End-to-End lobes segmentation using vnet with coordconv layers

Junxuan Chen, Wenjia Wang

This is a brief introduction of our method.

Give the overall structure of the algorithm. Does your algorithm search for boundaries or fissures? Does it use airway or other in- formation? Does it use an atlas and/or region growing?

We used a fully automatic approach to the problem of the segmentation of the five pulmonary lobes from a chest CT scan using end-to-end Vnet with coordconv layers. To improve the precision, we also used the lung segmentation result . The lung lobe borders and fissure maps used in the auxiliary task of the network were automatically obtained from manual lobe segmentations. The lobes were extracted after subtracting a morphological erosion with a unitary square kernel to the original lobe map. The fissure dice loss was the additional loss output to improve the model representation and the ability to separate the lobes. Airway or atlas or other information didn't been used.

Briefly describe each step in the structure of the algorithm (If applicable, which type of algorithms were used for preprocessing? How are different types of information combined?).

We proposed an end-to-end method to segment lobes from a chest CT scan. The basic architecture was 3D Fully Convolutional Neural Network dense Vnet, which was widely used in biomedical image because of its capability to solve segmentation problems relying on small sets of training data. The architecture was constituted by an encoding path followed by a decoding path. The encoding part followed a typical CNN architecture where convolutional layers iteratively decreased the feature resolution while the number of channels was increased in the same order. Batch normalization was employed to avoid overfitting.

To avoid wrong separation outside lung area, we used a 2D automatic lung segmentation preprocessing method before segment lung lobes. The outside area was background. The lung area must belong to five lobes classes.

The volume regions surrounded by elements of other classes were more difficult to segment so we used the additional fissure dice loss to focus more attention on the fissure borders.

By experiments we found there were wrong lobes classes between left and right lungs areas. The Convolutional neural network just used local information rather than global information. There

were so many similarities in grey values and shapes and other things in left lung lobes and right lung lobes. To solve this problem we used coordconv layers(see figure 1) before the last convolutional neural network to add coordinate information.

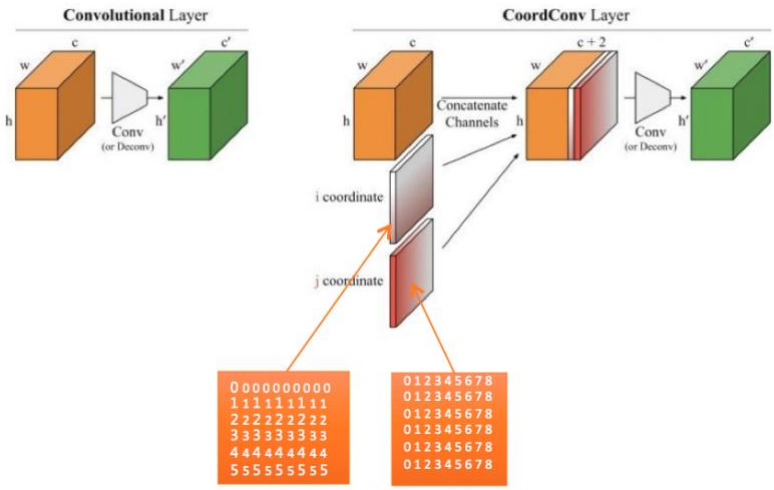

Figure 1 coordconv layers

List limitations of the algorithm. Is the algorithm specifically designed to segment only certain types of scans? Is your algorithm intended for segmenting pathological lungs? Was it optimized to work for scans with thick or thin slices, are other technical scan parameters expected to influence segmentation performance?

There were few pathological lungs in our train and test cases. In our cases the model had good robustness. However， there were only thick slices in our cases. We were not sure the inference result in serious pathological lungs.

Was the algorithm trained with example data? If so, describe the characteristics of the training data.

we selected 328 CT volumes for annotation totally, which contained 71 cases from the LUNA16 dataset, 195 cases from LIDC dataset, 62 cases from Meinian Onehealth Healthcare Holdings. The CT scans used in the experiment had a variable number of slices with each CT volume consisting between 200 to 528 slices of size 512 × 512 pixels. We used SlicerCIP to make ground truth masks.

If the algorithm has been tested on other databases, you could consider including those results.

We annotated 15 lola11 cases and tested on them. Our global lobes dice was 0.948

What is the avarage runtime of your algorithm, and on which system is this runtime achieved?

The average runtime of our algorithm was 12s per case using 1 NVIDIA TESLA P100.

Is your algorithm automatic or semi-automatic? If user input is used, how much is needed and in what way?

Our algorithm was fully automatic. No user input was used.