**Fissure Net: A Deep Learning Approach For Pulmonary Fissure Detection in CT Images**

# ABSTRACT:

Pulmonary fissure detection in computed tomography (CT) is a critical component for automatic lobar segmentation. The majority of fissure detection methods use feature descriptors that are hand-crafted, low-level, and have local spatial extent. The design of such feature detectors is typically targeted towards normal fissure anatomy, yielding low sensitivity to weak and abnormal fissures that are common in clinical datasets. Furthermore, local features commonly suffer from low specificity, as the complex textures in the lung can be indistinguishable from the fissure when global context is not considered. We propose a supervised discriminative learning framework for simultaneous feature extraction and classification. The proposed framework, called FissureNet, is a coarse-to-fine cascade of two convolutional neural networks. The coarse-to-fine strategy alleviates the challenges associated with training a network to segment a thin structure that represents a small fraction of the image voxels.

FissureNet was evaluated on a cohort of 3706 subjects with inspiration and expiration 3DCT scans from the COPDGene clinical trial and a cohort of 20 subjects with 4DCT scans from a lung cancer clinical trial. On both datasets, FissureNet showed superior performance compared to a deep learning approach using the U-Net architecture and a Hessian-based fissure detection method in terms of area under the precisionrecall curve (PR-AUC). The overall PR-AUC for FissureNet, UNet, and Hessian on the COPDGene (lung cancer) dataset was 0.980 (0.966), 0.963 (0.937), and 0.158 (0.182), respectively. On a subset of 30 COPDGene scans, FissureNet was compared to a recently proposed advanced fissure detection method called derivative of sticks (DoS) and showed superior performance with a PR-AUC of 0.991 compared to 0.668 for DoS.

# INTRODUCTION:

Computed Tomography (CT) measures X-ray projections of the body at different angles to reconstruct a volumetric image of the anatomy. The contrast produced in a CT image reflects differences in X-ray photon attenuation, which in the lungs broadly reflects tissue density. Technological advancements in CT hardware have made it possible to scan the entire thoracic cavity in less than one second and reconstruct images with submillimeter spatial resolution.

These properties make CT imaging the standard modality for imaging the intricate structures of the lung. Pulmonary CT is routinely used for diagnostics, treatment planning and delivery, and post-intervention evaluation.

CT images provide a rich source of information regarding the extent and spatial distribution of pulmonary disease. Computer-aided systems are essential for objective quantification and characterization of the complex information present in the image. Algorithms have been developed for detection and classification of nodules [1], texture classification of obstructive disease [2], pulmonary embolism detection [3], and quantitative airway analysis [4]. Although CT is an anatomical imaging modality, functional information about the lung may be derived from CT scans collected at different inspiration levels using image registration [5].

The human lungs are composed of five lobar compartments, which are separated anatomically by three lobar fissures. The left oblique (major) fissure (LOF) separates the lower and upper lobes of the left lung. The right oblique (major) fissure (ROF) separates the lower lobe from the middle and upper lobes, and the right horizontal (minor) fissure (RHF) separates the middle and upper lobes of the right lung. It is often of clinical interest to perform quantitative analysis within each lobe individually. Boueiz et al. recently identified subgroups of upper-lobe-predominant emphysema and lower-lobe-predominant emphysema and found associations with clinical and imaging outcomes [6]. Accurate knowledge of lobar anatomy is critical for successfully treating severe emphysema with bronchoscopic lung volume reduction [7]. Lobar information also serves as a precursor to other image analysis algorithms including image registration. Currents- and varifolds-based registration algorithms rely on accurate surface representations of the lungs, lobes, and vessel trees [8].

The lobes are generally anatomically independent, but incomplete fissures are possible and the detection of incompleteness may be clinically relevant. An individual’s unique lobar structure is likely to influence lung tissue mechanics and patterns of regional ventilation. Fissure incompleteness and the resulting collateral ventilation reduces the efficacy of endobronchial valves [9–11]. Gopelmann et al. recently showed that apical vs. basal emphysema distribution varies with fissure integrity [12]. However, Pu et al. found no relationship between fissure integrity and COPD severity [13].

Natural variability in lobar anatomy has impeded the development of robust CT analysis methods for fissure and lobar segmentation. In cross-sectional CT images the fissures appear as thin surface-like structures (less than 1 mm thick) withhigher image intensity than the surrounding lung parenchyma. This makes it difficult to identify fissures in low-dose or thickslice CT scans. Fissure segmentation in pathological lungs is further complicated by diseases that locally resemble fissures, for example, bullous lung disease and fibrosis may locally resemble fissures.

Despite these challenges, many attempts have been made to design automatic methods for lobar segmentation [14–20]. The majority of these methods consist of four common modules: lung segmentation, fissure detection using local appearance information, removal of falsely identified fissures, and surface fitting to interpolate and/or extrapolate incomplete fissures. Doel et al. presented an extensive review on pulmonary lobe segmentation and proposed that these individual components should be independently developed and evaluated, opposed to comparing entire pipelines [21]. We follow this proposal and focus on the fissure detection in this work.

Several methods have been proposed for the detection of fissures in CT images. Eigenanalysis of the Hessian matrix is commonly used to exploit the characteristic property that plane-like structures have one direction with large curvature in the intensity profile and two orthogonal directions with vanishing intensity curvature [19, 22–25]. Zhang et al. used a ridgeness operator based on 2D multi-local level set extrinsic curvature measure with structure tensor analysis (MLSECST) [14, 26]. Other works use knowledge of fissure appearance on 2D cross sections to design a filter bank of 2D line filters to detect fissure structures [27–29]. Traditional machine learning approaches use domain-specific hand-crafted features and labeled training data to train a classifier. van Rikxoort et al. used a feature set including intensity, Gaussian derivatives, gradient, and Hessian eigenvalues with labeled training data to build a kNN classifier [30]. The authors showed superior performance compared to conventional unsupervised fissure detection. Wei et al. trained an artificial neural network using texture-derived image features. However, a limitation of this method is that it requires extensive post-processing and only works on major fissures [31].

These existing fissure detection methods are limited to local descriptors of fissure shape and appearance. Although local information is necessary for the precise localization of the fissure, we argue that it is not sufficient. Weak and incomplete fissures diminish local response, and pulmonary disease can locally resemble fissures. We hypothesize that knowledge of global and contextual information can improve specificity by providing guidance when the fissure signal is low or noisy. However, it is far more challenging to design abstract features, such as those that capture global context, compared to low-level features, such as edges. Additionally, hand-crafting features requires domain expertise, and generalizing such a framework to other tasks is not trivial. Alternatively, convolutional neural networks (ConvNets or CNNs) are capable of learning abstract features directly from training data.

Several ConvNet architectures have been proposed for semantic segmentation; the majority are symmetrical networks consisting of an encoder and corresponding decoder [32–34]. Compared to a classification ConvNet which yields a single prediction for each class, a segmentation ConvNet produces aprediction map that has the same spatial resolution as the input. U-Net and SegNet are notable encoder-decoder networks, each of which incorporate skip connections between corresponding encoder and decoder elements to preserve precise localization information that would otherwise be lost with pooling operations [32, 33]. These symmetrical networks are memory intensive and cannot be trained on entire volumetric medical images due to current GPU memory limitations. The majority of ConvNet methods use either 2D slices or small image crops to accommodate memory limitations thereby compromising the capacity of the network to learn large-scale 3D features or global patterns.

For the task of fissure segmentation, both 3D structure and global context are critical for accurate segmentation. Therefore 2D slices or patchwise approaches are not ideal. Furthermore, directly training a network to segment fissures is challenging due to the large class imbalance between fissure and nonfissure voxels. High accuracy could be achieved by learning the trivial classifier that always predicts the majority class (i.e. non-fissure).

To address these challenges, we propose a new coarse-tofine deep learning segmentation approach called FissureNet. FissureNet achieves superior segmentation performance compared to other methods by concatenating two Seg3DNet ConvNets. The new Seg3DNet1 architecture is less memoryintensive compared to U-Net and SegNet, enabling it to learn global contextual information from entire lung images. Seg3DNet is a generic 3D segmentation network suitable for many applications. Within FissureNet, the first Seg3DNet is trained to detect an approximate fissure region of interest (ROI) and the second Seg3DNet is trained to detect precise fissure location within the ROI. The coarse-to-fine approach used by FissureNet overcomes the challenges associated with training a network to segment a thin structure that represents a very small fraction of the total voxel count.

**LITERATURE REVIEW**

**1. TOPIC**: Automatic 3D pulmonary nodule detection in CT images: A survey.

**AUTHOR’S**: I. R. S. Valente, P. C. Cortez, E. C. Neto, J. M. Soares, V. H. C. de Albuquerque, and J. M. R. Tavares.

**DESCRIPTION**:

This work presents a systematic review of techniques for the 3D automatic detection of pulmonary nodules in computerized-tomography (CT) images. Its main goals are to analyze the latest technology being used for the development of computational diagnostic tools to assist in the acquisition, storage and, mainly, processing and analysis of the biomedical data. Also, this work identifies the progress made, so far, evaluates the challenges to be overcome and provides an analysis of future prospects. As far as the authors know, this is the first time that a review is devoted exclusively to automated 3D techniques for the detection of pulmonary nodules from lung CT images, which makes this work of noteworthy value. The research covered the published works in the Web of Science, PubMed, Science Direct and IEEEXplore up to December 2014. Each work found that referred to automated 3D segmentation of the lungs was individually analyzed to identify its objective, methodology and results. Based on the analysis of the selected works, several studies were seen to be useful for the construction of medical diagnostic aid tools. However, there are certain aspects that still require attention such as increasing algorithm sensitivity, reducing the number of false positives, improving and optimizing the algorithm detection of different kinds of nodules with different sizes and shapes and, finally, the ability to integrate with the Electronic Medical Record Systems and Picture Archiving and Communication Systems. Based on this analysis, we can say that further research is needed to develop current techniques and that new algorithms are needed to overcome the identified drawbacks.

