Linking convolutional neural networks with graph convolutional networks: application in pulmonary artery-vein separation

Zhiwei Zhai[1], Marius Staring[1], Xuhui Zhou[3], Qiuxia Xie[3], Xiaojuan Xiao[3], M. Els Bakker[1], Lucia J. Kroft[1], Boudewijn P.F. Lelieveldt[1], Gudula J.A.M. Boon[2], Frederikus A. Klok[2], and Berend C. Stoel[1]

[1] Department of Radiology,
[2] Department of Thrombosis and Hemostasis,
Leiden University Medical Center, Leiden, The Netherlands
[3] Department of Radiology, Sun Yat-sen University, Shenzhen, China
`Z.Zhai@lumc.nl`

Abstract. Graph Convolutional Networks (GCNs) are a novel and powerful method for dealing with non-Euclidean data, while Convolutional Neural Networks (CNNs) can learn features from Euclidean data such as images. In this work, we propose a novel method to combine CNNs with GCNs (CNN-GCN), that can consider both Euclidean and non-Euclidean features and can be trained end-to-end. We applied this method to separate the pulmonary vascular trees into arteries and veins (A/V). Chest CT scans were pre-processed by vessel segmentation and skeletonization, from which a graph was constructed: voxels on the skeletons resulting in a vertex set and their connections in an adjacency matrix. 3D patches centered around each vertex were extracted from the CT scans, oriented perpendicularly to the vessel. The proposed CNN-GCN classifier was trained and applied on the constructed vessel graphs, where each node is then labeled as artery or vein. The proposed method was trained and validated on data from one hospital (11 patient, 22 lungs), and tested on independent data from a different hospital (10 patients, 10 lungs). A baseline CNN method and human observer performance were used for comparison. The CNN-GCN method obtained a median accuracy of 0.773 (0.738) in the validation (test) set, compared to a median accuracy of 0.817 by the observers, and 0.727 (0.693) by the CNN. In conclusion, the proposed CNN-GCN method combines local image information with graph connectivity information, improving pulmonary A/V separation over a baseline CNN method, approaching the performance of human observers.

1 Introduction

Graph Convolutional Networks (GCN) are a variant of Convolutional Neural Networks (CNN) applied on graphs [1,2]. Recently, there are many research fields processing graphs (non-Euclidean data), such as social networks, citation networks, applied chemistry, computer vision, anatomical structures etc. GCNs and their variants have obtained state-of-the-art performance in these fields [1,3].

In the medical imaging domain, GCNs achieved promising results as well. Parisot et al. [4] proposed a semi-supervised method for disease prediction using GCNs, where individuals were represented as nodes; the features consisted of both image and non-image information; a sparse graph was constructed among all individuals and partially labeled. The GCN was trained on the labeled nodes and inferred classes of unlabeled nodes, based on node features and their connections. Shin et al. [5] proposed a deep vessel segmentation by combining CNNs and GCNs, where a CNN was trained for generating features and vessel probabilities; a GCN was trained to predict the presence of a vessel, based on the features and connectivity, and an inference module to generate the final segmentation. The method achieved competitive results in both retinal vessel and coronary artery data sets. However, this classifier cannot be trained end-to-end, which may yield sub-optimal results. In this work, we propose therefore a network linking CNNs with GCNs, which can be trained end-to-end.

Separation of pulmonary arteries/veins (A/V) is a challenging problem, because of the complexity of their anatomical structures and the similarity in intensity and morphology. In recent years, only a few methods have been developed for separating A/V, including traditional methods [6,7] and deep learning based methods [8]. Both local and global information were considered by the traditional methods [6,7], including the property of parallel configuration and close proximity of arteries and bronchus, anatomical information on the A/V roots, and connectivity information. Nardelli et al. [8] proposed a CNN method for classifying vessel particles to A/V based on local patches and subsequently applied a graph-cut optimization to refine the classifications, which combined connectivity information and predictions by the CNN classifier.

In this work, we propose a novel network linking CNNs and GCNs (CNN-GCN), which considers both local image and graph connectivity features and the classifier can be trained end-to-end. To enable large graphs with huge amounts of nodes containing features from 3D patches to fit in GPU memory, we propose a batch-based strategy on graphs for CNN-GCN training and validation, instead of using entire graphs. The CNN-GCN method was applied to separate pulmonary A/V. These images were pre-processed by vessel graph construction and 3D patch extraction, and subsequently the classes of each node in the vessel graph was predicted by the CNN-GCN classifier, and A/V volume was re-built based on the classification of the nodes.

