Neural fiber tractography with Diffusion Tensor Imaging(DTI) and Deep Learning

Yixuan Tan, Zhicheng Gu
University of Wisconsin - Madison
ytan39@wisc.edu, zgu58@wisc.edu

Abstract

In-vivo neural fiber detection can be very useful for studying brain functionalities. With data gathered by diffusion tensor MRI (DTI), people used to perform tractography by tracing single fiber bundle starting from seed points whose performance depends strongly on the choice of seeds. In our project, we overcame this drawback by adopting deep learning network as the new tractography algorithm. We successfully trained our network to predict if a voxel is on neural fiber or not. We also disscussed advantages, disvantages and possible improvements for this DNN-based model.

1. Introduction

Anatomic study of spatial distribution of neural fiber bundles and their connections in human brains has played an important role in studies of brain functions including brain diseases, neural science, psychology etc. However, with more and more complicated situations being studied, the requirement of measurement being noninvasive and in-vivo becomes non-negligible. As the first and mostly used noninvasive, in-vivo imaging modality with the potential to generate fiber-tract trajectories in soft fibrous tissues, Diffusion tensor MRI (DTI) was first adopted for use in clinical neuroradiology during the early 1990s [9]. Its non-invasive nature and ease of whole-brain simultaneous measurement makes it possible to compare areal connections in humans across many cortical and subcortical sites. Furthermore, real-time results can be combined with other real-time in-vivo measures to reveal interconnections between neuron activities and human behavior across individuals [2].

One important aspect of studying DTI involves optimizing tractography algorithms to maximize accuracy and minimize errors in the diffusion tractography data. Diffusion is the spread of molecules in a fluid due to constant thermal motion. The extent of this spread depends on the diffusivity of the medium. In soft fibrous tissues such as brain white matter, diffusion is slower across than along such fibers. Hence, by measuring diffusion along many directions and observing that it is faster in one direction than in others, we can deduce the direction of the fiber bundles at every point in the brain. Traditional DTI fiber tracking algorithms can be divided into deterministic and probabilistic methods, both requires manually added seed voxels to start from and traces down a (probabilistic) voxel pathway along the direction of maximum diffusion speed. Some of the main drawbacks in traditional methods include [6]: 1. Seed: accuracy has a large dependence on choice seeds; 2. Nearsightedness: identification of fiber based solely on change of major diffusion direction in adjacent voxels; 3. Crossing: hard to trace in regions where fibers cross and merge into each other.

In this project, we tried to solve these drawbacks by using a deep learning neural network (DNN). We implemented and trained our DNN to perform tractography over the whole brain. Specifically, we tried to do two things: 1. classify if each voxel is on neural fiber or not and 2. output direction of neural fiber for voxels on neural fiber. The first task was successfully completed while the second was still in working. In this report, we will first introduce the implementation of the DNN-based tractographer (part 2 and 3) and its performance (part 4). Then we will discuss its advantages and disadvantages as compared with traditional methods (part 5), and diffficulties we encountered in doing this project.

2. Data Proprocessing

2.1. Raw MRI Data

The raw MRI data we used was from the Human Connectome Project (HCP) led by Washington University in Saint Louis, University of Minnesota and Oxford University. They provided publicly downloadable data at their CONNECTOME-DB [1]. The data we used was from the "diffusion preprocessed" dataset under "WU-Minn HCP Data - 900 Subjects + 7T". For this project, we downloaded only 10 uncorrelated subjects and used only 4 of them for

training/testing.

2.2. Synthesizing to DTI using Camino

In order to calculate a DTI image from the downloaded raw MRI data (each subject contains 200+ MRI images taken from different angles), we used the camino [7] library from UCL. (Files in bold font are original data in HCP's archive)

1. Generate a scheme file that stores information of gradient correction of each MRI image, data used were in downloaded package(bvecs and bvals):

   ```
   $ fsl2scheme -bvecfile bvecs
   -bvalfile bvals > hcp.scheme
   ```

2. Convert raw data in NIfTI data format to caminos own format .Bfloat:

   ```
   $ image2voxel -4dimage data.nii.gz
   -outputfile dwi.Bfloat
   ```

3. Fit diffusion tensor from raw data:

   ```
   $ modelfit -inputfile dwi.Bfloat
   -schemefile hcp.scheme -gradadj
   grad_dev.nii.gz -model ldt -bgmask
   nodif_brain_mask.nii -outputfile
   dt.Bdouble
   ```