**2. TOPIC**: Current- and varifoldbased registration of lung vessel and airway trees.

**AUTHOR ‘S:** Y. Pan, G. E. Christensen, O. C. Durumeric, S. E. Gerard, J. M. Reinhardt, and G. D. Hugo.

**DISCRIPTION**: In these approaches, curve-like structures in the lung—for example, the skeletons of vessels and airways segmentation—and surface of the lung are represented by currents or varifolds in the dual space of a Reproducing Kernel Hilbert Space (RKHS). Currents and varifolds representations are discretized and are parameterized via of a collection of momenta. A momenta corresponds to a line segment via the coordinates of the center of the line segment and the tangent direction of the line segment at the center. A momentum corresponds to a mesh via the coordinates of the center of the mesh and the normal direction of the mesh at the center. The magnitude of the tangent vector for the line segment and the normal vector for the mesh are the length of the line segment and the area of the mesh respectively. A varifolds-based registration approach is similar to currents except that two varifolds representations are aligned independent of the tangent(normal) vector orientation. An advantage of varifolds over currents is that the orientation of the tangent vectors can be difficult to determine especially when the vessel and airway trees are not connected. In this thesis, we examine the image registration sensitivity and accuracy of currents- and varifolds-based registration as a function of the number and location of momenta used to represent tree like-structures in the lung and the surface of the lung.

**3. TOPIC**: The fissure: Interlobar collateral ventilation and implications for endoscopic therapy in emphysema.

**AUTHOR’S**: T. D. Koster and D. J. Slebos.

**DESCRIPTION**: In patients with severe emphysema, bronchoscopic lung volume reduction using one-way valves is a promising therapeutic option to improve lung function and quality of life. The goal of this treatment is to achieve a complete lobar atelectasis. In a significant proportion of patients, this atelectasis cannot be achieved due to interlobar collateral ventilation. This collateral ventilation is generated through incomplete lobar fissures. Therefore, only patients with complete fissures and no collateral ventilation can be selected for endobronchial therapy with one-way valves. Incomplete fissures are very common and exhibit a great variation in anatomy. The reported prevalence is 17%–85% for the right major fissure, 19%–74% for the left major fissure, and 20%–90% for the minor fissure. There are several methods of measuring or predicting the presence of collateral ventilation, with computed tomography (CT)-fissure analysis and the Chartis measurement being the most important. CT-fissure analysis is an indirect method to measure the completeness of fissures as a surrogate for collateral ventilation. The Chartis system is an endobronchial method to directly measure the presence of collateral ventilation. Both methods have unique value, and the combination of both can accurately predict the treatment response to the bronchoscopic placement of endobronchial valves. This review provides an in-depth view of lung fissure and collateral ventilation to help understand its importance in selecting the appropriate patients for new emphysema treatments and thus avoid useless treatment in unsuitable patients.

**4. TOPIC**: A derivative of stick filter for pulmonary fissure detection in CT images.

**AUTHOR’S**: C. Xiao, M. Staring, J. Wang, D. P. Shamonin, and B. C. Stoel.

**DESCRIPTION**:Pulmonary fissures are important landmarks for automated recognition of lung anatomy and need to be detected as a pre-processing step. We propose a derivative of stick (DoS) filter for pulmonary fissures detection in thoracic CT scans by considering their thin curvilinear shape across multiple transverse planes. Based on a stick decomposition of a local rectangular neighborhood, a nonlinear derivative operator perpendicular to each stick is defined. Then, combining with a standard deviation of the intensity along the stick, the composed likelihood function will take a strong response to fissure-like bright lines, and tends to suppress undesired structures including large vessels, step edges and blobs. Applying the 2D filter sequentially to the sagittal, coronal and axial slices, an approximate 3D co-planar constraint is implicitly exerted through the cascaded pipeline, which helps to further eliminate non-fissure tissues. To generate a clear fissure segmentation, we adopt a connected component based post-processing scheme, combined with a branch-point finding algorithm to disconnect the residual adjacent clutters from the fissures. The performance of our filter has been verified in experiments with a 23 patients dataset, where pathologies to different extents are included. The DoS filter compared favorably with prior algorithms.

**5. TOPIC**: Lung lobe segmentation in volumetric X-ray CT images.

**AUTHOR’S**: L. Zhang, E. Hoffman, and J. M. Reinhardt.

**DESCRIPTION**: High resolution X-ray computed tomography (CT) imaging is routinely used for clinical pulmonary applications. Since lung function varies regionally and because pulmonary disease is usually not uniformly distributed in the lungs, it is useful to study the lungs on a lobe-by-lobe basis. Thus, it is important to segment not only the lungs, but the lobar fissures as well. In this paper we demonstrate the use of an anatomic pulmonary atlas, encoded with a priori information on the pulmonary anatomy, to automatically segment the oblique lobar fissures. Sixteen volumetric CT scans from 16 subjects are used to construct the pulmonary atlas. A ridgeness measure is applied to the original CT images to enhance the fissure contrast. Fissure detection is accomplished in two stages: an initial fissure search and a final fissure search. A fuzzy reasoning system is used in the fissure search to analyze information from three sources: the image intensity, an anatomic smoothness constraint, and the atlas-based search initialization. Our method has been tested on 22 volumetric thin-slice CT scans from 12 subjects, and the results are compared to manual tracings. Averaged across all 22 data sets, the RMS error between the automatically-segmented and manually-segmented fissures is 1.96 +/- 0.71 mm and the mean of the similarity indices between the manually-defined and computer-defined lobe regions is 0.988. The results indicate a strong agreement between the automatic and manual lobe segmentations.

**EXISTING WORK:**

This work presents a systematic review of techniques for the 3D automatic detection of pulmonary nodules in computerized-tomography (CT) images. Its main goals are to analyze the latest technology being used for the development of computational diagnostic tools to assist in the acquisition, storage and, mainly, processing and analysis of the biomedical data. Also, this work identifies the progress made, so far, evaluates the challenges to be overcome and provides an analysis of future prospects. As far as the authors know, this is the first time that a review is devoted exclusively to automated 3D techniques for the detection of pulmonary nodules from lung CT images, which makes this work of noteworthy value. The research covered the published works in the Web of Science, PubMed, Science Direct and IEEEXplore up to December 2014. Each work found that referred to automated 3D segmentation of the lungs was individually analyzed to identify its objective, methodology and results. Based on the analysis of the selected works, several studies were seen to be useful for the construction of medical diagnostic aid tools.

**EXISTING DRAWBACKS:**

* low algorithm sensitivity
* little large number of false positives
* detection of different kinds of nodules with different sizes and shapes are need to improve.

**PROPOSED SYSTEM:**

we propose a new coarse-tofine deep learning segmentation approach called FissureNet. FissureNet achieves superior segmentation performance compared to other methods by concatenating two Seg3DNet ConvNets. The new Seg3DNet1 architecture is less memoryintensive compared to U-Net and SegNet, enabling it to learn global contextual information from entire lung images. Seg3DNet is a generic 3D segmentation network suitable for many applications. Within FissureNet, the first Seg3DNet is trained to detect an approximate fissure region of interest (ROI) and the second Seg3DNet is trained to detect precise fissure location within the ROI. The coarse-to-fine approach used by FissureNet overcomes the challenges associated with training a network to segment a thin structure that represents a very small fraction of the total voxel count.

**PROPOSED WORK IMPROVEMENTS:**

* Increasing algorithm sensitivity.
* Reducing the number of false positives.
* Improving and optimizing the algorithm detection of different kinds of nodules with different sizes and shapes.
* Image processing technique is a time saving process.

**ARCHITECTURE DIAGRAM:**

Input image in 3dim is converted into 2D by gray scalepreprocessing

Gray scale conversion

Filtering

Browse for an input image

|  |
| --- |
| CT scan image (jpg) |

2D image is undergoes for noise filtering by Gaussian filter

|  |
| --- |
| Filtered image(.fig) |

Masking the left and right lung region

Masking

Feature Extraction

Identified fissure is extracted for analysis

|  |
| --- |
| Segmented image(.fig) |

Masked image is split for fissure identification

Segmentation

|  |
| --- |
| Binary image(.fig) |

CNN

Identification of normal or abnormal case

Analysis had done by set of training set and testing set over neural network

|  |
| --- |
| Decision(.text) |

Prediction of pulmonary abnormalities using convolutional neural network.

**PSEUDO CODE (GAUSSIAN FILTER)**

For one dimensional,

**F(x)=1/d\*sqrt (2\*pi)[e^-(x-mu)^2/2d^2]**

In one dimensional ,the center point is orgin.so mu is zero(mu= 0)

**F(x)=1/d\*sqrt (2\*pi)[e^-(x)^2/2d^2]**

For 2 dimensional,

**F(x, y)=1/(d\*sqrt (2\*pi))^2[e^-(x^2 + y^2)^2/2d^2]**

**F(x, y) =1/(2\*pi \*d^2)[e^-(x^2 + y^2)^2/2d^2]**

X -> 1 dimensional state

Y -> 2 dimensional state

d -> input percentage for filtering

Pi -> 3.14

Mu -> average of x

**EVALUATION METRIES:**

Input image: CT scan image(.jpg)

Gray-scale image: jpg image of 3 dimensional is converted to 2 dimensional by converting to gray-scale

Filtered image: noise removal in gray image by gaussian filter.

Segmentation: The image is segmented by segmentation process by segmented the nodule in lung.

Feature extraction: Feature extraction a type of dimensionality reduction that efficiently represents interesting parts of an image as a compact feature vector. This approach is useful when image sizes are large and a reduced feature representation is required to quickly complete tasks such as image matching and retrieval.

**MODULE DESCRIPTION:**

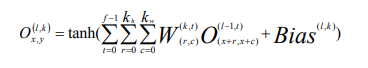
**i) OVERVIEW:**

.We model fissure detection as a probabilistic classification problem. Given a dataset X and a finite class set Y , a probabilistic classifier models the conditional probability distribution P(Y |X). That is, given a feature vector x ∈ X, the classifier predicts a probability distribution over the class set Y . The features and the conditional probability distribution are learned jointly through end-to-end training of a Seg3DNet. For pulmonary fissure classification the class set Y consists of the three fissures and a non-fissure class, such that all voxels that are not fissure are assigned to the non-fissure class. The number of fissure voxels is very small compared to the number of non-fissure voxels; there is approximately one fissure voxel for every 100 non-fissure voxels within the lung mask (at the image resolution used in this study). FissureNet uses a coarse-to-fine approach by cascading two Seg3DNets (Fig. 1). The first Seg3DNet is trained to detect an approximate fissure region of interest (ROI) and the second Seg3DNet is trained to detect the precise fissure location within the ROI. Separate pipelines are trained for the left and right lungs, yielding four total Seg3DNet classifiers: left fissure ROI, right fissure ROI, left fissure, and right fissure. The proposed Seg3DNet architecture is illustrated.

