Automatic Lung Vessel Segmentation via Stacked Multiscale Feature Learning

Ryan Kiros, Karteek Popuri, Matthew Low, Dana Cobzas, Martin Jagersand
Department of Computing Science, University of Alberta

We introduce a representation learning approach to segmenting vessels in the lungs. Our algorithm takes as input a CT volume and outputs a learned feature vector for each voxel. These feature vectors are then passed into a logistic regression classifier for making a probabilistic prediction on the occurrence of a vessel. Our features are learned using both multiple layers and scales which capture both local and global image characteristics into their representation. We note that unlike the hand-crafted features used by other competitors, our approach makes no assumption of the task at hand or even the modality being used.

1 Method

We assume we are given m volumes with s channels $\{\{V^{(j)}\}_{j=1}^{s}\}_{i=1}^{m}$ with each $V_i^{(j)} \in \mathbb{R}^{n_V \times n_H \times n}$ where $n_V \times n_H$ is the spatial dimension of a slice and n is the number of slices. For the competition data, there is only a single input channel. For simplicity, we assume each volume $V_i^{(j)}$ has dimensionality $n_V \times n_H \times n$ although this need not be the case. The general outline of our feature learning framework is as follows:

- Extract patches at multiple scales using a Gaussian pyramid.
- Learn a filter bank using spherical k-means.
- Convolutionally extract feature maps using the learned filters as kernels.
- Repeat the above steps, using the computed features maps as input to another layer. The number of feature maps (next layer channels) corresponds to the number of filters.

In each of the following subsections, we describe the above operations in detail.

1.1 Pre-processing and dictionary learning

Given a volume V, a Gaussian pyramid with Γ scales is applied to each channel of each slice. Let $\{p^{(1)}, \ldots, p^{(m_P)}\}$ denote a set of m_P patches randomly extracted from the scaled volumes. Each patch $p^{(l)}$ is of spatial dimension $r \times c \times s$ where $r \times c$ is the receptive field size. These patches are then flattened into column vectors. Per patch contrast normalization and patch-wise mean subtraction is performed. For dictionary learning we use orthogonal matching pursuit, described by the following optimization problem:

$$\begin{aligned} \underset{D, x^{(i)}}{\text{minimize}} \quad & \sum_{i=1}^{m_P} ||Dx^{(i)} - p^{(i)}||_2^2 \\ \text{subject to} \quad & ||D^{(j)}||_2^2 = 1, \forall j \\ & ||x^{(i)}||_0 \leq q, \forall i \end{aligned} \tag{1}$$

where $D \in \mathbb{R}^{n_P \times k}$ and $D^{(j)}$ is the j-th column of D. Optimization is done using alternation over the dictionary D and codes x. For all of our experiments we set $q = 1$, which reduces to spherical k-means. In

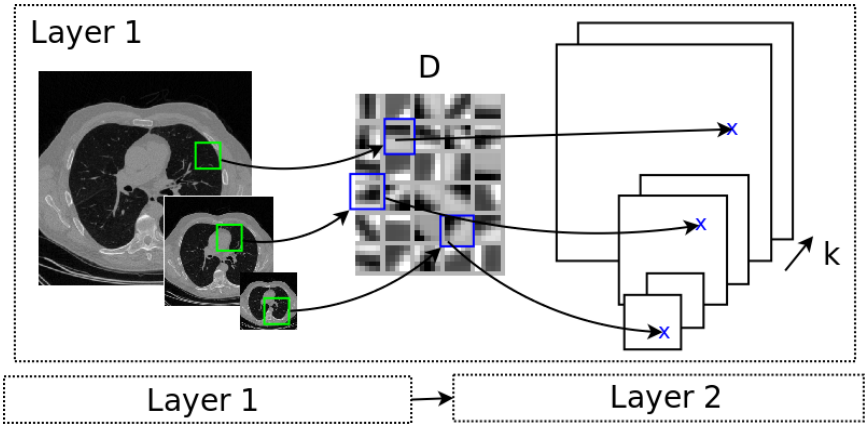

Figure 1: Visualization of our feature learning approach. Each volume slice is scaled using a Gaussian pyramid. Patches are extracted at each scale to learn a dictionary D using spherical k-means. Convolution is performed over all scales with the dictionary filters, resulting in Γk feature maps. After training the first layer, the feature maps can then be used as input to a second layer.

particular, given a dictionary D, an index k is chosen as

$$k = \underset{j}{\text{argmax}} |D^{(j)^T} p^{(i)}| \tag{2}$$

for which the k-th index of $x^{(i)}$ is set as $x_k^{(i)} = D^{(k)^T} p^{(i)}$ with all other indices left as zero in order to satisfy the constraint $\|x^{(i)}\|_0 \leq 1$ for all i. Given the one-hot codes X, the dictionary is easily updated by first solving the unconstrained problem, followed by re-normalization as to satisfy the constraint $\|D^{(j)}\|_2^2 = 1$ for all j. Additional details on learning features with k-means can be found by [1].

