University of Bonn

Master's thesis for obtaining the academic degree
„Master of Science (M.Sc.)“

Detecting Anomalous Curves with Deep Autoencoders

Author:
Ce Liang

First Examiner:
Prof. Dr. Thomas Schultz

Second Examiner:
Dr. Michael Weinmann

Advisor:
M.Sc. Rasha Sheikh

Submitted: November 16, 2020

Declaration of Authorship

I declare that the work presented here is original and the result of my own investigations. Formulations and ideas taken from other sources are cited as such. It has not been submitted, either in part or whole, for a degree at this or any other university.

Location, Date

Signature

Abstract

In the field of tractography, anomalous fibers detection is an unavoidable step for future analyzing, researching, and diagnosing. The traditional methods require medical experts to pick out them. In this report, we explored the possibility of detecting anomalous fibers automatically with deep learning methods.

To build whole processing and evaluating schemes, firstly, a manually detecting method based on three measurements is proposed: cluster confidence index, distance score, and isolation forest. For each measurement, we set a strict threshold to mark the anomalous curves and took the union set as a manually generated label set. We wish the cluster confidence index could find streamlines in abnormal trajectory, the distance score could find streamlines transit out of the region of interest, the isolation forest could find anomalous streamlines within the bundle.

For Deep learning detection, we applied the idea of autoencoders, making the model learn the reconstruction of the inputs. In the detecting step, a high reconstruction means square error implies a high probability of being marked as anomalous fiber. We trained and tested the performance of anomaly detection under deep autoencoder, undercomplete autoencoder, bidirectional autoencoder. For each autoencoder, we also train with different recurrent types: recurrent neurons, long-short memory, and gated recurrent units. Next, we explored the performance of sequence to sequence model, sequence to sequence model with attention mechanism, and ensemble sequence to sequence model.

From the evaluation of the results, we found the manually detecting method could find a rough subset of the anomalous curves, whereas each of the measurements has its drawback and could not satisfy the expectation. The deep autoencoder, undercomplete autoencoder, bidirectional autoencoder all did a better job than the manually detecting method. We also did experiments on the capacities of the sequence to sequence-based models. As a result, we found the deep autoencoder could finish this task better and the result could be adjusted by the demand of user while the sequence to sequence models are more generalized. At last, we summarized the good features of a deep learning model and cleared the direction of future work.

Contents

Acronyms

1 Introduction

The brain is the most important and most complex organ, plus the difference between individuals; it has never been easy to analyze the structure and functionality.

Diffusion tensor imaging (DTI) is the only non-invasive method for imaging human organs and tissues from the perception of the Brownian motion of the water molecules, and DTI is an advanced magnetic resonance imaging (MRI) technique which could quantitatively detect brain tissue. In the past decade, DTI technology is playing an important role in neuroanatomy, neurosurgery, and diagnosis of multiple sclerosis and some brain tumors by detecting human white matter fiber tracts, cardiac morphology, etc [1].

Within the DTI method, one core challenge is: how to reconstruct fiber tracts based on diffusion tensor imaging data. To this end, many brain white matter fiber tract tracking algorithms have been proposed. The basic idea is estimating the second-order tensor of each voxel, selecting the direction of the main feature from the tensor as the tracking direction, the final fiber bundle, as a result, is a series of points traced sequentially from the designated initial seed point. Some basic algorithms are: FACT(Fiber assignment by continuous tracking)[2], TEND(Tensor deflection) [3] and so on.

However, even processed by the state of art tractography algorithms, the result still contains a number of anomalous curves. To tackle these anomalous curves, prior knowledge of anatomy is usually required, the researcher needs to know geometric features of normal fibers, manually remove them, or apply some highly subject-oriented algorithm. Figure 1.1 compares one possible situation what fibers look like in space before and after anomalous fiber removal.