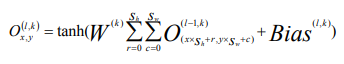
**ii)CNN ALGORITHM**

**1.Forward pass:**

output of neuron of row k , column y in the convolution layer and



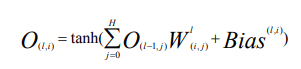
among them, f is the number of convolution cores in a feature pattern. output of neuron of row x , column y in the l th sub sample layer and k th feature pattern



the output of the j th neuron in l th hide layer H ：



among them, sis the number of feature patterns in sample layer. output of the i th neuron l th output layer F

****

**2.Back propagation**

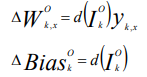
output deviation of the k th neuron in output layer O :



input deviation of the k th neuron in output layer:



weight and bias variation of k th neuron in output O :



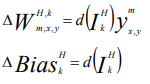
output bias of k th neuron in hide layer H :



input bias of k th neuron in hide layer H :



weight and bias variation in row x , column y in the m th feature pattern ,a former layer in front of k neurons in hide layer H



Output bias of row x , column y in m th feature pattern ,sub sample layer S



Input bias of row x , column y in m th feature pattern , sub sample

layer S



weight and bias variation of row x , column y in m th feature pattern ,sub sample layer S



among them, C represents convolution layer.



output bias of row x ,column y in k th feature patter ,convolution layer C



output bias of row x ,column y in k th feature patter ,convolution layer C



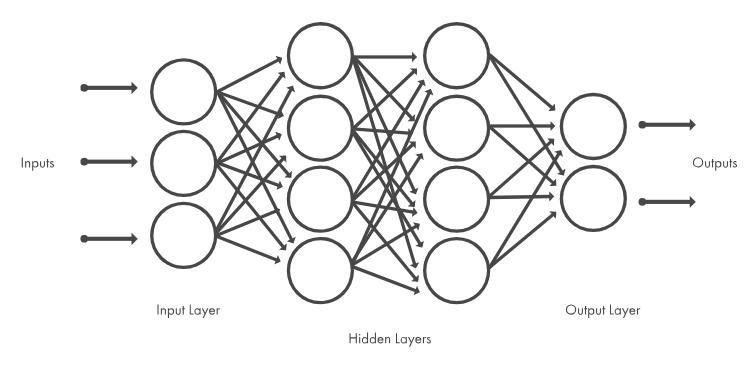
weight variation of row r ,column c in m th convolution core,corresponding to k th feature pattern in l th layer ,convolution C

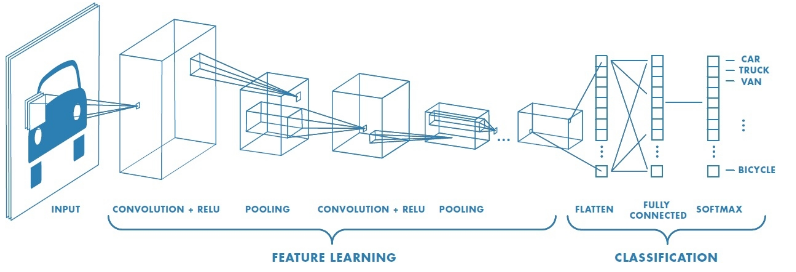


total bias variation of the convolution core



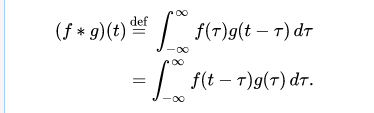
# CONVOLUTIONAL NEURAL NETWORK (CNN)

These layers perform operations that alter the data with the intent of learning features specific to the data. Three of the most common layers are: convolution, activation or ReLU, and pooling.



**1. Convolution**

A [convolution](https://en.wikipedia.org/wiki/Convolution) is a combined integration of two functions that shows you how one function modifies the other.



There are three important items to mention in this process: the input image, the feature detector, and the feature map. The input image is the image being detected. The feature detector is a matrix, usually 3x3 (it could also be 7x7). A **feature detector** is also referred to as a kernel or a filter.

Intuitively, the matrix representation of the input image is multiplied element-wise with the feature detector to produce a feature map, also known as a convolved feature or an activation map. The aim of this step is to reduce the size of the image and make processing faster and easier. Some of the features of the image are lost in this step.

However, the main features of the image that are important in image detection are retained. These features are the ones that are unique to identifying that specific object. For example each animal has unique features that enable us to identify it. The way we prevent loss of image information is by having many feature maps. Each feature map detects the location of certain features in the image.

## 2. Apply the ReLu (Rectified Linear Unit)

In this step we apply the [rectifier function](https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning) to increase non-linearity in the CNN. Images are made of different objects that are not linear to each other. Without applying this function the image classification will be treated as a linear problem while it is actually a non-linear one.

## ****3. Pooling****

Spatial invariance is a concept where the location of an object in an image doesn’t affect the ability of the neural network to detect its specific features. Pooling enables the CNN to detect features in various images irrespective of the difference in lighting in the pictures and different angles of the images.

There are different types of pooling, for example, max pooling and min pooling. [Max pooling](https://computersciencewiki.org/index.php/Max-pooling_/_Pooling) works by placing a matrix of 2x2 on the feature map and picking the largest value in that box. The 2x2 matrix is moved from left to right through the entire feature map picking the largest value in each pass.

These values then form a new matrix called a pooled feature map. Max pooling works to preserve the main features while also reducing the size of the image. This helps reduce overfitting, which would occur if the CNN is given too much information, especially if that information is not relevant in classifying the image.

## 4. Flattening

Once the pooled featured map is obtained, the next step is to flatten it. [Flattening](https://www.superdatascience.com/convolutional-neural-networks-cnn-step-3-flattening/) involves transforming the entire pooled feature map matrix into a single column which is then fed to the neural network for processing.

## 5. Full connection

After flattening, the flattened feature map is passed through a neural network. This step is made up of the input layer, the fully connected layer, and the output layer. The fully connected layer is similar to the hidden layer in ANNs but in this case it’s fully connected. The output layer is where we get the predicted classes. The information is passed through the network and the error of prediction is calculated. The error is then backpropagated through the system to improve the prediction.

The final figures produced by the neural network don’t usually add up to one. However, it is important that these figures are brought down to numbers between zero and one, which represent the probability of each class. This is the role of the Softmax function.

**6.softmax**

In [mathematics](https://en.m.wikipedia.org/wiki/Mathematics), the **softmax function,** also known as **softargmax** or **normalized exponential function**,198 is a function that takes as input a vector of *K* real numbers, and normalizes it into a [probability distribution](https://en.m.wikipedia.org/wiki/Probability_distribution) consisting of *K* probabilities proportional to the exponentials of the input numbers. That is, prior to applying softmax, some vector components could be negative, or greater than one; and might not sum to 1; but after applying softmax, each component will be in the [interval](https://en.m.wikipedia.org/wiki/Interval_(mathematics)) {\displaystyle (0,1)} and the components will add up to 1, so that they can be interpreted as probabilities. Furthermore, the larger input components will correspond to larger probabilities. Softmax is often used in [neural networks](https://en.m.wikipedia.org/wiki/Artificial_neural_network), to map the non-normalized output of a network to a probability distribution over predicted output classes.

**iii)Seg3DNet :**

In this work, we propose a 3D ConvNet architecture for image segmentation called Seg3DNet (Fig. 1). Seg3DNet consists of an encoder which generates a high dimensional feature representation of the image, and a decoder which decodes the features to produce a segmentation. Unlike many segmentation architectures, the encoder and decoder modules in Seg3DNet are asymmetrical. The encoder module consists of L resolution levels li for i = 0, 1, ..., L − 1, where the activation maps in level li are downsampled by a factor of 2 i relative to the full resolution level l0. Each level of the encoder has two convolutional layers followed by a max-pooling layer. All convolutional layers use 3 × 3 × 3 voxel kernels, and the number of kernels in level li is given by Ni = 2i+5. After the second convolution layer of each level, max pooling with kernel size 2×2×2 and stride of 2 produces the downsampling factor of 2 between levels. While recent ConvNet architectures have eliminated pooling layers, downsampling is necessary to achieve a global receptive field on large input volumes. To mitigate the loss of precise localization information from the pooling layers, the decoder network combines representations from all scale levels. The decoder module condenses the representation at each scale level to a single activation map using a convolutional layer with a single voxel kernel of size 1 × 1 × 1 × Ni . The lower resolution activation maps are upsampled to full resolution using nearest neighbor interpolation followed by a convolution with filter size 2 i + 1, effectively performing a variant of deconvolution [36]. The resulting activation maps, one from each scale level, are concatenated along the feature dimension to form a multi-scale representation. Two more convolutional layers are used to combine information from different scales. The representation at the last layer of the Seg3DNet has |Y | activation maps each with the same spatial dimensions as the input volume. The output at spatial location x of activation map y, fy(x), is interpreted as an unnormalized log probability of x belonging to class y. The softmax vector nonlinearity is used to obtain the conditional probability distribution, given by:

P(Y = y|x) = e fy(x) P j∈Y e fj (x) . (1)

We denote the probability for each class y ∈ Y as Py(x). By construction, Y is a valid probability distribution function with Py(x) ∈ [0, 1] ∀y ∈ Y , and P y∈Y Py(x) = 1. For the right lung ConvNets, we define separate classes to distinguish between oblique and horizontal fissures. Therefore, the class set cardinality for the left and right lung ConvNets is |Y | = 2 and |Y | = 3, respectively. Batch normalization [37] and ReLU nonlinearities [38] are used after each convolution layer with the exception of the last layer. All convolutional layers use zero-padding to prevent reduction in spatial dimensions.

