Graph Refinement based Airway Extraction

Using Mean-field and Graph Neural Networks

Raghavendra Selvan
raghav@di.ku.dk
Department of Computer Science

Outline

1. Airway Diseases
2. Understanding the Data
3. Methods
   - Existing Methods
   - Graph Refinement Model
   - Mean-Field Networks
   - Graph Neural Networks
4. Experiments

Outline

COPD: Leading Factor of Morbidity & Mortality

- Tobacco Smoking
- Indoor & Outdoor Air pollution

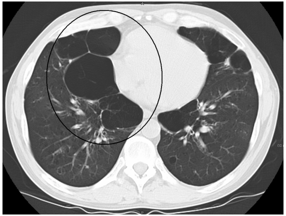
COPD: Leading Factor of Morbidity & Mortality

- Tobacco Smoking
- Indoor & Outdoor Air pollution
- Third leading cause of death by 2030
- 174.5M affected; 3.2M deaths (2015)

COPD: Leading Factor of Morbidity & Mortality

- Tobacco Smoking
- Indoor & Outdoor Air pollution
- Third leading cause of death by 2030
- 174.5M affected; 3.2M deaths (2015)
- Destruction of lung tissue (Emphysema)
- Change of airway morphology (Bronchiectasis)

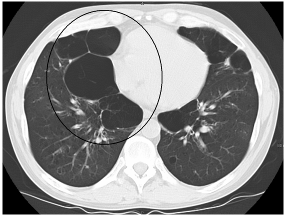
COPD: Leading Factor of Morbidity & Mortality

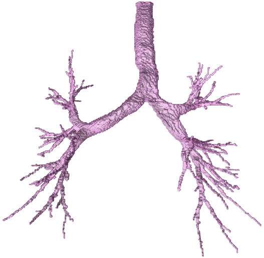


- Tobacco Smoking
- Indoor & Outdoor Air pollution
- Third leading cause of death by 2030
- 174.5M affected; 3.2M deaths (2015)
- Destruction of lung tissue (Emphysema)
- Change of airway morphology (Bronchiectasis)

COPD: Leading Factor of Morbidity & Mortality

- Tobacco Smoking
- Indoor & Outdoor Air pollution
- Third leading cause of death by 2030
- 174.5M affected; 3.2M deaths (2015)
- Destruction of lung tissue (Emphysema)
- Change of airway morphology (Bronchiectasis)



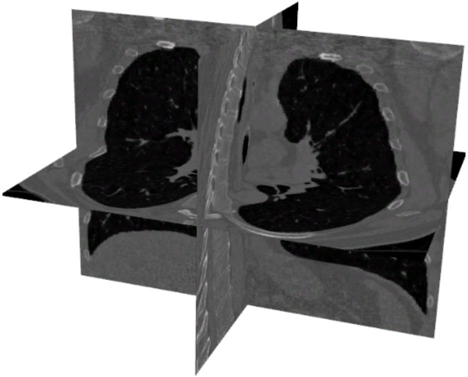


Existing Diagnostics are Rudimentary & Tedious

- Lung Function Tests
  - \+ Simple and inexpensive
  - – Patient dependent
  - – Low reproducibility
  - – Mild cases go unnoticed

Existing Diagnostics are Rudimentary & Tedious

- Lung Function Tests
  + Simple and inexpensive
  – Patient dependent
  – Low reproducibility
  – Mild cases go unnoticed
- 3D CT Scans
  + Provide more information
  – Arduous to read the data; even for experts
  – Low inter-observer agreement



Objective

Automatic Airway Segmentation and obtain **useful** COPD biomarkers

Outline

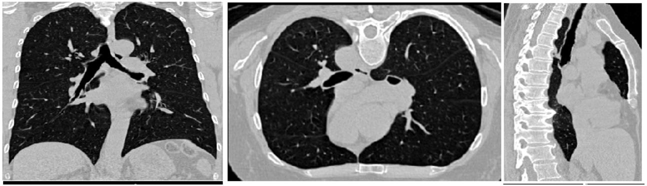
1. Airway Diseases
2. Understanding the Data
3. Methods
   - Existing Methods
   - Graph Refinement Model
   - Mean-Field Networks
   - Graph Neural Networks
4. Experiments