In the computer science field, we have more interest in finding an automatic approach to find anomalous fibers. Techniques of machine learning, deep learning could detect hidden features which hardly expressed by physic and mathematics theories, they have been successfully managed several challenging tasks, such as sequence matching, pattern recognition, anomaly detection[5], clustering, trend analysis, similarity detection, classification and long and short term prediction[6]. Therefore, for this task of anomalous fiber detection and removal, we could simplify it as anomaly detection, and for the unity data, a fiber is a sequential data

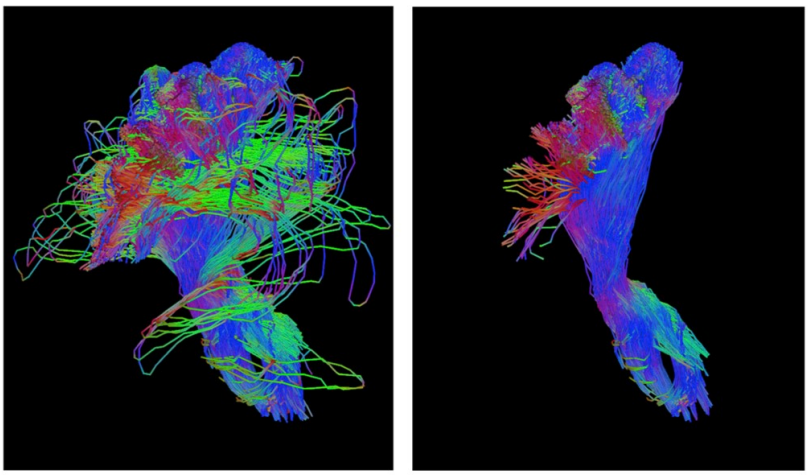

Figure 1.1: Left: A result of fiber tracking. **Right:** The result after manually anomalous fiber removing. Image from[4]

composed of a series of space positions of vertexes. One intuitive idea is using a sequential deep anomaly detection model. Since lacking ground truth from medical experts, an unsupervised learning model fits more to our task, so we explored the performance of deep auto-encoding models.

Structure of the thesis

1. Chapter 3 introduces the background knowledge of the model and principles we applied in this anomalous curve detection task. At the same time, we applied a framework for this task.

2. Chapter 4 described the dataset and types of anomalous curves. We also proposed a manual detection method here.

3. Chapter 5 described the autoencoder structures we utilized to reconstruct the fibers.

4. In chapter 6 we analyzed the performance of manually detection and deep learning detection. We evaluated the results of 13 models and explained why the model worked or not.

2 Related Work

The traditional way of anomalous curves detection is the geometry-based method. In [4], they proposed a meticulous method based on geometry and expert experience to remove anomalous curves in bundles: first manually define ROIs of anomalous fibers and remove them, second remove streamlines that twisting and making 180 degree turns within a length of a threshold, then manually retrieve some removed normal streamlines.

Applying anomaly detection in sequential data automatically can be organized into two categories. The first category detects anomalous series from a group of streamlines. Research in the second category identifies if an input streamline is an anomaly or not. This report concerns the anomaly detection in the second category under the deep learning method.

In studies of the first strategy, [7] uses the Nearest Neighbor based method, the anomaly score is calculated by distance of a sequential data to its k nearest neighbors. The clustering algorithm could also work in this task. In [8], they cluster a group of data by the CLARA k-medoids algorithm [9], then assign the outlier score of a sequential data by calculating the distance the medoid to the closest cluster. Cabrera et al.[10] break sequential data into overlapping subsequences. They created dictionaries of normal subsequences and abnormal subsequences. For each data, the anomaly score is counted from the scores of the subsequences; the sequential data is an anomaly if its subsequences do not exist in the normal dictionary or anomaly in the abnormal dictionary.

For the second strategy, local outlier factor (LOF) [11] could detect more precisely in local outliers since it enables to calculate the local density; this is a clustering-based method as well. In [12], they extract mean and covariance values for each streamline and applied One-Class SVM [13] to mark the anomalous curves.

The deep learning based studies also follow the idea of the second strategy. In [14], they did multiple experiments on the MNIST dataset that the autoencoder based model has good performance with a wide range outlier ratio in an unsupervised way. In [15], they proposed a framework of outlier detection for time series using deep recurrent autoencoder models; they also found that embedding the contextual information could improve the performance. In [16], they proposed

an ensemble learning suture based on the sequence to sequence model to improve the performance.