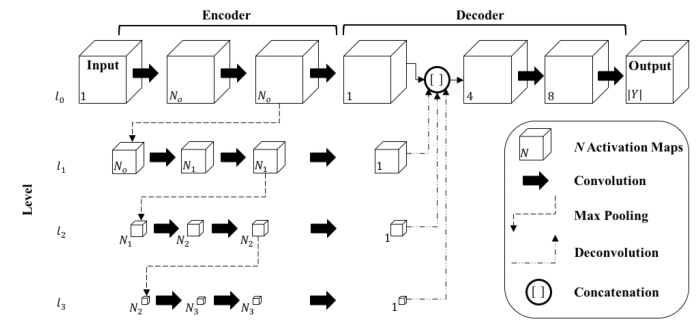


Fig. 1: Proposed Seg3DNet architecture. Each arrow represents an operation performed by a layer and each cube represents the intermediate feature representations produced by a layer. For visualization purposes, only the spatial dimensions of the feature representations are illustrated. The number of activation maps (size of channel dimension) is denoted in the lower left corner. For the encoder module, we define Ni = 2i+5 so that the number of activation maps increases by a factor of two at each level. The number of kernels used in each convolutional layer can be inferred by the number of activation maps in the layer’s output representation, e.g., the first convolutional layer has N0 = 20+5 = 32 kernels. The relative spatial size of the activation maps are drawn to scale. At each level the feature representation is spatially downsampled by a factor of two. Batch normalization and ReLU nonlinearity are performed after each convolution except the last.

**iv)FissureNet :**

As shown in Fig. 2, FissureNet has two parallel pipelines, each of which is a coarse-to-fine cascade of two Seg3DNets. The first Seg3DNet is trained to detect a fissure ROI. The original ground truth fissure segmentations are modified to produce the fissure ROI training labels. A voxel belongs to the fissure ROI if it is located within 5 mm of the corresponding fissure, otherwise it is non-fissure. This dilation of the singlevoxel ground truth reduces the class skewness. Additionally, by dilating the ground truth fissure the network is able to focus on global patterns rather than precise fissure appearance. As a result, the network is more robust to weak and radiographically incomplete fissures. The fissure ROI allows for small imperfections in the training data which are expected due to the nature of manually tracing a single voxel curve. For training the first Seg3DNet, we define the loss associated with each voxel using categorical cross entropy of the form

L(x, Y ) = − X y∈Y ty(x) log Py(x), (2)

where ty(x) represents a one-hot encoding of the target label for voxel x and class y, i.e., ty(x) is one when y corresponds to the true class and zero for all other classes The total loss for an input image is given by

LROI = P x∈Ω L(x, YROI) |Ω| , (3)

where Ω is the input image domain and YROI is ROI classifier class set. The second Seg3DNet is trained to detect the precise fissure location. The original ground truth fissure segmentations are used as training labels. The loss associated with each voxel is the same as the first Seg3DNet (2). However, the total loss is a weighted average using the probability that the voxel is in a fissure

ROI LF = P x∈Ω (1 − PNR(x))L(x, YF) P x∈Ω (1 − PNR(x)) , (4)

where PNR(x) is the probability that voxel x is non-fissure ROI as predicted by the first Seg3DNet and YF is the fissure classifier class set. This weighting limits the contribution of the large number of non-fissure voxels to the loss function, mitigating the class imbalance problem while allowing for precise fissure localization.

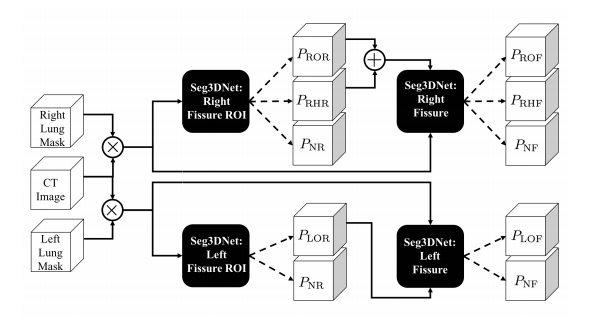


Fig. 2: FissureNet: coarse-to-fine network cascade. Black boxes represent four Seg3DNet classifiers, each trained for a different classification task as indicated in the box. For the right lung pipeline (top), the CT image is masked with the right lung mask and input to the right fissure ROI Seg3DNet. The output of the ROI Seg3DNet represents the probability that each voxel is right oblique ROI (PROR), right horizontal ROI (PRHR), and non-fissure ROI (PNR). The input to the right fissure Seg3DNet is the masked CT image and the probability maps PROR or PRHR. The output of the right fissure Seg3DNet gives the probability that each voxel is right oblique fissure (PROF ), right horizontal fissure (PRHF ), and non-fissure (PNF ). The left lung pipeline (bottom) is similar, except each classifier only predicts two classes corresponding to left oblique fissure and non-fissure.

**SYSTEM REQUIREMENT:**

**HARDWARE REQUIREMENTS:**

Processor : DUAL CORE 2.5GHZ

Ram : 1 GB SD RAM

Monitor : 15” COLOR

Hard Disk : 80 GB

Keyboard : STANDARD 102 KEYS

Mouse : 3 BUTTONS

**SOFTWARE CONFIGURATION:**

Operating System : Windows xp Professional

Environment : MATLAB

Mat lab : Version 18a

**SOFTWARE TOOLS AND METHODOLOGY**

**ABOUT MATLAB**

MATLAB is a software package for high performance numerical computation and visualization. It provides an interactive environment with hundreds of built-in functions for technical computation, graphics and animation. Best of all, it provides easy extensibility with its own high-level programming language. The name MATLAB stands for Matrix Laboratory. The basic building block of MATLAB is the matrix. The fundamental data type is the array.

MATLABs built-in functions provide excellent tools for linear algebra computations, data analysis, signal processing, optimization, numerical solutions of ODES, quadrature and many other types of scientific computations. Most of these functions use the state-of-the art algorithms. There are numerous functions for 2-D and 3-D course, MATLAB even provides an external interface to run those programs from within MATLAB. The user, however, is not limited to the built-in functions, he can write his own functions in the MATLAB language. Once written, these functions behave just like the built-in functions. MATLAB’s language is very easy to learn and to use.

**MATLAB TOOLBOXES**

There are several optional ‘Toolboxes’ available from the developers of the MATLAB. These tool boxes are collection of functions written for special applications such as Symbolic Computations Toolbox, Image Processing Toolbox, Statistics Toolbox, and Neural Networks Toolbox, Communications Tool box, Signal Processing Toolbox, Filter Design Toolbox, Fuzzy Logic Toolbox, Wavelet Toolbox, Data base Toolbox, Control System Toolbox, Bioinformatics Toolbox, Mapping Toolbox.

**BASICS OF MATLAB**

**MATLAB WINDOWS**

On all UNIX systems, Macs, and PC, MATLAB works through three basic windows. They are:

**a. Command window**:

This is the main window. The MATLAB command prompt characterizes it ‘>>’.when you launch the application program, MATLAB puts you in this window .All commands, including those for running user-written programs, are typed in this window at the MATLAB prompt.

**b. Graphics window:**

The output of all graphics commands are typed in the command window are flushed to the graphics or figure window, a separate gray window with(default) white background color. The user can create as many figure windows, as the system memory will allow.

**c. Edit window:**

This is where you write, edit, create, and save your own programs in files called M-files. We can use any text editor to carry out these tasks. On the most systems, such as PC’s and Macs, MATLAB provides its build in editor. On other systems, you can invoke the edit window by typing the standard file editing command that you normally use on your systems. The command is typed at the MATLAB prompt following the special character ‘!’ character. After editing is completed, the control is returned to the MATLAB.

**On-Line Help**

**a. On-line documentation:**

MATLAB provides on-line help for all its built-in functions and programming language constructs. The commands look for, help, help win, and helpdesk provides on-line help.

**b. Demo:**

MATLAB has a demonstration program that shows many of its features. The program includes a tutorial introduction that is worth trying. Type demo at the MATLAB prompt to invoke the demonstration program, and follow the instruction on the screen.

**Input-Output**

MATLAB supports interactive computation taking the input from the screen, and flushing the output to the screen. In addition, it can read input files and write output files. The following features hold for all forms of input-output.

**a. Data type**

The fundamental data type in the MATLAB is the array. It encompasses several distinct data objects-integers, doubles, matrices, character strings, and cells. In most cases, however, we never have to worry about the data type or the data object declarations. For example there is no need to declare variables, as real or complex .When a real number is entered as the variable, MATLAB automatically sets the variable to be real.

**b. Dimensioning**

Dimensioning is automatic in MATLAB. No dimensioning statements are required for vectors or arrays. We can find the dimension of an existing matrix or a vector with the size and length commands.

**C. Case sensitivity**

MATLAB is case sensitive i.e. it differentiates between the lower case and the uppercase letters. Thus A is different variables. Most MATLAB commands are built-in function calls are typed in lower case letters. We can turn case sensitivity on and off with casesen command.

**d. Output display**

The output of every command is displayed on the screen unless MATLAB is directed otherwise. A semicolon at the end of a command suppresses the screen output, except for graphics and on-line help command. The following facilities are provided for controlling the screen output.

**i. Paged output**

To direct the MATLAB to show one screen of output at a time, type more on the MATLAB prompt. Without it, MATLAB flushes the entire output at once, without regard to the speed at which we read.

**ii. Output format**

Though computations inside the MATLAB are performed using the double precision, the appearance of floating point numbers on the screen is controlled by the output format in use. There are several different screen output formats. The following table shows the printed value of 10pi in different formats.

|  |
| --- |
| Format short 31.4159 |
| Format short e 3.1416e+01 |
| Format long 31.41592653589793 |
| Format long e 3.141592653589793e+01 |
| Format short g 31.416 |
| Format long g 31.4159265358979 |
| Format hex 403f6a7a2955385e |
| Format rat 3550/113 |
| Format bank 31.42 |

**e. Command History**

MATLAB saves previously typed commands in a buffer. These commands can be called with the up-arrow key. This helps in editing previous commands. You can also recall a previous command by typing the first characters and then pressing the up-arrow key. On most UNIX systems, MATLABS command line editor also understands the standard emacs key bindings.