2 Methods

In this section, we provide the theoretical background of GCNs, and motivation for linking CNNs with GCNs. We demonstrate its usage by an application in separating pulmonary arteries-veins.

2.1 Graph convolution networks

We follow the works of Kipf et al. [1] and Wu et al. [2] to introduce the theoretical background of GCNs. A graph is defined as $\mathcal{G} = (\mathcal{V}, A)$, where $\mathcal{V} = \{v_i, i = 1, \ldots, N\}$ is the vertex set consisting of N nodes, and $A = (a_{ij})_{N \times N}$ is the adjacency matrix of the graph: $a_{ij} = 1$ if nodes v_i and v_j are connected, otherwise $a_{ij} = 0$. For consistency, each node is always connected to itself, i.e. $a_{ii} = 1$. Each node v_i has an F-dimensional feature vector $x_i \in \mathbb{R}^F$, and the feature matrix for all nodes is $X = [x_1, \ldots, x_N]^T$ i.e. $X \in \mathbb{R}^{N \times F}$. Each node v_i belongs to a class c out of C, which can be encoded as a one-hot vector $y_i \in \{0, 1\}^C$.

In a graph convolution layer, the operations can be divided into three stages: feature representation, feature transform and nonlinear activation. Commonly, the input layer $H^{(0)}$ is the input feature matrix: $H^{(0)} = X$. Within a GCN, the input for the k^{th} layer is the output from the $(k-1)^{th}$ layer: $H^{(k-1)}$ with size $N \times F^{(k-1)}$, and as output $H^{(k)}$ of size $N \times F^{(k)}$. The processing within this layer is expressed as:

$$H^{(k)} = \sigma\left(W H^{(k-1)} \Theta^{(k)}\right), \tag{1}$$

where W is an $N \times N$ weight matrix among nodes $\mathcal{V}$; $\Theta^{(k)}$ is a layer-specific trainable parameter matrix of size $F^{(k-1)} \times F^{(k)}$; $\sigma(\cdot)$ is an activation function, like ReLU. The operation $\hat{H}^{(k)} = W H^{(k-1)}$ is considered to be a 'feature representation', which averages the feature vectors of neighbors around nodes. Based on the spectral graph convolution, Kipf et al. [1] estimated the weight matrix using Chebyshev polynomials: $W = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, with D the diagonal degree matrix of A and $w_{ij} = \frac{a_{ij}}{\sqrt{d_i \cdot d_j}}$, i.e. not trainable. Instead of using a fixed handcrafted weight matrix, Monti et al. [3] proposed a method to calculate W using parametric kernels $w_{ij} = k(x_i, x_j | \theta)$, where θ represents the trainable parameters and $k(\cdot)$ is a Gaussian kernel. In this study we use the trainable method. The operation: $\tilde{H}^{(k)} = \hat{H}^{(k)} \Theta^{(k)}$ corresponds to a 'feature transform', where the feature vector can be transformed from $F^{(k-1)}$ dimensions to $F^{(k)}$ dimensions. If needed, a trainable bias ϵ of size $1 \times F^{(k)}$ can be involved as well after transformation. Finally, a 'nonlinear activation' operation is applied: $H^{(k)} = \sigma\left(\tilde{H}^{(k)}\right)$.

2.2 Linking CNN with GCN

A GCN can combine both local and connectivity information, which may be useful to analyze vascular trees. To combine CNNs with GCNs we let the feature matrix X be learned by a CNN: $x_i = \Phi(p_i | \Theta)$, where $\Phi(\cdot)$ is a general CNN with an arbitrary sequence of layers, and p_i a local image patch with size S. Then, the CNN and GCN can be linked as follows:

$$H = \sigma\left(W_\theta X \Theta\right), \text{ where } X = \Phi\left(P | \Theta\right), \tag{2}$$

Algorithm 1 Model linking CNN with GCN

```
1: procedure CnnGcnModel( $B$, $NB$ )
2:     Inputs of the model
3:     Batch: $B$ with size $b \times S$
4:     Neighborhood Batch: $NB$ with size $b \times n \times S$
5:     $\tilde{X} = \Phi(B|\Theta)$          ▷ $\Phi(\cdot)$ is a shared function with sequential layers
6:     $\tilde{NX} = \Phi(NB|\Theta)$
7:     $H = GcnLayer(\tilde{X}, \tilde{NX})$
8:     $Y = softmax(H)$          ▷ other activation may also be used here
9:     return $Y$
```