3 Preliminaries and Framework

In this chapter we defined the task of anomalous curves detection, and introduced the background knowledge of the structures and models we used in this task. At last we proposed a framework of this task based on three parts: preprocessing, manually anomaly detection and anomalous curves detection with deep autoencoders.

3.1 Task Definition

Sequential Data A group of N curves are represented in a sequential format $\mathcal{C} = \langle \mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_N \rangle$. Each sequential data $\mathcal{S}_i = \langle \mathbf{s}_i^1, \mathbf{s}_i^2, ..., \mathbf{s}_i^L \rangle$ is a series of vectors, the vectors are the vertexes of the streamline, L is the number of vertexes. And each vector contains the spacial information in the coordinate, $\mathbf{s}_i^j = (x_i^j, y_i^j, z_i^j)$. This example is also called multivariate sequential data since s is a vector.

Anomaly Detection Given a group of curves $\mathcal{C} = \langle \mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_N \rangle$, we compute the anomaly score $AS(\mathcal{S}_i)$, a high anomaly score implies there is high possibility that this curve is an anomaly. IF we could find a threshold $thre$ properly, then :

$$\mathcal{S}_i \text{ is an anomaly if } AS(\mathcal{S}_i) > thre$$

3.2 Autoencoder

The deep auto-encoder is a stack of multiple auto-encoders (AE). AE is an unsupervised learning technique proposed in 2006 [17] and has been proved have good performance in data dimensionality reduction [17], image feature extraction [18], and Alzheimer's disease diagnosis [19]. According to the functions, they can be roughly classified into standard auto-encoders, undercomplete auto-encoders, sparse auto-encoders, and so on.

Fig3.1 demonstrated the structure of a standard autoencoder, it is composed of three layers: input layer, hidden layer and output layer. The encoding network is realized by the input and hidden layer: $\mathbf{h} = f(\mathbf{x})$, and the decoding process is

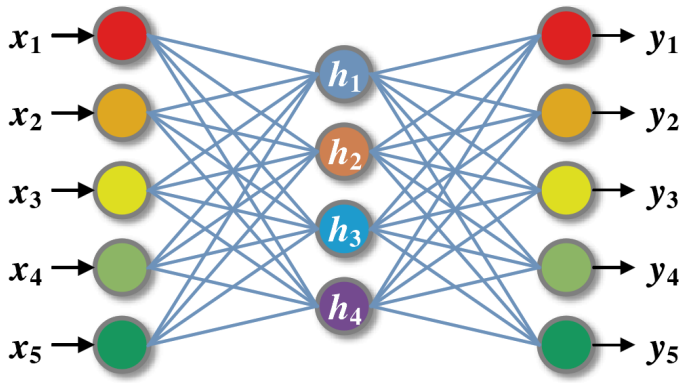

Figure 3.1: Structure of a standard autoencoder, three layers, form left to right: input layer, hidden layer, output layer.

realized by the hidden and output layer: $\mathbf{y} = f(\mathbf{h})$, where $\mathbf{x} = [x_1, x_2, ..., x_n] \in \mathbb{R}^n$ stands for input values, $\mathbf{h} = [h_1, h_2, ..., h_m] \in \mathbb{R}^m$ stands for hidden values, $\mathbf{y} = [y_1, y_2, ..., y_n] \in \mathbb{R}^n$ stands for output values. $f(\mathbf{z})$ is the activate function, in encoder, $\mathbf{z} = \mathbf{W}_E x + \mathbf{b}_E$, in decoder $\mathbf{z} = \mathbf{W}_h x + \mathbf{b}$, $\mathbf{W}_E \in \mathbb{R}^{m \times n}$ and $\mathbf{W}_D \in \mathbb{R}^{n \times m}$ are connecting weights in encoder and decoder network,

The autoencoder model extract the feature expression by reconstructing the input from hidden values, in another word, minimizing the difference between input $\mathbf{x}$ and $\mathbf{y}$, the loss J_{AE} is defined as:

$$J_{AE} = \frac{1}{2N} \sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \mathbf{y}^{(i)}\|_2^2 \tag{3.1}$$

where N is the number of training samples and $\|\|_2$ is the norm two calculation. Essentially, the learning process of AE is an unconstrained minimization problem and the domain is the real domain. From Eq.3.1, we could say that the objective function is a quadratic convex function, so it can be inferred that the learning process of AE is a convex quadratic optimization problem. Batch Gradient Descent (BGD) [20] and Stochastic Gradient Descent (SGD)[20] are two of the common methods to solve the problem.