**File Types**

 MATLAB has three types of files for storing information

**M-files**: M-files are standard ASCII text files, with a .m extension to the file name. There are two types of these files: script files and function files. Most programs we write in MATLAB are saved as M-files. All built-in functions in MATLAB are M-files, most of which reside on our computer in precompiled format. Some built in functions are provided with source code in readable M-files so that can be copied and modified.

**Mat-files**: Mat-files are binary data-files with a .mat extension to the file name. Mat-files are created by MATLAB when we save data with the save command. The data is written in a special format that only MATLAB can read. Mat-files can be loaded into MATLAB with the load command.

**Mex-files**: Mex-files are MATLAB callable FORTRAN and C programs, with a.mex extension to the file name. Use of these files requires some experience with MATLAB and a lot of patience.

**Platform independence**

One of the best features of MATLAB is its platform-independence. Programs written in the MATLAB language work exactly the same way on all computers. The user interface however, varies from platform to platform. For example, on PC’s and Macs there are menu driven commands for opening, writing, editing, saving and printing files whereas on Unix machines such as sun workstations, these tasks are usually performed with Unix commands.

**Images in MATLAB**

The project has involved understanding data in MATLAB, so below is a brief review of how images are handled. Indexed images are represented by two matrices, a color map matrix and image matrix.

 (i)The color map is a matrix of values representing all the colours in the image.

(ii)The image matrix contains indexes corresponding to the colour map color map.

 A color map matrix is of size N\*3, where N is the number of different colors I the image. Each row represents the red, green, blue components for a colour.

E.g. the matrix

Represents two colors, the first have components r1, g1, b1 and the second having the components r2, g2, b2

The wavelet toolbox only supports indexed images that have linear, monotonic color maps. Often color images need to be pre-processed into a grey scale image before using wavelet decomposition. The Wavelet Toolbox User’s Guide provides some sample code to convert color images into grey scale. This will be useful if it is needed to put any images into MATLAB.

 This chapter dealt with introduction to MATLAB software which we are using for our project. The 2-D wavelet Analysis, the decomposition of an image into approximations and details and the properties of different types of wavelets will be discussed in the next chapter.

**MATLAB**

* Matlab is a high-performance language for technical computing.
* It integrates computation, programming and visualization in a user-friendly environment where problems and solutions are expressed in an easy-to-understand mathematical notation.
* Matlab is an interactive system whose basic data element is an array that does not
* Require dimensioning.
* This allows the user to solve many technical computing problems, especially those with matrix and vector operations, in less time than it would take to write a program in a scalar non-interactive language such as C or FORTRAN.
* Matlab features a family of application-specific solutions which are called toolboxes.
* It is very important to most users of Matlab, that toolboxes allow to learn and apply
* Specialized technology.
* These toolboxes are comprehensive collections of Matlab functions, so-called M-files that extend the Matlab environment to solve particular classes of problems.
* Matlab is a matrix-based programming tool. Although matrices often need not to be
* Dimensioned explicitly, the user has always to look carefully for matrix dimensions.
* If it is not defined otherwise, the standard matrix exhibits two dimensions n × m.
* column vectors and row vectors are represented consistently by n × 1 and 1 × n matrices, respectively.

**MATLAB OPERATIONS**

* Matlab operations can be classified into the following types of operations:
  + Arithmetic and logical operations,
  + Mathematical functions,
  + Graphical functions, and
  + Input/output operations.
* In the following sections, individual elements of Matlab operations are explained in detail.

**EXPRESSIONS**

* Like most other programming languages, Matlab provides mathematical expressions,

But unlike most programming languages, these expressions involve entire matrices. The building blocks of expressions are

* Variables
* Numbers
* Operators
* Functions

**VARIABLES**

* Matlab does not require any type declarations or dimension statements.
* When a new variable name is introduced, it automatically creates the variable and allocates the appropriate amount of memory.
* If the variable already exists, Matlab changes its contents and, if necessary, allocates new storage.
* For example
* >> books = 10
* It creates a 1-by-1 matrix named books and stores the value 10 in its single element.
* In the expression above, >> constitutes the Matlab prompt, where the commands can be entered.
* Variable names consist of a string, which start with a letter, followed by any number of letters, digits, or underscores. Matlab is case sensitive; it distinguishes between uppercase and lowercase letters. A and a are not the same variable.
* To view the matrix assigned to any variable, simply enter the variable name.

**NUMBERS**

* Matlab uses the conventional decimal notation.
* A decimal point and a leading plus or minus sign is optional. Scientific notation uses the letter e to specify a power-of-ten scale factor.
* Imaginary numbers use either i or j as a suffix.
* Some examples of legal numbers are:

7 -55 0.0041 9.657838 6.10220e-10 7.03352e21 2i -2.71828j 2e3i 2.5+1.7j.

**OPERATORS**

Expressions use familiar arithmetic operators and precedence rules. Some examples are:

* + Addition
* - Subtraction
* Multiplication
* / Division
* ’ Complex conjugate transpose
* ( ) Brackets to specify the evaluation order.

**FUNCTIONS**

* Matlab provides a large number of standard elementary mathematical functions, including sin, sqrt, expand abs.
* Taking the square root or logarithm of a negative number does not lead to an error; the appropriate complex result is produced automatically.
* Matlab also provides a lot of advanced mathematical functions, including Bessel and Gamma functions. Most of these functions accept complex arguments.
* For a list of the elementary mathematical functions, type
* >> help elfun
* Some of the functions, like sqrt and sin are built-in. They are a fixed part of the Matlab core so they are very efficient.
* The drawback is that the computational details are not readily accessible. Other functions, like gamma and sinh, are implemented in so called M-files.
* You can see the code and even modify it if you want.

**MATLAB IMAGE PROCESSING:**

## INTRODUCTION

When working with images in Matlab, there are many things to keep in mind such as loading an image, using the right format, saving the data as different data types, how to display an image, conversion between different image formats, etc. This worksheet presents some of the commands designed for these operations. Most of these commands require you to have the Image processing tool box installed with Matlab. To find out if it is installed, type over at the Matlab prompt. This gives you a list of what tool boxes that are installed on your system.

For further reference on image handling in Matlab you are recommended to use Mat lab’s help browser. There is an extensive (and quite good) on-line manual for the Image processing tool box that you can access via Mat lab’s help browser.

The first sections of this worksheet are quite heavy. The only way to understand how the presented commands work is to carefully work through the examples given at the end of the worksheet. Once you can get these examples to work, experiment on your own using your favorite image!

## FUNDAMENTALS

A digital image is composed of pixels which can be thought of as small dots on the screen. A digital image is an instruction of how to color each pixel. We will see in detail later on how this is done in practice. A typical size of an image is 512-by-512 pixels. Later on in the course you will see that it is convenient to let the dimensions of the image to be a power of 2. For example, 29=512. In the general case we say that an image is of size m-by-n if it is composed of m pixels in the vertical direction and n pixels in the horizontal direction.

Let us say that we have an image on the format 512-by-1024 pixels. This means that the data for the image must contain information about 524288 pixels, which requires a lot of memory! Hence, compressing images is essential for efficient image processing. You will later on see how Fourier analysis and Wavelet analysis can help us to compress an image significantly. There are also a few "computer scientific" tricks (for example entropy coding) to reduce the amount of data required to store an image.

## IMAGE FORMATS SUPPORTED BY MATLAB

The following image formats are supported by Matlab:

* BMP
* HDF
* JPEG
* PCX
* TIFF
* XWB

Most images you find on the Internet are JPEG-images which is the name for one of the most widely used compression standards for images. If you have stored an image you can usually see from the suffix what format it is stored in. For example, an image named myimage.jpg is stored in the JPEG format and we will see later on that we can load an image of this format into Matlab.

## WORKING FORMATS IN MATLAB

If an image is stored as a JPEG-image on your disc we first read it into Matlab. However, in order to start working with an image, for example perform a wavelet transform on the image, we must convert it into a different format. This section explains four common formats.

#### INTENSITY IMAGE (GRAY SCALE IMAGE)

This is the equivalent to a "gray scale image" and this is the image we will mostly work with in this course. It represents an image as a matrix where every element has a value corresponding to how bright/dark the pixel at the corresponding position should be colored. There are two ways to represent the number that represents the brightness of the pixel: The double class (or data type). This assigns a floating number ("a number with decimals") between 0 and 1 to each pixel.

The value 0 corresponds to black and the value 1 corresponds to white. The other class is called uint8 which assigns an integer between 0 and 255 to represent the brightness of a pixel. The value 0 corresponds to black and 255 to white. The class uint8 only requires roughly 1/8 of the storage compared to the class double. On the other hand, many mathematical functions can only be applied to the double class. We will see later how to convert between double and uint8.

#### BINARY IMAGE

This image format also stores an image as a matrix but can only color a pixel black or white (and nothing in between). It assigns a 0 for black and a 1 for white.

#### INDEXED IMAGE

This is a practical way of representing color images. (In this course we will mostly work with gray scale images but once you have learned how to work with a gray scale image you will also know the principle how to work with color images.) An indexed image stores an image as two matrices. The first matrix has the same size as the image and one number for each pixel. The second matrix is called the color map and its size may be different from the image. The numbers in the first matrix is an instruction of what number to use in the color map matrix.

#### RGB IMAGE

This is another format for color images. It represents an image with three matrices of sizes matching the image format. Each matrix corresponds to one of the colors red, green or blue and gives an instruction of how much of each of these colors a certain pixel should use.

#### MULTIFRAME IMAGE

In some applications we want to study a sequence of images. This is very common in biological and medical imaging where you might study a sequence of slices of a cell. For these cases, the multiframe format is a convenient way of working with a sequence of images. In case you choose to work with biological imaging later on in this course, you may use this format.

#### How to convert between different formats

The following table shows how to convert between the different formats given above. All these commands require the Image processing tool box!