Primary Data from Danish Lung Cancer Study

- Danish Lung Cancer Screening Trial [1]
- Low-dose CT
- $> 10,000$ scans
- Age 50-70 years.
- Smoker or former smoker ($>$ 20 pack years)
- 32/10,000 have segmentations verified by an expert user!

[1] Pedersen, J. H., et.al : The Danish randomized lung cancer CT screening trialoverall design and results of the prevalence round. Journal of Thoracic Oncology, (2009)

CT Images are noisy, low contrast & low-res.



- Volume resolution $\sim$ 300x250x275
- Voxels $\sim$ 0.75mmx0.75mmx1mm
- Challenges
  - Acquisition noise
  - Inter-patient variability
  - Several "interfering" structures
  - Labels/Annotations

Outline

1. Airway Diseases
2. Understanding the Data
3. Methods
   - Existing Methods
   - Graph Refinement Model
   - Mean-Field Networks
   - Graph Neural Networks
4. Experiments

Most Existing Methods handle occlusions poorly

- State-of-the-art: Region-growing (!) based methods
- EXACT Study[2] compares 15 methods; No clear winner
- Small airways are challenging
- Challenging to overcome occlusions

[2] Lo, P., et.al : Extraction of airways from CT (EXACT'09). IEEE Transactions on Medical Imaging, (2012)

Graph Refinement Model for Airway Extraction

Graph Refinement Model for Airway Extraction

Desired Properties

- Exploratory (to overcome occlusions)
- Detect small airways
- Uncertainty estimates

Preprocess Image to Graph Model

One possibility ....

Preprocess Image to Graph Model

One possibility ....

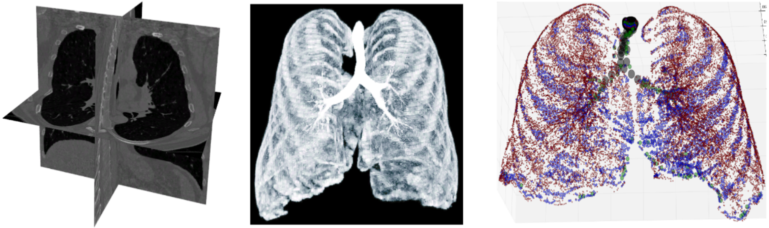


Figure 1: Visualisation of the pre-processing carried out to transform the input image (left) into a probability image (center) and then into graph format (right). Nodes in the graph are shown in scale to capture the variations in their local radius.

Graph Refinement Model

- Input graph: $\mathcal{G}_i : \{\mathcal{N}, \mathcal{E}_i\}$
- Node features: $\mathbf{X} \in \mathbb{R}^{F \times N}$
- Input adjacency: $\mathbf{A}_i \in \{0, 1\}^{N \times N}$

Graph Refinement Model

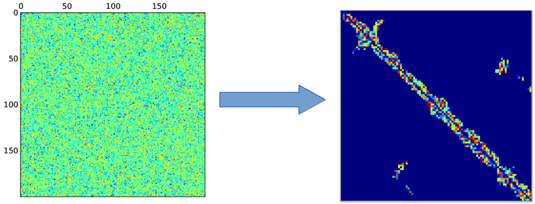
- Input graph: $\mathcal{G}_i : \{\mathcal{N}, \mathcal{E}_i\}$
- Node features: $\mathbf{X} \in \mathbb{R}^{F \times N}$
- Input adjacency: $\mathbf{A}_i \in \{0,1\}^{N \times N}$