3.3 Recurrent Neural Network

Recurrent neural networks (RNNs)[21] are a family of neural networks have a cycling structure: for each hidden layer, the output is also input in next state,this mechanism enable the network learn from history, as a result, RNNs are specialized for processing sequences data $\mathbf{x}^{(1)}, ..., \mathbf{x}^{(\tau)}$.

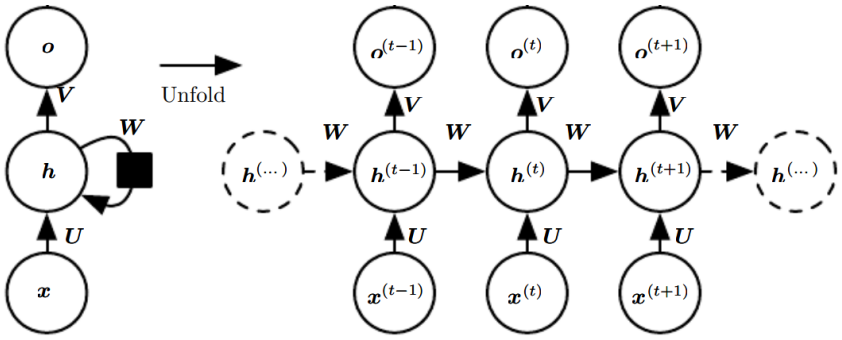

Figure 3.2: Overall and unfolded Structure of a RNN, image from[21].

Figure 3.2 shows the structure of a simple RNN, firstly the hidden state $\mathbf{h}^{(0)}$ is initialized, then at time t, the input of RNN is $\mathbf{x}^{(t)}$, the hidden layer result is $\mathbf{h}^{(t)}$, $\mathbf{o}^{(t)}$ stands for the output result, they are updated as follow, where $\mathbf{U}, \mathbf{V}$ and W are weight matrices and $\mathbf{b}$, $\mathbf{c}$ are biases.

$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} \tag{3.2}$$

$$\mathbf{h}^{(t)} = tanh(\mathbf{a}^{(t)}) \tag{3.3}$$

$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)} \tag{3.4}$$

The parameters in RNN are updated by gradient descent through back propagation as well. Since the back-propagation is based on the time/sequential dimension, the back-propagation algorithm for RNN is also called BPTT (back-propagation through time).

When the length of sequential data is getting longer, the RNN model will lost its capacity, the model could hardly learn the relation between current data and the data long time ago. As two of the most popular variant, long-short term memory and gated recurrent unit could deal with the problem.

3.4 LSTM and GRU

The long-short term memory(LSTM)[22][23] retains the external chain network structure of the RNN, but the structure and organization within the LSTM network is different. In RNN the recurrent mechanism is controlled just by a tanh function, while the recurrent module of LSTM is composed of three gates(input gate, output gate, and forgetting gate) to control the network, which greatly enhances the long-term memory capability of the network, and the gradient disappearance problem is effectively solved.

The three gates mechanism in LSTM also improved the memory ability is improved, and filtered out unimportant information more efficiently, then reduced the training time.