Algorithm 2 Graph convolutional layer with batch strategy on graphs

```
1: procedure GcnLayer($X$, $NX$)
2:     Inputs: $X$, $NX$          ▷ $X$ with size $b \times F$; $NX$ with size $b \times n \times F$
3:     $W = k(X, NX|\theta)$          ▷ $k(\cdot)$ is a trainable function.
4:     $\tilde{W} = expend\_dims(W, axis = 1)$     ▷ $W$ with size $b \times n$; $\tilde{W}$ with size $b \times 1 \times n$
5:     $\hat{X} = K.reshape(K.batchdot(\tilde{W}, NX), (b, F))$          ▷ $\hat{X}$ with shape $b \times F$
6:     $\hat{H} = (X + \hat{X})/(1 + sum(W))$     ▷ feature representation and normalization.
7:     $\tilde{H} = \hat{H}\Theta$          ▷ feature transform
8:     $H = Relu(\tilde{H})$          ▷ other activations may be used here
9:     return $H$
```

with $P = [p_1, ..., p_N]^T$ the set of all patches. In other words, X are the learned feature maps from the CNN, which is then followed by a step of a normal GCN akin to Eq. (1). The CNN-GCN classifier is linked as a function chain, which can be straightforwardly optimized by gradient decent using back-propagation. The patch p_i of each node is processed with a shared CNN function $\Phi(\cdot)$. Pseudo-code of the proposed method is given in Algorithm 1.

Straightforward training of the CNN-GCN requires loading the entire graph in GPU memory. As features are learned, local patches of all nodes need to be available as well, i.e. the entire matrix P which has size around $45000 \times S$. Since this is not feasible with current GPUs, we propose a sampling strategy on graphs instead of using entire graphs, similarly to batch processing for training CNNs. Given a graph $\mathcal{G}$, we randomly select b nodes with their patches as an input batch B, with size $b \times S$. Since we use a graph structure, we also need the image patches of these neighbors. This is denoted by NB which has size $b \times n \times S$, where n is the number of neighbors. Both the batch B and the neighborhood batch NB are processed with a shared CNN function $\Phi(\cdot)$, which may consist of multiple layers, such as convolution layers, max-pooling layers, activation layers etc. In every iteration a new selection of B and NB is made. Pseudo-code of a GCN layer with batch strategy on graphs is presented in Algorithm 2.

2.3 Application to pulmonary artery-vein separation

Lung vessel trees were extracted from the chest CT scans by a vessel segmentation method [9] and then skeletonized by a skeletonization method [10] (using

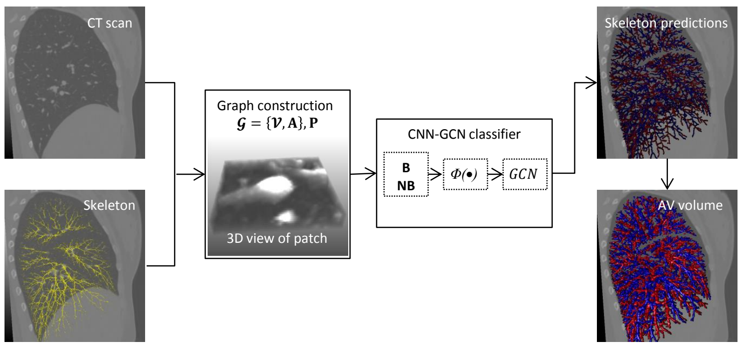

Fig. 1: An overview of proposed method for pulmonary arteries-veins separation.

MevisLab 2.7.1). All voxels on the vessel skeleton were added as a set of nodes $\mathcal{V}$, and the adjacency matrix A was constructed based on their connections. In this study, only one-degree (direct) neighbors were considered. A graph $\mathcal{G}$ was constructed for left and right lung separately. For each voxel on the vessel skeleton, a local patch p_i perpendicular to the vessel orientation and of size $S = [32, 32, 5]$ was extracted from the CT images. A patch p_i is labeled either artery or vein, i.e. $y_i \in \{0, 1\}^2$, based on the label of the center voxel. The CNN architecture $\Phi(\cdot|\Theta)$ for processing patches is adopted from [8]. Following the proposed batch strategy, input batch B and its neighbors NB were processed with the shared function $\Phi(\cdot|\Theta)$ to the feature vectors X and NX, respectively. The feature vectors X and NX were inserted into a GCN layer, represented and transformed to a new dimension. After an activation layer, the output is predicted with 2 dimensions. The architecture for pulmonary artery-vein separation by linking CNN and GCN is demonstrated in Fig. 1. Based on the predictions of center voxels, the A/V volume is re-built, where each voxel on the cross-sectional area is labeled with the prediction of corresponding center-voxel.