Graph Refinement Task

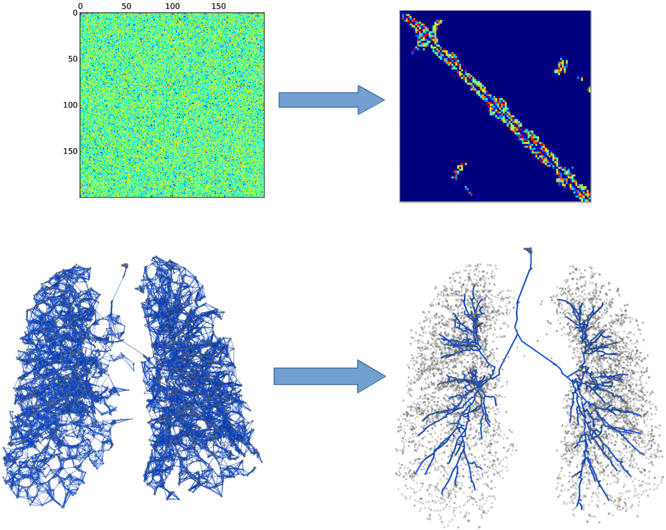
$$f(\mathcal{G}_i) \rightarrow \mathcal{G}$$

Output subgraph $\mathcal{G}$ with $\mathcal{E} \subset \mathcal{E}_i$; $\mathbf{A} \in \{0,1\}^{N \times N}$

Visualise Graph Refinement Task



Visualise Graph Refinement Task



Updating Probabilistic Graphical Model

- Binary random variable to capture existence of edge between nodes
- $s_{ij} \in \{0, 1\}$ with prob. $\alpha_{ij} \in [0, 1]$

Updating Probabilistic Graphical Model

- Binary random variable to capture existence of edge between nodes
- $s_{ij} \in \{0, 1\}$ with prob. $\alpha_{ij} \in [0, 1]$
- For each node: $\mathbf{s}_i = \{s_{ij}\} : j = 1 \dots N$
- Global connectivity variable: $\mathbf{S} = [\mathbf{s}_1 \dots \mathbf{s}_N]$
- Each instance of $\mathbf{S}$ an $N \times N$ adjacency matrix

Updating Probabilistic Graphical Model

- Binary random variable to capture existence of edge between nodes
- $s_{ij} \in \{0, 1\}$ with prob. $\alpha_{ij} \in [0, 1]$
- For each node: $\mathbf{s}_i = \{s_{ij}\} : j = 1 \dots N$
- Global connectivity variable: $\mathbf{S} = [\mathbf{s}_1 \dots \mathbf{s}_N]$
- Each instance of $\mathbf{S}$ an $N \times N$ adjacency matrix

Input Graph $\mathcal{G}_i$ is completely described by $\mathbf{X}, \mathbf{A}_i$,

Updating Probabilistic Graphical Model

- Binary random variable to capture existence of edge between nodes
- $s_{ij} \in \{0, 1\}$ with prob. $\alpha_{ij} \in [0, 1]$
- For each node: $\mathbf{s}_i = \{s_{ij}\} : j = 1 \dots N$
- Global connectivity variable: $\mathbf{S} = [\mathbf{s}_1 \dots \mathbf{s}_N]$
- Each instance of $\mathbf{S}$ an $N \times N$ adjacency matrix

Input Graph $\mathcal{G}_i$ is completely described by $\mathbf{X}, \mathbf{A}_i$,

Posterior density of interest: $p(\mathbf{S}|\mathbf{X}, \mathbf{A}_i)$

Updating Probabilistic Graphical Model

- Binary random variable to capture existence of edge between nodes
- $s_{ij} \in \{0, 1\}$ with prob. $\alpha_{ij} \in [0, 1]$
- For each node: $\mathbf{s}_i = \{s_{ij}\} : j = 1 \dots N$
- Global connectivity variable: $\mathbf{S} = [\mathbf{s}_1 \dots \mathbf{s}_N]$
- Each instance of $\mathbf{S}$ an $N \times N$ adjacency matrix