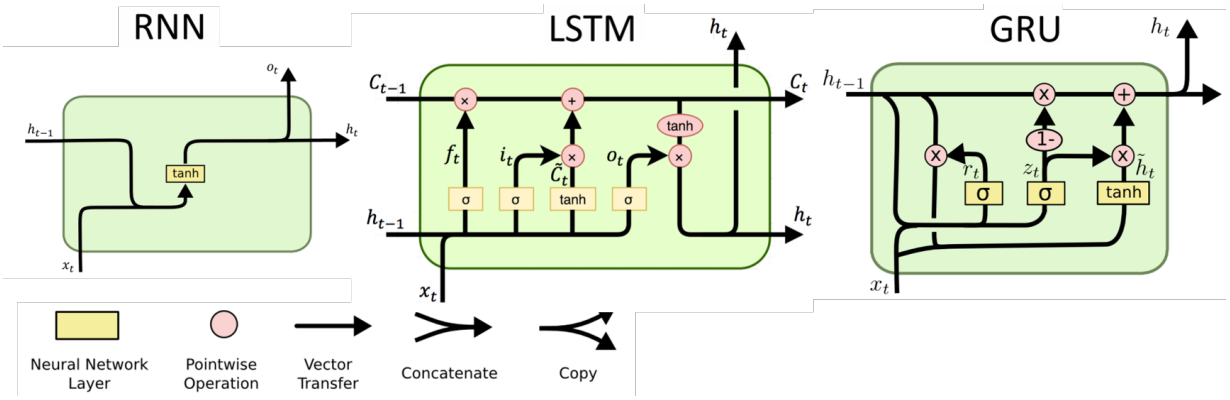

Figure 3.3: Difference of the recurrent module in RNN, LSTM, GRU. Image from [24]

Figure 3.3 compares the setting of the recurrent module in RNN, LSTM, GRU. We could see that in LSTM, another type of information $\mathbf{C}_t$ is spreading between recurrent modules. The following equations 3.5-3.8 show how values are calculated, where $\sigma()$ is the sigmoid activate function. in equation 3.6, the result of forget gate $\mathbf{f}_t$ is determined by hidden state $\mathbf{h}_{t-1}$ and input $\mathbf{x}_t$, in equation 3.6, the input gate result $\mathbf{i}_t$ and candidate cell value $\check{\mathbf{C}}$ are calculated, in 3.7, the real cell value is $\check{\mathbf{C}}$, equation 3.8 shows how to get the final output result $\mathbf{o}_t$.

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f\right) \tag{3.5}$$

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i\right), \quad \check{\mathbf{C}}_t = tanh\left(\mathbf{W}_C \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C\right) \tag{3.6}$$

$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \check{\mathbf{C}}_t \tag{3.7}$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o\left[\mathbf{h}_{t-1}, \mathbf{x}_t\right] + \mathbf{b}_o\right), \quad \mathbf{h}_t = \mathbf{o}_t \odot tanh(\mathbf{C}_t) \tag{3.8}$$

Gated recurrent unit(GRU)[25], is a simplified version of LSTM, it is composed of update gate $\mathbf{z}_t$ and reset gate $\mathbf{r}_t$, and the update gate $\mathbf{z}_t$ in GRU could be considered as the merged result of forget gate $\mathbf{f}_t$ and input gate $\mathbf{i}_t$ in LSTM. The gates values in GRU are calculated as follow:

$$\mathbf{z}_t = \sigma\left(\mathbf{W}_z \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_z\right) \tag{3.9}$$

$$\mathbf{r}_t = \sigma\left(\mathbf{W}_r \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_r\right) \tag{3.10}$$

$$\check{\mathbf{h}}_t = tanh\left(\mathbf{W}_h \cdot [\mathbf{r}_t \odot \mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_h\right), \quad \mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \check{h}_t \tag{3.11}$$

3.5 Seq2Seq and Attention

The sequence to sequence model(seq2seq) was invented in natural language processing area aimed to enable both input and output of arbitrary length, see fig 3.4. It has an end to end structure, composed with an encoder, a decoder and a context vector $\mathbf{z}$ in between. In the encoder part, the recurrent unit is working as the normal RNN, the hidden state is determined by the current input and hidden state from last time state:

$$\mathbf{h}_i^t = f * (\mathbf{h}_i^{t-1}, \mathbf{x}_i^t) \tag{3.12}$$

And in the decoder, since there is no input sequence, just the context vector, the hidden state is updated by the hidden state from last time state and the encoder's output from last time state, the equation is:

$$\mathbf{h}_j^t = f * (\mathbf{h}_j^{t-1}, \mathbf{o}_j^{t-1}, \mathbf{z}) \tag{3.13}$$