3 Experiments

The CNN-GCN method was implemented in 'TensorFlow.Keras', version 1.12.2, where the GCN layer was implemented inheriting from the 'Keras.layers.Layer' class. Categorical cross-entropy was used as loss function and the Stochastic Gradient Descent method was used as optimizer. Training was performed on a local GPU cluster (Nvidia Titan Xp 12 GB). The source code is publicly available via `https://github.com/chushan89/Linking-CNN-GCN.git`.

We collected contrast enhanced CT scans of 11 cases, scanned with a Toshiba Aquilion ONE, from Sun Yat-sen University Hospital (SunYs data set) and contrast enhanced CT scans of 10 cases, scanned with a Toshiba Aquilion 64, from Leiden University Medical Center (LUMC data set). All CT scans were resampled by a cubic B-spline filter to obtain isotropic voxels, with a size of 0.625mm^3.

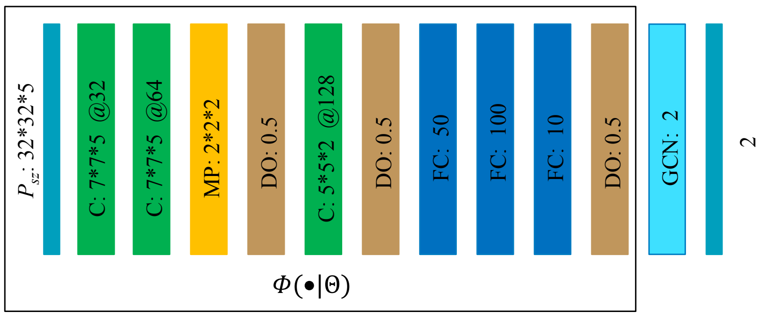

Fig. 2: Architecture of CNN-GCN model for pulmonary arteries-veins separation, by linking CNN and GCN layer. C: convolution layer; MP: max-pooling layer; DO: dropout layer; FC: fully-connected layer (or dense layer); $\Phi(\cdot|\Theta)$: a CNN function with multiple layers; GCN: graph convolutional layer.

Lung vessels were segmented and provided for each case [9]. For the SunYs data set, two radiologists of Sun Yat-sen University Hospital labeled the segmented lung vessels into arteries or veins, as initial annotations. The initial labels were checked and corrected by three experts at the LUMC. The corrected annotations were used as ground truth, where initial annotation was used to assess observer performance. In total 22 lungs from 11 patients (consisting of 1,041,463 patches) with fully annotated pulmonary arteries-veins were obtained, referred to as the SunYs dataset. From this data set, 16 lungs (722,013 patches) were used for training and 6 lungs (319,450 patches) for validation. For the LUMC data set, either right or left lung vessels of each case were labeled by two experts independently, in total 10 lungs (504,527 patches) with fully A/V annotations were prepared as an independent test set, from a different patient population, CT protocol and scanner, that was not seen during training.

The CNN3D architecture proposed by Nardelli et al. [8] was used for comparison. The weights of the function $\Phi(\cdot)$ in the CNN-GCN method were initialized randomly using the Glorot uniform initializer. Alternatively, we transferred the learned weights from the CNN3D for initialization of the CNN-GCN method, which we refer to as 'CNN-GCNt'. All three methods were trained and validated with the same data, and their hyper-parameter settings were kept the same: learning rate=1e-3, batch size=128, epoch=100. As a benchmark, the initial annotations of observers were validated against the ground truth. Accuracy was used as a key measure for comparing all methods and observer performance.

4 Results

The trained classification models were used to predict labels of pulmonary A/V, and results were evaluated against the ground truth. The results are available in Fig. 3. With the validation set from the SunYs data set, the automatic methods

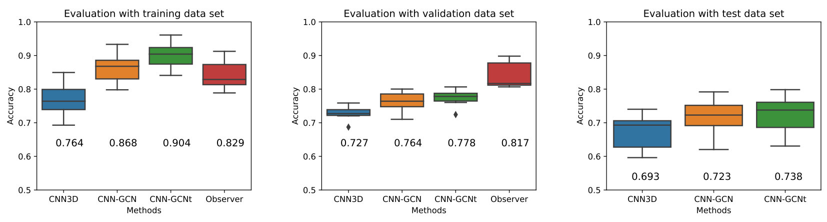

Fig. 3: Accuracy of the automatic methods and observers in training, validation and test sets, respectively.