1.2 Convolutional feature extraction

Let T_j^γ denote a volume slice of channel j and scale γ. Each $r \times c \times s$ patch in T_j^γ is pre-processed by contrast normalization and mean subtraction. Let $D_j^{(l)} \in \mathbb{R}^{r \times c}$ denote the l-th basis for channel j of D. We will define the feature encoding for basis l to be given by:

$$f_l^\gamma = \sum_{j=1}^{s} T_j^\gamma * D_j^{(l)} \tag{3}$$

where * denotes convolution. The resulting feature maps $\{f_l^\gamma\}_{l=1}^k$ are of the same spatial dimensions as T_j^γ. The feature maps are finally upsampled to the original $n_V \times n_H$ spatial dimension. Figure 1 illustrates our approach.

1.3 Stacking multiple layers

Our described setup for feature learning has involved scaling, dictionary learning and convolutional extraction. Just as the volumes slices were inputs to a first layer with s channels, the upsampled output feature maps $\{\{f_l^\gamma\}_{\gamma=1}^\Gamma\}_{l=1}^k$ may be seen as inputs to a second layer but with Γk channels. All the same described operations are applied a second time resulting in additional second layer output feature maps. These groups of feature maps can be concatenated together resulting in a total number of $\Gamma_1 k_1 + \Gamma_2 k_2$ feature maps, where Γ_1, k_1 are the number of first layer scales and filters while Γ_2, k_2 are the number of second layer scales and filters. Thus each pixel in a volume slice can be represented as a $\Gamma_1 k_1 + \Gamma_2 k_2$ dimensional feature vector.

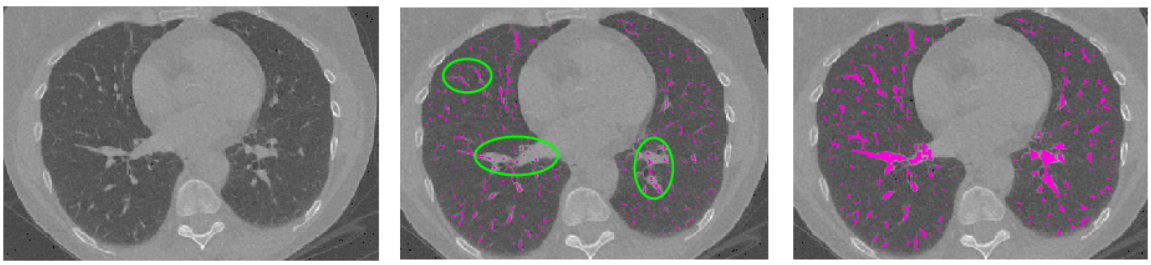

Figure 2: Visualizing the importance of scale and depth for vessel segmentation.

1.4 Segmentation

We utilize the 3 labeled volumes for training a L2-regularized logistic regression classifier using the learned voxel features as inputs. For hyperparameter selection, 10-fold cross validation is used. To make predictions on an unseen volume, voxel features are extracted using the learned dictionaries, which was also learned just on the 3 training slices. The classifier is used to make a prediction to each voxel independently. For the purpose of the competition, the predicted probabilities are linearly scaled to [0,255] which are used for submission.

2 Results

The experimental setup of our algorithm is as follows. Dictionaries are learned using 6×6 first layer patches and 2×2 second layer patches. We use 32 first layer features, 32 second layer features and 6 scales on both layers. This results in $32 \times 6 + 32 \times 6 = 384$ features for each voxel. To determine the importance of using multiple layers and scales, we visually analyzed the algorithm's performance on a subset of unseen slices. Figure 2 illustrates segmentation results on a single slice. The middle slice illustrates segmentation using a single layer and only one scale, while the rightmost slice illustrates segmentation with 6 scales and 2 layers. Overall, our result obtains first place over 20 other submissions based on results computed via AUC.

3 Additional Details

Our algorithm was implemented using Matlab and ran on a Intel i7 3.6GHz machine with 32GB RAM . Dictionary learning on both layers took under 5 minutes. Extracting features and classifying each volume took over an hour. The bottleneck of our approach was in the convolution of the filters with each slice. We expect that a highly optimizated convolution routine and/or extracting features on the GPU would greatly speed up the algorithm's run-time performance. Memory requirements are low since only features learned from a single slice of a volume need to be computed at once for predictions.

References

[1] Adam Coates and Andrew Y Ng. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade*, pages 561–580. Springer, 2012.