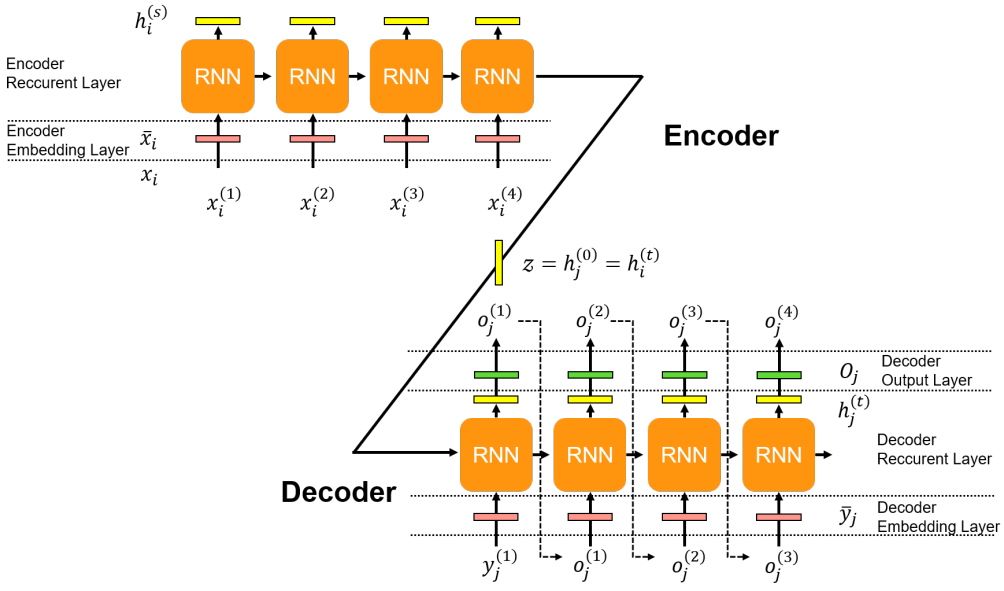

Figure 3.4: Structure of a sequence to sequence model.

Attention mechanism is borrowed from the attention mechanism of the human brain that processes signals, the goal is to learn and have more attention to the critical information over the global information. The Attention mechanism is helping the model to focus on a subset of features, i.e., to select feature elements that are more critical to the target's output. Besides, the Attention mechanism could also be used as a resource allocation scheme in the case of limited computing power, allowing more important tasks to receive more computing resources [26]. The rough process to add it to the seq2seq model is as follow:

$$\text{[Bahdanau]} \quad \mathbf{score}_j^t = FC\left(tanh\left(\mathbf{EO}^i\right) + FC\left(\mathbf{H}_j^t\right)\right) \tag{3.14}$$

$$\text{[Luong]} \quad \mathbf{score}_j^t = \mathbf{EO}^i \times \mathbf{W} \times \mathbf{H}_j^t \tag{3.15}$$

$$\mathbf{attention}_j^t = softmax\left(\mathbf{score}_j^t, \text{axis} = 1\right) \tag{3.16}$$

$$\mathbf{z}_j^t = sum\left(\mathbf{attention}_j^t \times \mathbf{EO}^i, \text{axis} = 1\right) \tag{3.17}$$

$$\mathbf{De\_in}_j^t = concat\left(\mathbf{z}_j^t, \mathbf{x}_j^t\right) \tag{3.18}$$

Firstly the attention score is calculated, most popular two scores are Bahdanau attention and Luong attention. In equation 3.14, FC stands for a fully connect layer, $\mathbf{EO}^i$ stands for the output of the encoder, $\mathbf{H}_j^t = ep\left(\mathbf{h}_j^t\right)$, ep stands for copying and expand the dimensionality for computational matching. In 3.15, $\mathbf{W}$ is a weight matrix, the other definitions are the same. Once we have the attention score, the attention weights could be got from 3.16, then calculate the new context vector $\mathbf{z}_j^t$. Finally the decoder input $\mathbf{De\_in}_j^t$ is done by equation 3.18.

3.6 Framework Overview

We proposed a framework utilizing the deep autoencoders for anomalous curves detection, see figure 3.5. Firstly we preprocess the raw streamline collection by removing the identical fibers and subsampling the fibers into a lower number of vertexes for complex models. Then we feed the preprocessed sequential data to the deep autoencoder models to learn the reconstruction of curves. For each curve, we calculate the reconstruction mean square error and consider it as the anomaly score $AS(\mathcal{S}_i)$. At the same time, we also implemented a manually detecting method and utilized the results as a referencing result to evaluate the performance of different deep autoencoder model.