Input Graph $\mathcal{G}_i$ is completely described by $\mathbf{X}, \mathbf{A}_i$,

Posterior density of interest: $p(\mathbf{S}|\mathbf{X}, \mathbf{A}_i)$

$$\begin{aligned}\ln p(\mathbf{S}|\mathbf{X}) &\propto \ln p(\mathbf{S}, \mathbf{X}) \\ &= -\ln Z + \sum_{i \in \mathcal{N}} \phi_i(\mathbf{s}_i) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}(\mathbf{s}_i, \mathbf{s}_j),\end{aligned}$$

Variational Approximate Inference

Variational Approximate Inference

$p(\mathbf{S}|\mathbf{X}, \mathbf{A}_i)$ is intractable except for in trivial cases.

Variational Approximate Inference

$p(\mathbf{S}|\mathbf{X}, \mathbf{A}_i)$ is intractable except for in trivial cases.

$$p(\mathbf{S}|\mathbf{X}, \mathbf{A}_i) \approx q(\mathbf{S})$$

Variational Approximate Inference

$p(\mathbf{S}|\mathbf{X}, \mathbf{A}_i)$ is intractable except for in trivial cases.

$$p(\mathbf{S}|\mathbf{X}, \mathbf{A}_i) \approx q(\mathbf{S})$$

Variational Approximate Inference

Minimize ELBO to obtain $q(\mathbf{S}) \in \mathcal{Q}$

$$\mathcal{F}(q_{\mathbf{S}}) = \ln Z + \mathbb{E}_{q_{\mathbf{S}}}\Big[\ln p(\mathbf{S}|\mathbf{X}) - \ln q(\mathbf{S})\Big]$$

Simpler approximation using MFA

Simpler approximation using MFA

Factorisable $q(\mathbf{S})$

$$q(\mathbf{S}) = \prod_{i=1}^{N}\prod_{j=1}^{N} q_{ij}(s_{ij}),$$

$$\text{where, } q_{ij}(s_{ij}) = \begin{cases} \alpha_{ij} & \text{if } s_{ij} = 1 \\ (1-\alpha_{ij}) & \text{if } s_{ij} = 0 \end{cases},$$

Simpler approximation using MFA

Factorisable $q(\mathbf{S})$

$$q(\mathbf{S}) = \prod_{i=1}^{N} \prod_{j=1}^{N} q_{ij}(s_{ij}),$$

$$\text{where, } q_{ij}(s_{ij}) = \begin{cases} \alpha_{ij} & \text{if } s_{ij} = 1 \\ (1 - \alpha_{ij}) & \text{if } s_{ij} = 0 \end{cases},$$

Assumes edges to be independent.

Node and Pairwise Potentials for MFA

Node and Pairwise Potentials for MFA

Node Potential

For each node $i \in \mathcal{N}$,

$$\phi_i(\mathbf{s}_i) = \sum_{v=0}^{D} \beta_v \mathbb{I}\Big[\sum_j s_{ij} = v\Big] + \mathbf{a}^T \mathbf{x}_i \sum_j s_{ij},$$

Node and Pairwise Potentials for MFA

Node Potential

For each node $i \in \mathcal{N}$,

$$\phi_i(\mathbf{s}_i) = \sum_{v=0}^{D} \beta_v \mathbb{I}\Big[\sum_j s_{ij} = v\Big] + \mathbf{a}^T \mathbf{x}_i \sum_j s_{ij},$$

Pairwise Potential

For each edge, $(i,j) \in \mathcal{E}_i$

$$\phi_{ij}(\mathbf{s}_i, \mathbf{s}_j) = \lambda\big(1 - 2|s_{ij} - s_{ji}|\big) + (2s_{ij}s_{ji} - 1)\Big[\boldsymbol{\eta}^T|\mathbf{x}_i - \mathbf{x}_j| + \boldsymbol{\nu}^T(\mathbf{x}_i\mathbf{x}_j)\Big].$$