**IMAGE FORMAT CONVERSION**

(Within the parenthesis you type the name of the image you wish to convert.)

|  |  |
| --- | --- |
| Operation: | Matlab command: |
| Convert between intensity/indexed/RGB formats to binary format. | dither() |
| Convert between intensity format to indexed format. | gray2ind() |
| Convert between indexed format to intensity format. | ind2gray() |
| Convert between indexed format to RGB format. | ind2rgb() |
| Convert a regular matrix to intensity format by scaling. | mat2gray() |
| Convert between RGB format to intensity format. | rgb2gray() |
| Convert between RGB format to indexed format. | rgb2ind() |

The command mat2gray is useful if you have a matrix representing an image but the values representing the gray scale range between, let's say, 0 and 1000. The command mat2gray automatically re scales all entries so that they fall within 0 and 255 (if you use the uint8 class) or 0 and 1 (if you use the double class).

## How to convert between double and uint8

When you store an image, you should store it as a uint8 image since this requires far less memory than double. When you are processing an image (that is performing mathematical operations on an image) you should convert it into a double. Converting back and forth between these classes is easy.

I=im2double (I);

Converts an image named I from uint8 to double.

I=im2uint8 (I);

An image named I from double to uint8.

## READ MATLAB FILE:

When you encounter an image you want to work with, it is usually in form of a file (for example, if you down load an image from the web, it is usually stored as a JPEG-file). Once we are done processing an image, we may want to write it back to a JPEG-file so that we can, for example, post the processed image on the web. This is done using the imread and imwrite commands. These commands require the Image processing tool box!

Reading and writing image files

|  |  |
| --- | --- |
| Operation: | Matlab command: |
| Read an image.  (Within the parenthesis you type the name of the image file you wish to read.  Put the file name within single quotes ' '.) | imread() |
| Write an image to a file. (As the first argument within the parenthesis you type the name of the image you have worked with. As a second argument within the parenthesis you type the name of the file and format that you want to write the image to.  Put the file name within single quotes ' '.) | imwrite( , ) |

Make sure to use semi-colon ; after these commands, otherwise you will get LOTS OF number scrolling on your screen... The commands imread and imwrite support the formats given in the section "Image formats supported by Matlab" above.

## Loading and saving variables in Matlab

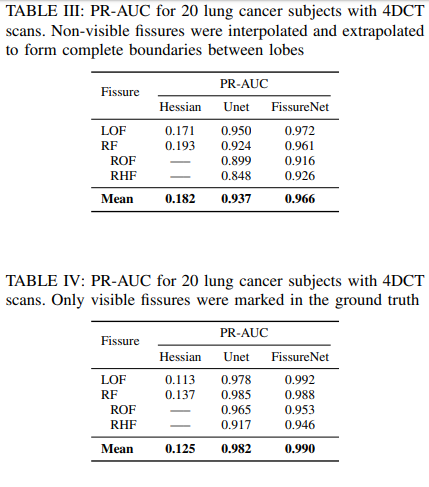
This section explains how to load and save variables in Matlab. Once you have read a file, you probably convert it into an intensity image (a matrix) and work with this matrix. Once you are done you may want to save the matrix representing the image in order to continue to work with this matrix at another time. This is easily done using the commands save and load. Note that save and load are commonly used Matlab commands, and works independently of what tool boxes that are installed.

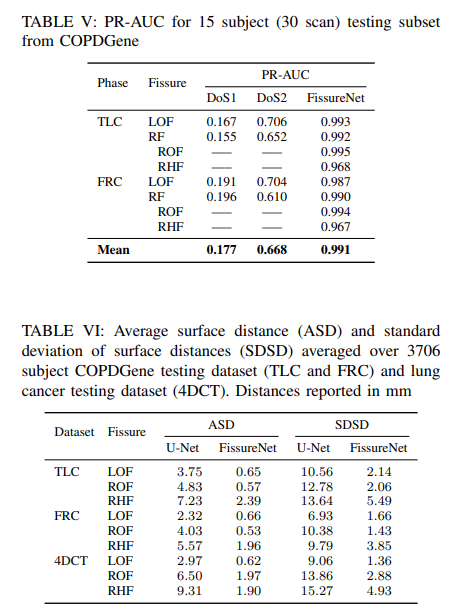
Loading and saving variables

|  |  |
| --- | --- |
| Operation: | Matlab command: |
| Save the variable X. | save X |
| Load the variable X. | load X |

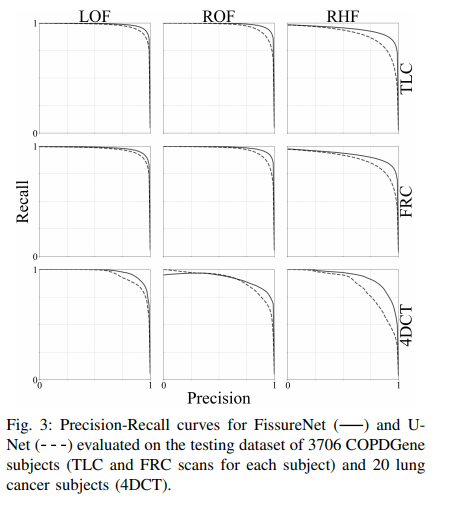
**RESULTS :**

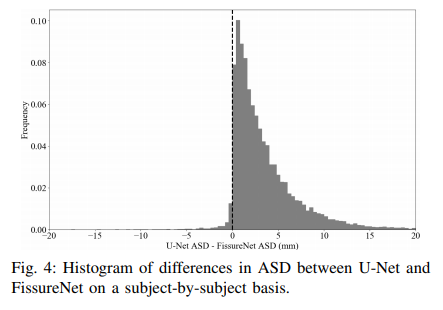
We compared FissureNet against three other fissure detection methods: Hessian-based [25], DoS [29], and U-Net [32]. The Hessian and the DoS methods do not distinguish between the right oblique and horizontal fissures. Therefore, only an aggregated right fissure (RF) measure is made for the right lung. For comparison, the RF measure is evaluated on FissureNet and U-Net by adding the ROF and RHF probabilities. Fig. 3 compares PR curves for FissureNet and U-Net methods on 3706 subjects (TLC and FRC scans for each subject) from COPDGene and 20 lung cancer subjects with 4DCT scans. PR-AUCs for FissureNet, U-Net, and Hessian are displayed in Tables II and III for the COPDGene and lung cancer datasets, respectively. Overall, PR-AUC for FissureNet, U-Net, and Hessian methods were 0.980, 0.963, and 0.158, respectively, on the COPDGene dataset and 0.966, 0.937, and 0.182, respectively, on the lung cancer dataset. All methods had similar performance on the COPDGene and the lung cancer datasets and FissureNet performed best with regards to PR-AUC. Table IV shows PR-AUCs on the lung cancer dataset using a ground truth which only indicates radiographically visible fissures. FissureNet and U-Net performed slightly better using the visible-only ground truth, while Hessian performed slightly worse. Table V shows PR-AUCs for FissureNet and DoS evaluated on a subset of 15 subjects (30 scans). The post-processing in the DoS method greatly improves the PRAUC from 0.177 (DoS1) to 0.668 (DoS2), however, FissureNet consistently performed better than DoS2 without any postprocessing with an overall PR-AUC of 0.991.

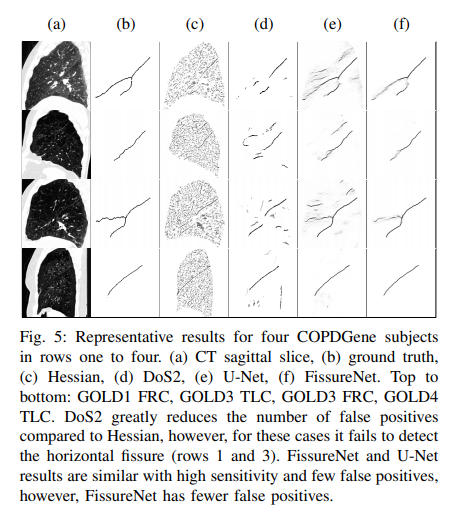




In Tables II-V, only LOF and RF fissures are included in mean calculation to avoid over-weighting right lung results. Table VI shows ASD and SDSD averaged over all subjects. On average, the ASD for FissureNet was less than U-Net for all scan types and fissures. Fig. 4 shows a histogram of the differences in ASD between U-Net and FissureNet on a subject-by-subject basis; 97% of the histogram area is to the right of the vertical line corresponding to cases where FissureNet has a lower ASD compared to U-Net.





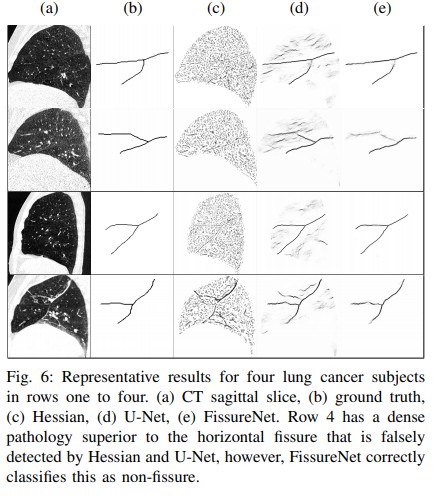


Statistical testing was performed to test for significant differences in performance between methods with regards to evaluation metrics. Paired t-tests showed that FissureNet had a significantly greater PR-AUC and a significantly lower ASD compared to U-Net on both the COPDGene and lung cancer datasets (p < 0.001). Additionally, FissureNet had a significantly greater PR-AUC compared to Hessian on both the COPDGene and lung cancer datasets (p < 0.001). Representative fissure detection results are displayed in Fig. 5 for the COPDGene dataset and Fig. 6 for the lung cancer dataset. These results show DoS2 and U-Net have far fewer false positives compared to Hessian, however, FissureNet produces the fewest false positives while maintaining high sensitivity. The difference in false positive behavior between FissureNet and U-Net is further emphasized in Fig. 7, where surface renderings are annotated in red to depict false positives. The only post-processing performed to generate the renderings was thresholding at the optimal PR-AUC thresholds.