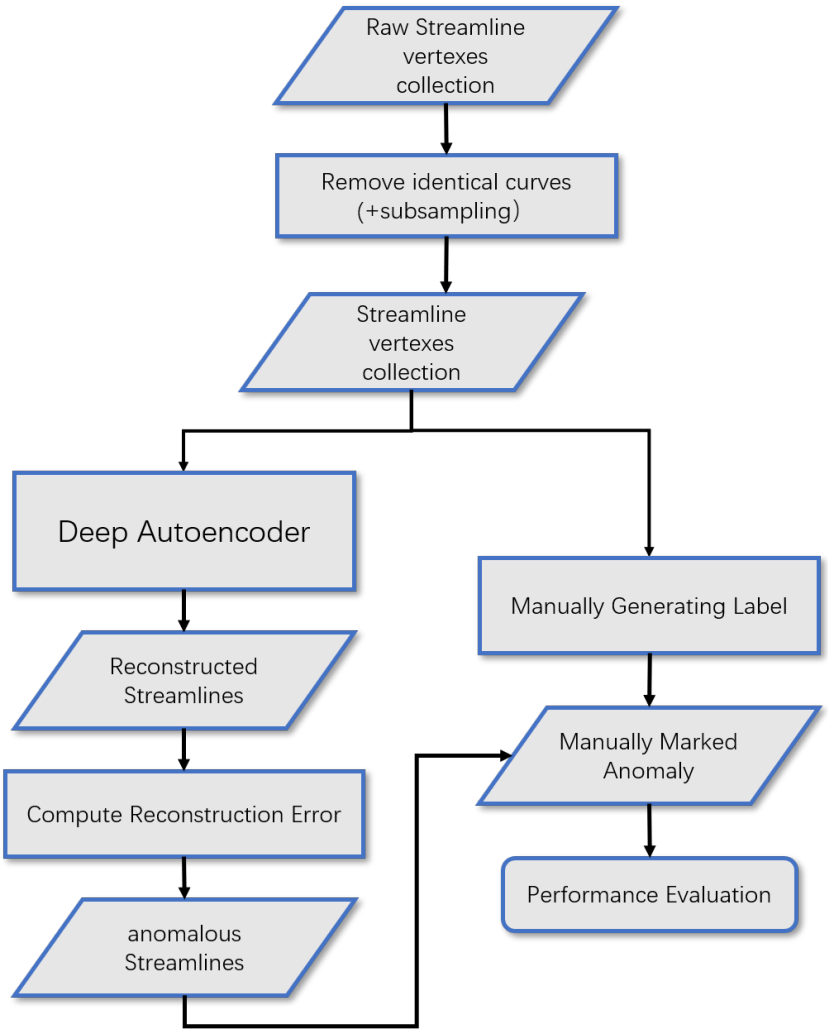

Figure 3.5: Framework Overview of the task.

4 Data Pre-processing

4.1 Dataset description

The data in this experiment is originate from the Human Connectome Project(HCP), a consortium of HCP investigators studied a population of 1200 healthy adults using multiple imaging modalities[27]. From the HCP, we could get very high-quality in-vivo MRI data, diffusion MRI (dMRI) data and related description and related mask data to run the tractography algorithm. The following described one possible method to get streamlines of fibers.

1. In HCP, for each subject, download the DMRI, related description and related mask data.
2. Extract the DWI (diffusion-weighted image) and the weighted(T1w) image.
3. Generate the five-tissue-type (5TT) segmented tissue image.
4. Derive MSMT-CSD tissue response functions based on a co-registered five-tissue-type (5TT) image
5. Estimate the fibre orientation distribution from diffusion data using constrained spherical deconvolution.
6. Extract the first three picks of the fibre orientation distribution.
7. Apply the tractography algorithm to the white matter fibre orientation distribution to generate streamlines.