Parameters $= [\boldsymbol{\beta}, \mathbf{a}, \lambda, \boldsymbol{\eta}, \boldsymbol{\nu}]$

Minimize ELBO to get MFA Iterations

MFA Iterations

$$\alpha_{kl}^{(t+1)} = \sigma(\gamma_{kl}) = \frac{1}{1 + \exp^{-\gamma_{kl}}} \;\forall\; k = \{1 \dots N\},\; l \in \mathcal{N}_k,$$

where $\sigma(\cdot)$ is sigmoid activation, $\mathcal{N}_k$ are neighbours of node k, and

Minimize ELBO to get MFA Iterations

MFA Iterations

$$\alpha_{kl}^{(t+1)} = \sigma(\gamma_{kl}) = \frac{1}{1+\exp^{-\gamma_{kl}}} \; \forall \; k = \{1 \dots N\}, \; l \in \mathcal{N}_k,$$

where $\sigma(\cdot)$ is sigmoid activation, $\mathcal{N}_k$ are neighbours of node k, and

$$\begin{aligned}\gamma_{kl} = & \prod_{j \in \mathcal{N}_k \setminus l} (1-\alpha_{kj}^{(t)}) \Bigg\{ \sum_{m \in \mathcal{N}_k \setminus l} \frac{\alpha_{km}^{(t)}}{(1-\alpha_{km}^{(t)})} \Big[(\beta_2 - \beta_1) \\ & - \beta_2 \sum_{n \in \mathcal{N}_k \setminus l,m} \frac{\alpha_{kn}^{(t)}}{(1-\alpha_{kn}^{(t)})} \Big] + (\beta_1 - \beta_0) \Bigg\} + \mathbf{a}^T \mathbf{x}_k \\ & + (4\alpha_{lk}^{(t)} - 2)\lambda + 2\alpha_{lk}^{(t)} (\boldsymbol{\eta}^T |\mathbf{x}_k - \mathbf{x}_l| + \boldsymbol{\nu}^T (\mathbf{x}_k \mathbf{x}_l)).\end{aligned} \tag{1}$$

MFA to Mean-Field Networks

MFA to Mean-Field Networks

- MFA Iterations resemble feed-forward operations in neural network

$$\alpha_{kl}^{(t+1)} = \sigma(\gamma_{kl}) = \frac{1}{1 + \exp^{-\gamma_{kl}}} \; \forall \; k = \{1 \dots N\}, \; l \in \mathcal{N}_k,$$

$\boldsymbol{\alpha}$ is soft prediction of global connectivity variable

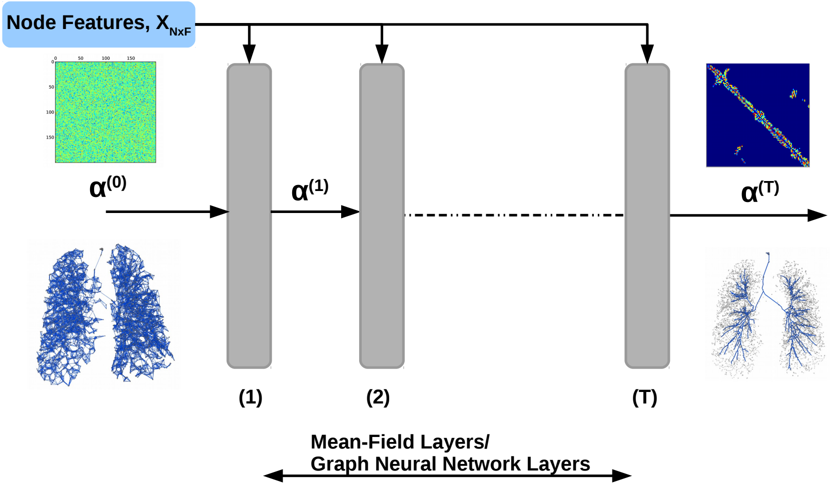
MFA to Mean-Field Networks

- MFA Iterations resemble feed-forward operations in neural network

$$\alpha_{kl}^{(t+1)} = \sigma(\gamma_{kl}) = \frac{1}{1+\exp^{-\gamma_{kl}}} \ \forall\ k = \{1 \dots N\},\ l \in \mathcal{N}_k,$$