**DISCUSSION :**

Existing fissure detection methods are limited to handcrafted and local features. These features typically suffer from low specificity as it is difficult to differentiate fissures from the other structures in the lung without global context. Additionally, it is difficult to design features that are robust against all fissure variations, especially for global compared to local features. To overcome the challenge of designing robust and discriminative features we use a deep learning approach to learn the feature detectors from labeled training cases.

The main challenges associated with training a ConvNet to detect fissures in CT images are the size of the input images and the highly skewed class distributions. The majority of ConvNets used in medical imaging applications use 2D image slices or use a sliding window approach with small image crops to overcome limitations in GPU memory. While this is a reasonable approach for some tasks, for fissure segmentation it is not desirable. The 3D appearance of a fissure is important to distinguish it from other structures that would otherwise appear similar on 2D slices. Global information provides additional context which is especially important when the fissure signal is weak, however, this information is not considered by patch-based approaches.



Compared to other segmentation architectures, Seg3DNet is an asymmetrical encoder-decoder network which uses less memory in order to accommodate a 3D network, larger input images, and more network levels. This allows for global information to be learned and results in higher specificity. By training separate Seg3DNets for the left and right lungs, we were able to reduce the size of the input image by a factor of two. This optimization does not degrade performance, as information from one lung does not provide global information for fissure detection in the other lung. To handle the class imbalance, we use a coarse-to-fine ConvNet cascade: the first ConvNet learns the fissure ROI and the second ConvNet learns the precise fissure location. In addition to mitigating the effect of class imbalance, the fissure ROI classifier is more sensitive to weak and incomplete fissures. Since the second training phase weights the voxel misclassification costs by the probability of being in the fissure ROI, the contribution of costs from the large number of nonfissure voxels is limited.

Therefore, the class imbalance problem is mitigated while allowing for precise fissure prediction. A similar, and more elegant, approach would be to train a single network with two outputs: one for the fissure ROI and one for precise fissure prediction. However, current limitations on GPU memory do not allow for this. Fully-connected layers are not used in Seg3DNet, making it a fully-convolutional network (FCN) [34]. This greatly reduces the number of parameters and makes the network less prone to overfitting; the proposed network has 3 million parameters compared to the popular VGG-16 network which has 138 million parameters. Furthermore, in a FCN the number of parameters is not dependent on the input image size, so the network can be trained and deployed on images of different sizes. Our network was trained on fixed-size image crops of 64x200x200 due to limited GPU memory, however, in some cases the entire lung field does not fit in this crop. At test time there is more memory available as mini-batches are not used and gradients do not need to be stored for backpropagation.

As a result, at test time much larger inputs can be used. In fact, the entire lung region, regardless of size, can be used as input and inference can be done in one forward pass per image. This is extremely efficient compared to patchwise approaches. In addition, Seg3DNet can accommodate different input image sizes, avoiding aggressive rescaling and interpolation that might degrade the fissure signal. This is the first study to evaluate a fissure detection method on a dataset of this size and diversity: 3706 COPDGene subjects with TLC and FRC scans and 20 lung cancer subjects with 4DCT scans. The COPDGene data used for training and evaluation came from 21 different institutions. Different scanner makes and models were used, as well as different reconstruction algorithms. The diversity of the evaluation set was further enriched with a lung cancer dataset of 4DCT scans. These scans were acquired at a lower dose during breathing, resulting in poorer image quality, motion blurring, and/or artifacts which were not present in the training dataset.

Robustness to such diversity is generally a challenge when designing rule-based algorithms for image segmentation: it can be difficult to achieve similar performance across different scanning protocols and diseases. Fissure detection performance was evaluated on four methods: Hessian-based, DoS, a deep learning approach using the U-Net architecture, and the proposed FissureNet. FissureNet and U-Net both greatly outperformed the Hessian and DoS methods on all datasets. Hessian and DoS methods were not able to detect weak fissures and produced many false positives at blood vessels and diseased regions. FissureNet consistently outperformed U-Net; while both methods demonstrated high sensitivity for fissure detection, FissureNet predicted fewer false positives. This can be attributed to the larger input patches and coarse-to-fine cascade, allowing the network to use more global context to differentiate true fissures from disease that resembles fissures. On the COPDGene evaluation dataset, all methods performed better on TLC scans compared to FRC scans in terms of PR-AUC. However, in the COPDGene trial the TLC scans were acquired at a higher dose and thus the image quality was better, so better performance was expected. In the future, comparing images of the lung at different inspiration levels acquired using the same dose would help determine which inspiration level is best for fissure detection. Although the performance on FRC images was worse, the FissureNet results are nonetheless impressive for lower dose scans.

This demonstrates the ability of FissureNet to generalize across different scanning protocols. The COPDGene dataset consisted of subjects with a wide range of disease severity, encompassing all GOLD stages. It is more challenging to detect fissures in heavily diseased cases as alterations in the underlying tissue can resemble the fissure and/or result in abnormal tissue appearance. Performance of FissureNet was robust to these challenges. Training a multi-class network for the right lung results in the ability to distinguish between oblique and horizontal fissures. This is the first fissure detection method to make this distinction. Since the ultimate goal is to divide the lungs into lobes, unique predictions for different fissures facilitates straightforward post-processing. A limitation of training FissureNet using a ground truth containing only oblique and horizontal fissures is an inability to detect accessory fissures. While accessory fissures have exhibit similar local appearance compared to the major fissures, the proposed FissureNet learns high level information encoded in the particular shapes and orientations of the oblique and horizontal fissures.

However, introducing an accessory fissure class and providing additional annotation in the training data could extend the network’s capability. Detection of the right horizontal fissure was consistently worse than the oblique fissures for the COPDGene dataset. The orientation of the horizontal fissure is often parallel with the axial imaging plane, potentially obscuring the fissure in CT images. It is not uncommon for horizontal fissures to be radiographically incomplete or missing, hindering identification even by human analysts. Interestingly, on the 4DCT dataset the ROF has a higher ASD compared to the RF. The COPDGene ground truth fissures used for evaluation have several limitations. The fissures were automatically extracted from lobar segmentations resulting in complete fissure boundaries for all cases even those with radiographically incomplete or missing fissures.

In such cases, the extrapolated or interpolated fissure location is highly subjective and evaluating the performance of any automated method using such a ground truth is limited in these regions. Furthermore, the ground truth fissures in the COPDGene evaluation dataset were generated using the same method as the training dataset (Apollo software followed by manual correction). This introduces a bias for learning-based methods to identify complete fissures in unseen subjects regardless of actual fissure integrity. An additional possible bias may be attributed to the FissureNet and U-Net methods being trained on the COPDGene dataset, while the Hessian and DoS methods were developed on an independent dataset. To address these limitations, evaluation was performed on a dataset of lung cancer subjects with 4DCT scans. The ground truth fissure segmentations for this dataset were generated manually.

Additionally, both complete and visible-only fissures were annotated. All fissure detection methods performed worse on the 4DCT dataset compared to the COPDGene dataset. The 4DCT scans use a lower dose and commonly have motion artifacts and blurring, resulting in decreased fissure visibility. All methods performed better using the visible-only fissure ground truth. A drawback of our method, and of deep learning in general, is the requirement of a large training dataset with ground truth segmentations. Manual segmentation is tedious, timeconsuming, and typically performed by a medical imaging expert analyst. Additionally, a high-end GPU card was required for training the network and such a card may not be available on a standard workstation. However, once the network is trained, it can be deployed on a low-end consumer GPU. Although there is a large overhead in training time (48 hours), processing time is only 20 seconds per image. Tajbakhsh et al. [47] analyzed how well networks trained on natural images transferred to medical images and found pretraining resulted in improved or equal performance compared to random initialization. No transfer learning was used in this study due to limited availability of pretrained weights for 3D architectures. This is an area for potential further development. The proposed method is designed exclusively for fissure detection and does not provide a complete lobar segmentation.

However, the high specificity of our method facilitates lobar segmentation with simple post-processing (i.e. thresholding, morphological operations, and connected component analysis). For challenging cases with incomplete fissures, a more sophisticated surface-fitting technique might be used for postprocessing. For example, optimal surface finding graph search could be used to divide the lung into lobes, defining the graph costs by fissure probabilities.

**CONCLUSION**:

We have proposed a method for automatic detection of pulmonary fissures in CT images using a deep learning framework. We presented a novel coarse-to-fine cascade of ConvNets called FissureNet, and a novel 3D segmentation architecture called Seg3DNet. Fissure detection was evaluated with two rule-based methods (Hessian and DoS) and two learning-based methods (FissureNet and U-Net). The learningbased methods outperformed the rule-based methods. Furthermore, FissureNet outperformed U-Net as it was capable of learning larger-scale global features. FissureNet achieves high sensitivity for fissure detection while producing very few false positives, allowing for straightforward post-processing to obtain a final lobar segmentation. The results show that FissureNet is robust to different CT scanners, scanning protocols (low-dose and normal-dose), inspiration levels (TLC and FRC), imaging modalities (breath-hold vs. 4DCT), and severities of pulmonary disease.

**REFERENCES:**

[1] I. R. S. Valente, P. C. Cortez, E. C. Neto, J. M. Soares, V. H. C. de Albuquerque, and J. M. R. Tavares, “Automatic 3D pulmonary nodule detection in CT images: A survey,” Computer methods and programs in biomedicine, vol. 124, pp. 91–107, 2016.

[2] L. Sorensen, S. B. Shaker, and M. de Bruijne, “Quantitative analysis of pulmonary emphysema using local binary patterns,” IEEE Transactions on Medical Imaging, vol. 29, no. 2, pp. 559–569, 2010.

[3] M. Remy-Jardin, J. Remy, F. Deschildre, D. Artaud, J. P. Beregi, C. Hossein-Foucher, X. Marchandise, and A. Duhamel, “Diagnosis of pulmonary embolism with spiral CT: Comparison with pulmonary angiography and scintigraphy.” Radiology, vol. 200, no. 3, pp. 699–706, 1996.

[4] H. O. Coxson, “Quantitative computed tomography assessment of airway wall dimensions: Current status and potential applications for phenotyping chronic obstructive pulmonary disease,” Proceedings of the American Thoracic Society, vol. 5, no. 9, pp. 940–945, 2008.