4. Convert the output back to NIfTI format so we can read it to generate feature vector to feed in our NN:

   ```
   $ dt2nii -inputfile dt.Bdouble
   -outputroot nifti_
   -header data.nii.gz
   ```

5. Also, for future visualization purposes, compute fractional anisotropy graph:

   ```
   $ for PROG in fa md; do
   $ cat dt.Bdouble | $PROG |
   voxel2image -outputroot
   $PROG -header data.nii.gz
   $ done
   ```

2.3. Seed

To obtain label/ground truth, we need to apply traditional tractography algorithm on DTI image above. To do that, we need to first manually seed the DTI image. We used the ITK snap [10] tool to open the original data.nii file, which displays it as a figure. We then put some seeds in the figure and save the seeds in a file called cc.nii.gz.

2.4. Tract fibers

Then we go back to camino and use the track tool with default parameters to perform tractography:

```
$ track -inputmodel dt -seedfile
cc.nii.gz -anisthresh 0.2
-curvethresh 60 -inputfile
dt.Bdouble > CCtracts.Bfloat
```

In order to be able to read and visualize our ground truth, we need to convert it to some general file format, we choose to convert to .vtk as follows:

```
$ vtkstreamlines -colourorient
< CCtracts.Bfloat > CCtracts.vtk
```

Then we used Paraview [5] to look at the tracts and save it as a .csv file which we will read and use to generate labels.

2.5. Data Generation

With synthesized DTI image file and traced tracts file, we wrote a piece of matlab code to generate the training and test data. This piece of code can perform three tasks:

1. Form the traced tract file, read all points. Round their coordinates to integer to represent voxel position. From the DTI image, pick these points and store their diffusion tensor data (of 5*5*5 vicinity) and label it as on fiber (label = 1).

2. From DTI data, sparsely sample equally spaced voxels and store same information as in 1). Note here the label is stored as the actual label, but since the on-fiber voxels are very few and compact distributed, there is little overlap between data in 2.5.1 and 2.5.2.

3. Store same structured data (w/o label) for all voxels in a DTI image.

We use 2.5.1 and 2.5.2 to generate training data and 2.5.3 to generate test data to perform DNN-based tractography over the whole brain on.

3. Deep Learning Network

3.1. Framework

We use Tensorflow [4] as the computing system. TensorFlow is an open source software library provided by Google. It does numerical computation using data flow graphs.

We use Keras [3] to build the deep learning model. Keras is a high-level neural networks library, which is running on the TensorFlow or Theano. We choose TensorFlow because it is more famous and easy to install and use. Keras is wriiten in Python, and enables fast building and experimentation. This great feature saves us a lot of time and energy.