  $\boldsymbol{\alpha}$ is soft prediction of global connectivity variable
- $T-$iterations as a $T-$layered network
- Automatic differentiation to learn parameters: $\mathcal{L}(\boldsymbol{\alpha}, \mathbf{A}_r)$, where $\mathbf{A}_r$ is reference adjacency.

Summarising MFN



Summarising MFN

- Yields approximation to underlying posterior
- Simple factors
- Handful of parameters
- Easy to optimise
- – Hand-crafting potentials is cumbersome
- – Might not generalise across applications

Graph Neural Networks

Graph Neural Networks

- Neural networks operating directly on graph structured data

Graph Neural Networks

- Neural networks operating directly on graph structured data
- Generalisation of message passing algorithms

Graph Neural Networks

- Neural networks operating directly on graph structured data
- Generalisation of message passing algorithms
- Arbitrarily complex messages

Graph Neural Networks

- Neural networks operating directly on graph structured data
- Generalisation of message passing algorithms
- Arbitrarily complex messages
- End-to-end trainable inference systems

Graph Neural Networks

- Neural networks operating directly on graph structured data
- Generalisation of message passing algorithms
- Arbitrarily complex messages
- End-to-end trainable inference systems

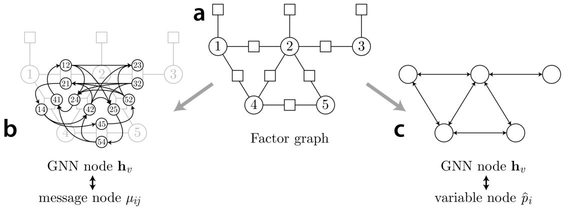


Figure: Two mappings of Factor Graphs into GNNs

Figure from Yoon et al. "Inference in probabilistic graphical models by Graph Neural Networks" (2018)

Graph Auto Encoder (GAE) for Graph Refinement

Graph Auto Encoder (GAE) for Graph Refinement

- Encoder comprises of GNNs; Message passing between nodes

Graph Auto Encoder (GAE) for Graph Refinement

- Encoder comprises of GNNs; Message passing between nodes
- Jointly train to learn useful embeddings

Graph Auto Encoder (GAE) for Graph Refinement

- Encoder comprises of GNNs; Message passing between nodes
- Jointly train to learn useful embeddings
- Suitable for graph refinement: $f(\mathcal{G}_i) \rightarrow \mathcal{G}$

Graph Auto Encoder (GAE) for Graph Refinement

- Encoder comprises of GNNs; Message passing between nodes
- Jointly train to learn useful embeddings
- Suitable for graph refinement: $f(\mathcal{G}_i) \rightarrow \mathcal{G}$

GAE Model for Graph Refinement: Encoder

GAE Model for Graph Refinement: Encoder

Encoder:

$$
\begin{aligned}
\text{Node Embedding:} \quad & \mathbf{h}_j^1 = & g_n(\mathbf{x}_j) \\
\text{Node-to-Edge mapping:} \quad & \mathbf{h}_{(i,j)}^1 = & g_{n2e}([\mathbf{h}_i^1, \mathbf{h}_j^1]) \\
\text{Edge-to-Node mapping:} \quad & \mathbf{h}_j^2 = & g_{e2n}(\sum_i^{N_j} \mathbf{h}_{(i,j)}^1]) \\
\text{Node-to-Edge mapping:} \quad & \mathbf{h}_{(i,j)}^2 = & g_{n2e}([\mathbf{h}_i^2, \mathbf{h}_j^2])
\end{aligned}
$$

