four pooling layers, and a set of lateral and feedback connections. The features of the final feedback layer is used to compute the output, whose size is $32 \times 32 \times 32$. Two $1 \times 1 \times 1$ convolutions are applied to obtain the classification score and bounding box regression terms.

The network has three anchors of different scales, corresponding to 10, 30, and 60 mm, respectively. Intersection over Union (IoU) are used to determine the class of pixels on on the final feedback layer. Pixels whose IoU with the target nodule larger than 0.5 and smaller than 0.02 are treated as positive and negative samples, respectively. Other pixels are neglected in the training process. Binary entropy loss is used for classification. The regression terms contain bounding box regression labels $(\Delta x/x, \Delta y/y, \Delta z/z, \log(\Delta r))$, and smooth L1 Loss is used. So the shape of final label map is $32 \times 32 \times 32 \times 5 \times 3$.

## Fully convolutional output during the test phase

After the network is trained, the entire lung scans are used as input to obtain all the suspicious nodule detections. Because the network is fully convolutional, it is straightforward do this. However there is still some problem with the GPU memory constraint. Even though the network needs much fewer memories in the testing phase, it still exceeds the maximum memory of our GPU. To overcome this problem, we split the lung scans into several parts, process them separately, and then combine the results. Note that we

keep these splits overlapped by a large margin to eliminate the unwanted border effects during convolution computations.

# 5 Cancer classification

## Network structure

After the nodule detection net (N-Net for short) is trained, we applied it onto the whole 3D data of each patient. The top 5 proposals are picked out based on their confidence score in N-Net. During training stage, proposals are picked stochastically in a softmax way, while in testing stage no softmax is used.

We extracted the last convolution layer of N-Net for each proposal, which is a $32 \times 32 \times 32$ cube of 128 features shown in Fig. 3 as a green volume. We max pool over the central $2 \times 2 \times 2$ voxels of each proposal, and pass the resulting 128-D features to two succeeding fully connected layers to generate a prediction of cancer probability. We assume the final diagnosis is made from these 5 proposals and a hypothetical dummy nodule (in case N-Net missed some nodules) independently. Prediction loss is defined on the final prediction $P = 1 - (1 - P_d)\prod_i (1 - P_i)$, in which $P_d$ is a free parameter representing the cancer probability from the dummy nodule (Fig. 4).

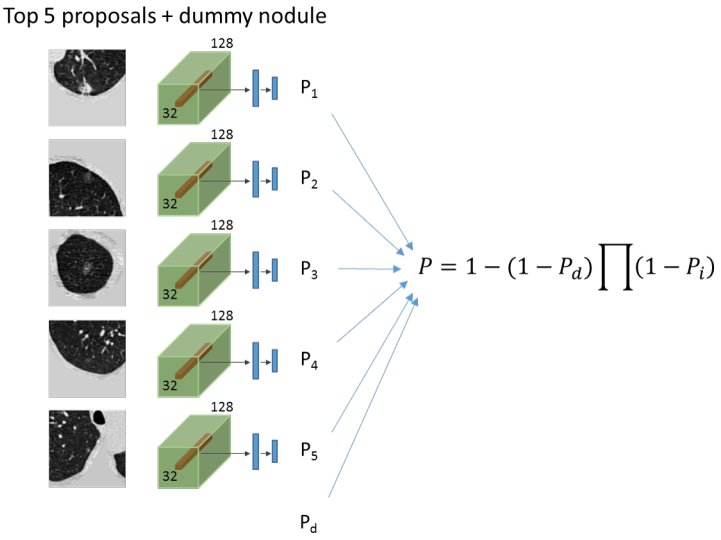


Figure 4: Structure of cancer classification net.

An additional loss is added to facilitate training. If $P_i$ is smaller than 0.03 and the proposal is labeled as nodule in N-Net training, loss for classification net is added with $-\ln(P_i)$.

### Interleaving training

To keep N-Net making sense all the time, we also trained it during training cancer classification net (C-Net for short). In each epoch, network parameters are updated by training C-Net and N-Net one step respectively. Convolution layers are shared between C-Net and N-Net.

Only flip augmentation is used in classification task in the first stage. Then it was fine-tuned with other data augmentation types (flip, resize, swap, rotation).

## 6 Implementation

We adopted stochastic gradient descent (SGD) and the common used step-wise learning rate schedule to train the network. An initial learning rate is set, and when the loss stops decreasing, the learning rate is reduced by 1/10. This process is repeated until the network converges. The batch size is set according to the size of models and GPU memory.

The model is implemented using Pytorch, which is a very flexible and efficient deep learning framework. The model is run over 8 TITAN X GPUs.