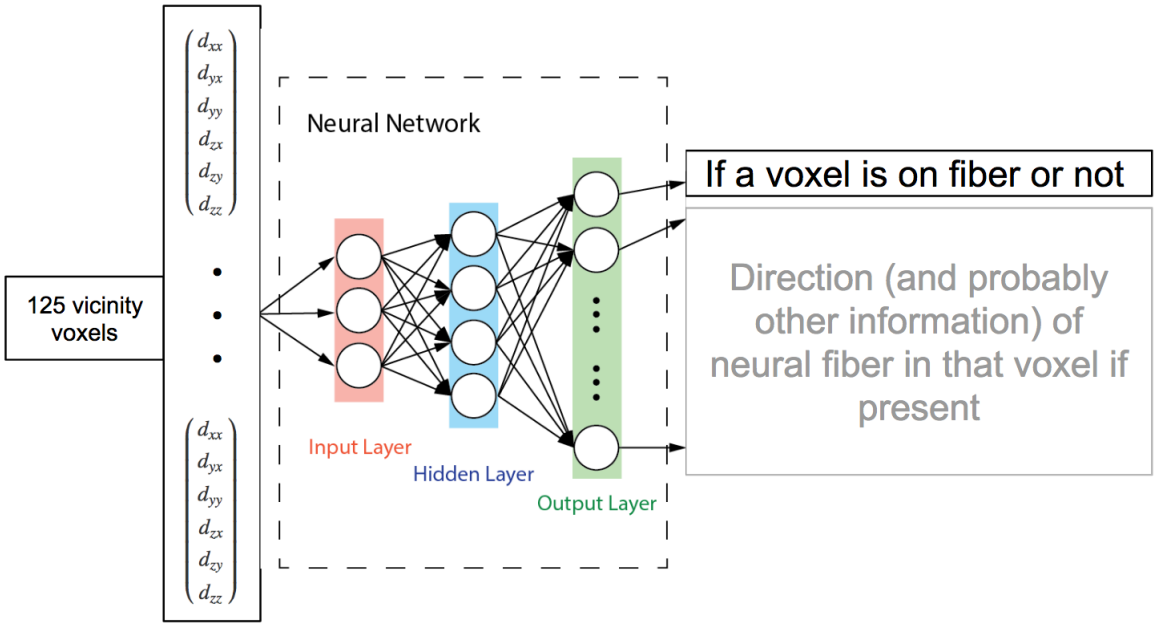

Figure 1. Deep learning network structure.

3.2. Network Structure

We cut the entire figures into 5*5*5 blocks. Each voxels will be at the center of one block. The tensor values of all of the neighbors in this 5*5*5 block is used to predict the feature of the center voxel. The network structure is shown in Figure 1.

We use a fully connected neural network to train the data. The detail of the network is described below. We also consider using the convolutional neural network for this task, but the convolutional neural network in 3D space is hard to implement. Also, the data we are using has 6 parameters at each voxel, which makes it even harder to implement the convolutional neural network.

- Input Layer.

  The input layer of the network has 750 dimensions, which is equal to 6 (number of tensor parameters at each voxel) * 5 (length) * 5 (width) * 5 (height) = 750 features dimensions. Each feature is a float number. The original number is really small, which is around 1E-10 to 1E-15. We scale the features to about -10.0 to 10.0.

- Hidden Layer

  We try different structures of hidden layers, such as [8]. We find that increasing number of hidden layers does not increase the final result. Two hidden layers already gives out the best result we can get. Also, when the neurals in each hidden layers is more than 2000, the result does not increase any more. Hence, the final structure we are using has two hidden layers, each has 2048 neurals.

- Output Layer

  Currently, the output has only dimension, which means if this voxel is on fiber or not. In general, we also want to get the direction of each points on the fibers. This requires the network has multidimensional output. Due to the time limit we couldn't finish this part, but the current result is good enough to find all of the voxels on the fibers.

3.3. Parameters

- Activation Function

  We try the following three activation functions: *relu*, *sigmoid* and *tanh*. The performance of *relu* activation function is really bad. The outputs are all 1 or 0 because the network can't extract meaningful features from the training datas. The performance of *sigmoid* and *tanh* function is similar, and the *sigmoid* function is a little better. So we choose *sigmoid* activation function at the activation function of our network.

- Dropout

  Dropout in deep learning network is used to avoid overfitting. Our network should not have overfitting problem because our training time is relative short due

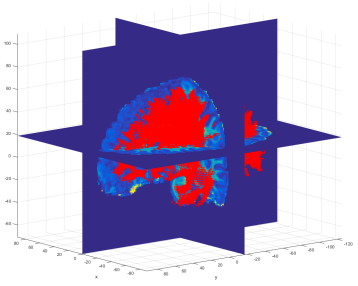

Figure 2. Result of deep learning training.

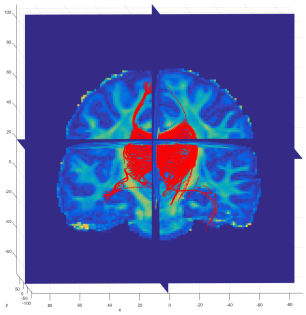

Figure 3. Result of traditional streamline algorithm with seed in the center.

to the resource constraint. Since this technique is always used in deep learning network, we still add a 0.5 dropout parameter to our network. The result of network with dropout is almost the same with the network without dropout.

- Loss Function

  Since the output is only one dimension, we use binary cross entropy as the loss function. This is the most widely used loss function for binary classification.

3.4. Training

We set the training epoch to 20 and batch size to 32. It takes us about ten minutes to training the network with training data from one MRI figure, and it takes more time to train the network with data from four MRI figures. Each MRI figure has more than one million voxels and the file size after processing is larger than 1 GB. Due to the time and resource limit we can't train our network on a larger dataset. The test result does not increase significantly after 10 epochs training.

4. Result

4.1. Prediction Demonstration

Results of an actual prediction using DNN of fiber location in an unseen brain is shown in Figure 2 and Figure 4. The background of Figure 2 shows the fractional anisotropy (FA) of DTI image we used. It has brighter color where diffusion has higher anisotropy and darker color where diffusion is more isotropic. As a result, bright-colored region should be most fiber bundles rest. Compared with FA as well as with Figure 3 which is the result of traditional streamline method, it can be clearly seen that our method predicted if a voxel is on fiber or not correctly for most voxels.