GAE Model for Graph Refinement: Encoder

Encoder:

$$
\begin{aligned}
\text{Node Embedding:} \quad & \mathbf{h}_j^1 = && g_n(\mathbf{x}_j) \\
\text{Node-to-Edge mapping:} \quad & \mathbf{h}_{(i,j)}^1 = && g_{n2e}([\mathbf{h}_i^1, \mathbf{h}_j^1]) \\
\text{Edge-to-Node mapping:} \quad & \mathbf{h}_j^2 = && g_{e2n}(\sum_i^{N_j} \mathbf{h}_{(i,j)}^1]) \\
\text{Node-to-Edge mapping:} \quad & \mathbf{h}_{(i,j)}^2 = && g_{n2e}([\mathbf{h}_i^2, \mathbf{h}_j^2])
\end{aligned}
$$

$g_{\dots}(\cdot)$ are 2-layered MLPs, with ReLU, dropout and layer normalisation

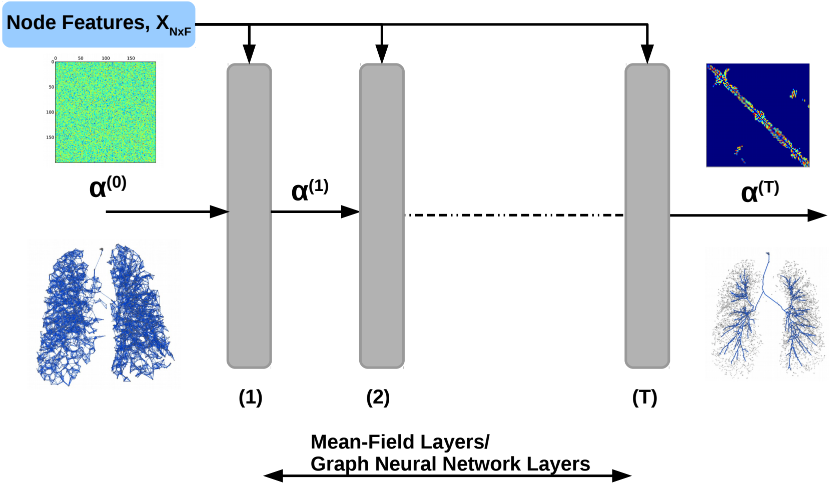
GAE Model for Graph Refinement: Decoder

Decoder:

$$\alpha_{ij} = \sigma(g_{dec}(\mathbf{h}^2_{(i,j)})) \tag{2}$$

g_{dec} is a 1D convolutional layer with one output channel
Model can be trained by computing the loss $\mathcal{L}\boldsymbol{\alpha}, \mathbf{A}_r$

Summarising GNNs



Outline

Training both models: Dice Loss

To tackle Class Imbalance:

Training both models: Dice Loss

To tackle Class Imbalance:

$$\mathcal{L}(\boldsymbol{\alpha}, \mathbf{A}_r) = 1 - \frac{2\sum_{i,j=1}^{N} \alpha_{ij} A_{ij}}{\sum_{i,j=1}^{N} \alpha_{ij}^2 + \sum_{i,j=1}^{N} A_{ij}^2},$$

A_{ij} are individual binary entries $\mathbf{A}_r$

Data

- Danish Lung Cancer Screening Trial
- Low-dose Chest CT scans
- 32 scans with “Reference” annotations
- 100 scans with automatic segmentation

Results

- Error Measure based on centerline distances
- Average of two distances, $d_{err} = (d_{FP} + d_{FN})/2$
- Compared with Top Performer on Airway Extraction Challenge

Table I: Performance comparison

Method	d_{FP}(mm)	d_{FN}(mm)	d_{err} (mm)
Voxel Classifier	3.871	5.108	4.489 ± 0.754
MFN	3.716	3.992	3.845 ± 0.415
GNN	3.513	2.890	3.202 ± 0.386