[5] J. M. Reinhardt, K. Ding, K. Cao, G. E. Christensen, E. A. Hoffman, and S. V. Bodas, “Registration-based estimates of local lung tissue expansion compared to xenon-CT measures of specific ventilation,” Med. Imag. Analysis, vol. 12, no. 6, pp. 752–763, Dec. 2008.

[6] A. Boueiz, Y. Chang, M. H. Cho, G. R. Washko, R. S. J. Estepar, R. P. Bowler, J. D. Crapo, D. L. DeMeo, J. G. ´ Dy, E. K. Silverman, P. J. Castaldi, and COPDGene Investigators, “Lobar emphysema distribution is associated with 5-year radiological disease progression,” Chest, 2017.

[7] National Emphysema Treatment Trial Research Group, “Patients at high risk of death after lung-volume– reduction surgery,” N Engl J Med, vol. 2001, no. 345, pp. 1075–1083, 2001.

[8] Y. Pan, G. E. Christensen, O. C. Durumeric, S. E. Gerard, J. M. Reinhardt, and G. D. Hugo, “Current- and varifoldbased registration of lung vessel and airway trees,” in 7 th International Workshop on Biomedical Image Registration, June 2016.

[9] T. D. Koster and D. J. Slebos, “The fissure: Interlobar collateral ventilation and implications for endoscopic therapy in emphysema,” Int J Chron Obstruct Pulmon Dis, vol. 11, pp. 765–773, 2016.

[10] T. D. Koster, E. M. van Rikxoort, R. H. Huebner, F. Doellinger, K. Klooster, J. P. Charbonnier, S. Radhakrishnan, F. J. Herth, and D. J. Slebos, “Predicting Lung Volume Reduction after Endobronchial Valve Therapy Is Maximized Using a Combination of Diagnostic Tools,” Respiration, vol. 92, no. 3, pp. 150–157, 2016.

[11] R. Eberhardt, D. Gompelmann, F. J. Herth, and M. Schuhmann, “Endoscopic bronchial valve treatment: Patient selection and special considerations,” Int J Chron Obstruct Pulmon Dis, vol. 10, pp. 2147–2157, 2015.

[12] D. Gompelmann, M. Schuhmann, K. Kontogianni, C. Heussel, J. Tschirren, R. Eberhardt, and F. J. F. Herth, “Fissure integrity and emphysema distribution in patients with severe chronic obstructive pulmonary disease,” Amer. J. Respiratory and Critical Care Medicine, p. A4985, 2017, (abstract).

[13] J. Pu, Z. Wang, S. Gu, C. Fuhrman, J. K. Leader, X. Meng, J. Tedrow, and F. C. Sciurba, “Pulmonary fissure integrity and collateral ventilation in COPD patients,” PLoS ONE, vol. 9, no. 5, p. e96631, 2014.

[14] L. Zhang, E. Hoffman, and J. M. Reinhardt, “Lung lobe segmentation in volumetric X-ray CT images,” IEEE Trans. Medical Imaging, vol. 25, no. 1, pp. 1–16, 2006.

[15] E. M. van Rikxoort, M. Prokop, B. de Hoop, M. A. Viergever, J. P. Pluim, and B. van Ginneken, “Automatic segmentation of pulmonary lobes robust against incomplete fissures,” IEEE Transactions on Medical Imaging, vol. 29, no. 6, pp. 1286–1296, 2010.

[16] J.-M. Kuhnigk, V. Dicken, S. Zidowitz, L. Bornemann, B. Kuemmerlen, S. Krass, H.-O. Peitgen, S. Yuval, H.- H. Jend, W. S. Rau et al., “New tools for computer assistance in thoracic CT. Part 1. Functional analysis of lungs, lung lobes, and bronchopulmonary segments,” Radiographics, vol. 25, no. 2, pp. 525–536, 2005.

[17] X. Zhou, T. Hayashi, T. Hara, H. Fujita, R. Yokoyama, T. Kiryu, and H. Hoshi, “Automatic segmentation and recognition of anatomical lung structures from highresolution chest CT images,” Computerized Medical Imaging and Graphics, vol. 30, no. 5, pp. 299–313, 2006.

[18] S. Ukil and J. M. Reinhardt, “Anatomy-guided lung lobar surface detection in X-ray CT images,” IEEE Trans. Medical Imaging, vol. 28, no. 2, pp. 202–214, 2009, pMID: 19188109.

[19] B. Lassen, E. M. van Rikxoort, M. Schmidt, S. Kerkstra, B. van Ginneken, and J.-M. Kuhnigk, “Automatic segmentation of the pulmonary lobes from chest CT scans based on fissures, vessels, and bronchi,” IEEE Transactions on Medical Imaging, vol. 32, no. 2, pp. 210–222, 2013.

[20] J. Pu, B. Zheng, J. K. Leader, C. Fuhrman, F. Knollmann, A. Klym, and D. Gur, “Pulmonary lobe segmentation in CT examinations using implicit surface fitting,” IEEE Transactions on Medical Imaging, vol. 28, no. 12, pp. 1986–1996, 2009.

[21] T. Doel, D. J. Gavaghan, and V. Grau, “Review of automatic pulmonary lobe segmentation methods from CT,” Computerized Medical Imaging and Graphics, vol. 40, pp. 13–29, 2015.

[22] A. F. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever, “Multiscale vessel enhancement filtering,” in International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, 1998, pp. 130–137.

[23] Y. Sato, S. Nakajima, N. Shiraga, H. Atsumi, S. Yoshida, T. Koller, G. Gerig, and R. Kikinis, “Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images,” Medical Image Analysis, vol. 2, no. 2, pp. 143–168, 1998.

[24] Q. Li, S. Sone, and K. Doi, “Selective enhancement filters for nodules, vessels, and airway walls in two-and threedimensional CT scans,” Medical Physics, vol. 30, no. 8, pp. 2040–2051, 2003.

[25] R. Wiemker, T. Bulow, and T. Blaffert, “Unsupervised ¨ extraction of the pulmonary interlobar fissures from high resolution thoracic CT data,” in International Congress Series, vol. 1281. Elsevier, 2005, pp. 1121–1126.

[26] A. M. Lopez, D. Lloret, J. Serrat, and J. J. Villanueva, ´ “Multilocal creaseness based on the level-set extrinsic curvature,” Computer Vision and Image Understanding, vol. 77, no. 2, pp. 111–144, 2000.

[27] T. Klinder, H. Wendland, and R. Wiemker, “Lobar fissure detection using line enhancing filters.” in Medical Imaging: Image Processing, 2013, p. 86693C.

[28] C. Xiao, M. Staring, J. Wang, D. P. Shamonin, and B. C. Stoel, “A derivative of stick filter for pulmonary fissure detection in CT images,” in SPIE Medical Imaging. International Society for Optics and Photonics, 2013, pp. 86 690V–86 690V.

[29] C. Xiao, B. C. Stoel, M. E. Bakker, Y. Peng, J. Stolk, and M. Staring, “Pulmonary fissure detection in CT images using a derivative of stick filter,” IEEE Transactions on Medical Imaging, vol. 35, no. 6, pp. 1488–1500, 2016.

[30] E. M. van Rikxoort, B. van Ginneken, M. Klik, and M. Prokop, “Supervised enhancement filters: Application to fissure detection in chest CT scans,” IEEE Transactions on Medical Imaging, vol. 27, no. 1, pp. 1–10, 2008.

[31] Q. Wei, Y. Hu, J. H. MacGregor, and G. Gelfand, “Automatic recognition of major fissures in human lungs,” International Journal of Computer Assisted Radiology and Surgery, vol. 7, no. 1, pp. 111–123, 2012.

[32] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, 2015, pp. 234–241.

[33] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A deep convolutional encoder-decoder architecture for image segmentation,” arXiv preprint arXiv:1511.00561, 2015.

[34] E. Shelhamer, J. Long, and T. Darrell, “Fully convolutional networks for semantic segmentation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 4, pp. 640–651, 2017.

[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” Nature, vol. 323, no. 6088, p. 533, 1986.

[36] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1520–1528.

[37] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in International Conference on Machine Learning, 2015, pp. 448–456.

[38] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 807–814.

[39] Theano Development Team, “Theano: A Python framework for fast computation of mathematical expressions,” arXiv e-prints, vol. abs/1605.02688, May 2016. [Online]. Available: <http://arxiv.org/abs/1605.02688>

[40] S. Dieleman, J. Schlter, C. Raffel, E. Olson, S. K. Snderby, D. Nouri et al., “Lasagne: First release.” Aug. 2015. [Online]. Available: <http://dx.doi.org/10.5281/zenodo.27878>

[41] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[42] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.

[43] E. A. Regan, J. E. Hokanson, J. R. Murphy, B. Make, D. A. Lynch, T. H. Beaty, D. Curran-Everett, E. K. Silverman, and J. D. Crapo, “Genetic epidemiology of COPD (COPDGene) study design,” COPD: Journal of Chronic Obstructive Pulmonary Disease, vol. 7, no. 1, pp. 32–43, 2011.

[44] A. S. Buist, W. M. Vollmer, and M. A. McBurnie, “Worldwide burden of COPD in high- and low-income countries. Part I. The burden of obstructive lung disease (BOLD) initiative,” Int. J. Tuberc. Lung Dis., vol. 12, no. 7, pp. 703–708, Jul 2008.

[45] E. S. Wan, P. J. Castaldi, M. H. Cho, J. E. Hokanson, E. A. Regan, B. J. Make, T. H. Beaty, M. K. Han, J. L. Curtis, D. Curran-Everett, D. A. Lynch, D. L. DeMeo, J. D. Crapo, and E. K. Silverman, “Epidemiology, genetics, and subtyping of preserved ratio impaired spirometry (PRISm) in COPDGene,” Respiratory research, vol. 15, no. 1, p. 89, 2014.

[46] T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets,” PloS one, vol. 10, no. 3, p. e0118432, 2015.

[47] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, “Convolutional neural networks for medical image analysis: Full training or fine tuning?” IEEE Transactions on Medical Imaging, vol. 35, no. 5, pp. 1299–1312, 2016.