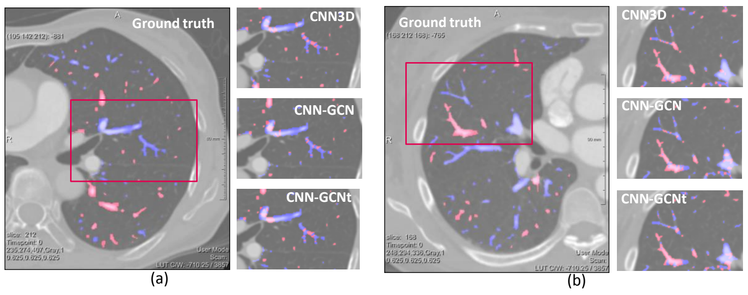

Fig. 4: A 2D visualization of a good and bad result in (a) and (b), respectively, where the accuracies of CNN3D, CNN-GCN and CNN-GCNt in the good one were 0.759, 0.800 and 0.807, respectively; the accuracies in the bad one were 0.687, 0.710 and 0.724, respectively.

obtained median accuracies of 0.727, 0.764 and 0.778 for the CNN3D, CNN-GCN, and CNN-GCNt method, respectively. This compares to a median inter-observer accuracy of 0.817. For the independent test set (LUMC data set), median accuracies of 0.693, 0.723 and 0.738 were obtained for these three methods. Example A/V separation results are shown in Fig. 4, showing a good and a bad case from the validation set.

5 Discussion and Conclusion

We proposed a novel deep-learning-based method by linking CNNs with GCNs, which can be trained end-to-end. The CNN-GCN method can consider both local image and connectivity information. A local batch strategy on graphs was proposed, in order to make graphs with huge amounts of nodes trainable within GPU memory. In the application of pulmonary artery-vein separation, the CNN-GCN method could provide a primary separation, which performs better than the CNN method and obtains slightly worse results compared to observers. There are some limitations of this study. In the test data set, we didn't independently verify the observers' annotations, therefore observer performance wasn't provided during testing. In the future, annotation correction will be added in the test set. Probably over-fitting occurred during training, which may be overcome by adding regularizers or more training samples. Even with GCN, some connectivity errors still remained. Including high-order information (such as branch information) or high-degree neighbors may be helpful in solving these remaining errors. Despite these limitations, we obtained encouraging results from the independent test set, especially considering the fact that these are from a different patient population, CT-protocol and CT-scanner. In conclusion, the proposed CNN-GCN method, with end-to-end training, successfully combines information from images and graphs.

6 Acknowledgements

This research was supported by the China Scholarship Council (No.201406120046) and supported in part by a research grant from the Investigator-Initiated Studies Program of Merck Sharp & Dohme Limited and Bayer AG. The opinions expressed in this paper are those of the authors and do not necessarily represent those of Merck Sharp & Dohme Limited or Bayer AG.

References

1. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
2. Wu, F., Zhang, T., et al.: Simplifying Graph Convolutional Networks. arXiv preprint arXiv:1902.07153 (2019)
3. Monti, F., Boscaini, D., et al.: Geometric deep learning on graphs and manifolds using mixture model cnns. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 5115–5124
4. Parisot, S., Ktena, S.I., et al.: Disease prediction using graph convolutional networks: Application to Autism Spectrum Disorder and Alzheimers disease. Medical image analysis **48** (2018) 117–130
5. Shin, S.Y., Lee, S., et al.: Deep vessel segmentation by learning graphical connectivity. arXiv preprint arXiv:1806.02279 (2018)
6. Charbonnier, J.P., Brink, M., et al.: Automatic pulmonary artery-vein separation and classification in computed tomography using tree partitioning and peripheral vessel matching. IEEE transactions on medical imaging **35**(3) (2015) 882–892
7. Payer, C., Pienn, M., et al.: Automated integer programming based separation of arteries and veins from thoracic CT images. Medical image analysis **34** (2016) 109–122
8. Nardelli, P., Jimenez-Carretero, D., et al.: Pulmonary Artery–Vein Classification in CT Images Using Deep Learning. IEEE transactions on medical imaging **37**(11) (2018) 2428–2440
9. Zhai, Z., Staring, M., et al.: Automatic quantitative analysis of pulmonary vascular morphology in CT images [published online ahead of print June 18, 2019; doi: 10.1002/mp.13659]. Medical Physics (2019)
10. Selle, D., Preim, B., et al.: Analysis of vasculature for liver surgical planning. IEEE transactions on medical imaging **21**(11) (2002) 1344–1357