For a better view of this result, we attached .fig files in our submission so that you can drag and look at it from different directions.

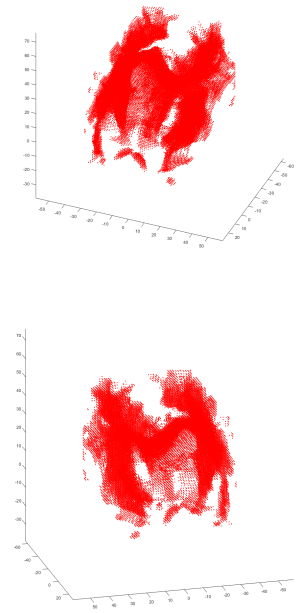

Figure 4. Result of deep learning training: a closer view.

4.2. Accuracy

The confusion matrix of our DNN is shown in Table 1. It shows the number of voxels of correct and incorect classifications. We need to emphasize that a high false positive rate (FPr) is not a big problem here because a high FPr could be because we identified on-fiber voxels not connected to seeds we used in traditional tractography method and therefore was not correctly marked in ground truth.

4.3. Learning Curve

The learning curve is shown in Figure 5. The test set prediction accuracy can reach 90% with only 20000 train-

		Prediction	
		0	1
Truth	0	111419	30859
	1	7624	8798

Table 1. Confusion Matrix

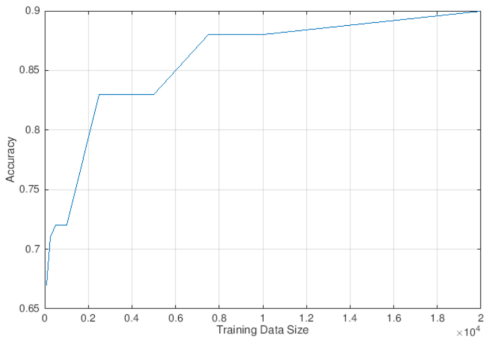

Figure 5. Learning Curve.

ing instances (voxels). Considering we have over 1 million voxels per DTI graph per brain, the efficiency of DNN in tiber tracking tasks is very high.

5. Discussion

5.1. Advantage of DNN

The advantage of using DNN-based tractography model lies in 3 parts:

1. **Seed-Free**: As is obvious, DNN doesnt require initial seeds to perform tractography which makes the algorithm more robust in two folds. On the one hand, this algorithm is operator independent making it stable and widens its potential usage. This means it's more reliable to compare results between different individuals or data acquired in different times. Although difference in DTI data acquisition persists, the effect of choice of different seeds among different samples will be eliminated.

   On the other hand, seed-free means increased accuracy because we can detect some on-fiber voxels which are not connected with seeds and thus can never be detected with seed-dependent algorithms. This can be clearly seen in our result Figure 2 and Figure 3, neural fibers near brain surface are missed with traditional algorithm but correctly captured in our DNN-based model.

2. **Real-time & fully automatic**: With our DNN-based model, its possible to make outputting of neural fiber distribution from measured MRI data fully automatic. Also, since classification using a well trained DNN is very fast, the whole process could be made semi-real-time with bottleneck only at speed of MRI data acquisition. This could enable real-time, in-vivo visualization and analyzation of brain activity on an unprecedented neural fiber bundle level. This could benefit any study that involves characterizing neural reaction to external stimuli.

3. **Convolutional neural network(CNN) with global information**: A DNN-based tractography model can be easily generalized to a CNN-based model. In our implementation, by picking only a 5*5*5 vicinity, we essentially performed the convolution step by hand. However, this implementation eliminated the possible mutual information from other parts of the brain. By generalizing to a real CNN, information from other parts of the brain could be combined into local information in later convolution layers, enabling us to use global brain information to classify each voxel which could add to the global accuracy.

5.2. Disadvantage of DNN

One major drawback for any voxel-based tractography model including DNN-based one, however, is that for non-voxel-aligned fibers and/or different fibers in adjacent voxels, its probable that these algorithms cannot differentiate between them. This can be solved by training the network to output, for each on-fiber voxel, the direction of its fiber and then apply some smoothing to voxels along that direction as postprocessing. If the goal for doing whole-brain neural tractography is only to identify connections between cortical regions, the direction of single fiber bundle can also be omitted.

6. Possible Improvements

6.1. Fiber direction prediction

Right now we only predict whether a voxel is on fiber or not. We mark the voxel with 1 if the voxel is on the fiber and 0 if the voxel is not on the fiber. Although this method can show the entire distribution of the fibers, it is more like finding the white matters in the MRI figure. To get a more meaningful result, we also need the direction of the fibers at each voxels. The deep learning network need to be changed in order to get the direction of the fibers. We don't have enough time to do this because we already spend a lot of time on data processing and building the basic neural network.

6.2. Optimize accuracy

The accuracy of our result can still be improved. To get a better result, we need to tune parameters of our system in a more precise way which usually costs a lot of time (see

details in the difficulties part) and need good understanding of the deep learning algorithm. After tuning we can get a better result with less noisy points.

Also, the feature vector can be changed. In this project we use six tensor values as in the original data. After doing some search we find that we can also get the direction features at each voxel. However, these features seems intuitive and we can't get more meaningful feature because the lack of domain knowledge.

Furthermore, we can also modify the structure of deep learning network. Like we already mentioned, the convolutional neural network usually has a better performance than the fully connected network when the dataset is 2D figures. When it comes to 3D space, the convolutional or other similar methods might also have better performance than fully connected network.

6.3. Try other labeling algorithm

The purpose of this project is to analysis the deep learning algorithm based tractography method. Different from traditional streamline methods, deep learning network needs a lot of training data. To get the label for training data, the only choice we have is to run a streamline algorithm (or some other traditional algorithm) and take the result as the label. Note that this is not the ground truth. If there is any error in the streamline algorithm, our deep learning network could also be influenced.

To get a better ground truth label, we need to get the label from real world or by manual labeling. If we can get the label from real world, this is much easier. If not, we can also get the label by manually labeling. Considering the size of the data, this require a lot of work. Another approach is combian different traditional algorithms and get a labeling system with a higher accuracy. Then use this system to train the deep learning network.

6.4. Release of current work

Since we did make some progress, we could release our code to the community to help further the study. We already gathered all pieces of the code we used (including code for generating training/test data, code for NN and other supporting functions like result visualization) and put them together with some demostration results in the attachment. Furthermore, we need to wrap all of the codes and release them to the community. We hope this could be a useful resource for people who wants do a tractography with deep learning approach.

7. Difficulties in doing the project

The major difficulty we encountered was time vs amount of data. We planned to optimize our network a bit more or even to CNN, but then realized generating training data could take us 5 hours per training/test set (originally this can take longer than 1 day and we optimized our matlab code to get it down to 5 hours). Also, training and classifying over a desired amount of data (20000 training instances for example) can take up to 15 minutes per trial which slows down our pace.

References

[1] The human connectome database. `http://www.humanconnectome.org/data/`.

[2] The human connectome project. `http://www.humanconnectome.org/about/project/tractography.html`.

[3] Keras: Deep learning library for theano and tensorflow. `https://keras.io/`.

[4] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[5] U. Ayachit. The paraview guide: a parallel visualization application. 2015.

[6] P. J. Basser, S. Pajevic, C. Pierpaoli, J. Duda, and A. Aldroubi. In vivo fiber tractography using dt-mri data. *Magnetic resonance in medicine*, 44(4):625–632, 2000.

[7] P. Cook, Y. Bai, S. Nedjati-Gilani, K. Seunarine, M. Hall, G. Parker, and D. Alexander. Camino: open-source diffusion-mri reconstruction and processing. In *14th scientific meeting of the international society for magnetic resonance in medicine*, volume 2759. Seattle WA, USA, 2006.

[8] L. M. G. J. S. Dan Claudiu Ciresan, Ueli Meier. Deep big simple neural nets excel on handwritten digit recognition, 2010. `https://arxiv.org/abs/1003.0358`.

[9] P. Mukherjee, J. Berman, S. Chung, C. Hess, and R. Henry. Diffusion tensor mr imaging and fiber tractography: theoretic underpinnings. *American journal of neuroradiology*, 29(4):632–641, 2008.

[10] P. A. Yushkevich, J. Piven, H. C. Hazlett, R. G. Smith, S. Ho, J. C. Gee, and G. Gerig. User-guided 3d active contour segmentation of anatomical structures: significantly improved efficiency and reliability. *Neuroimage*, 31(3):1116–1128, 2006.