In this experiment, we used the result streamlines of [28], they represented fiber orientation distribution functions as higher-order tensors, applied a novel positive definiteness constraint (H-psd) and achieved higher angular resolution result.

There are white matter fibers from 141 subjects in total, the number of vertexes of fibers differ a lot, mainly in the range from 400 to 600, we truncate them to 550 and pad with zero vectors. For the complex models we subsampled the fibers to number of 20 and 50 vertexes. For the part of manually detection, we applied

the methods to all fibers of all subjects. For the part of anomaly detection with deep autoencoders we split the subjects into $6 : 2 : 2$ for training, validating, and testing.

4.2 Outlier description

Even in the state of art tractography algorithm, there still exist a number of anomalous curves. This is due to the drawbacks in data acquisition steps. Here are three aspects that might cause anomalous curves in the final result.

1. Noise and artifact problem. Since the signal-to-noise ratio of DTI data is low, a series of phenomena such as disordered and irregular directions[29][30], will appear in the tensor data contaminated by noise. Besides, human motion and uneven magnetic field during tensor data acquisition will cause image artifacts.

2. Partial volume effect. The resolution of diffusion tensor imaging data voxels is approximately $1mm \times 1mm \times 1mm$, much larger than the diameter of individual fibers (typically no more than $20\mu m$). In the actual brain fibers, there are some multiple fibers structures, such as crossings, bifurcations, and sectors. Whereas the ordinary DTI model could only estimate the diffusion state of potential water molecules and consider high-order tensor values is direction of all fibers crossing the current voxel.

3. The problem of angular resolution. Theoretically, only 6 directions of the diffusion weighted imaging signal one could calculate the diffusion tensor value. However, this sparse angular sampling makes the calculated results very coarse, which affects the final fiber visualization[31][32].

From the described reasons and the observation in experiments, we found there are four main types of anomalous fibers:

1. Fibers transit out of the region of interest. The fibers we are dealing with are just depicting some of the tracts. For the streamlines transit out of the region of target tracts, they are obviously considered as an anomaly.

2. Abnormal trajectory within the target tract region. Sometimes the tractography algorithm gave wrong continue direction at some voxels. This kind of fibers will have a very different continued direction (90 degrees) to their neighbors.

3. Short fibers. This is an application-oriented type of outlier, should be decided by the user to consider or not.

4. Fibers at the border of the target tracts region. This is a difficult type for anomaly detection. The target tract region is different in subjects and a good feature of our detector to let the user decide this part, and the fibers in the main body should not be affected at the same time.

4.3 Manually Anomaly Detection

Our task is to detect anomalous curves with deep autoencoder models, and in fact, there are no ground truth labels to tell anomalous streamlines or not. To compare with some geometry-based algorithms and to have a roughly evaluation of the performance for our deep learning models, it would be nice to generate the label data combined with methods below.

4.3.1 Distance transform

As described in the last chapter, there exists a situation that multiple streamlines crossed in the same voxel; the tractography algorithm might follow the wrong direction. As a result, some fibers have already place out of the region that bundle belongs to. These streamlines are definitely anomalous ones. One easy method is to mask them out.

However, there are no precise masks for the fiber bundle we are using. We could find approximate ones from the project [4]. Besides, the mask is subject-depended since the detailed position of subjects differs a little. A compensate solution is to create a mask with a 'soft edge.' We could use the idea of Distance transform: for each voxel, calculate the distance to the nearest mask region volume[33]. For each streamline, we calculate the distance score and then decide it an anomalous streamline or not.

In fig 4.1**(a)** we demonstrated the idea to calculate the distance score with the distance transformation. Each small square refers to a voxel in our task. The labeled number stands for the distance that voxel to the mask volume, the mask volume is labeled with zero. In the figure, the toy mask is the white 'triangle' region. The red dotted line stands for a streamline going throng space. The distance score is $2.24 + 1 + 0 + 0 + 1 = 4.24$. In the 4.1**(b)** we showed a slice of the mask volume, the black part stands for the mask region. In the 4.1**(c)** we showed the corresponding distance matrix, the color is related to the distance value of the voxel. Algorithm 1 showed the detailed step to get a distance score for each

streamline. In the step calculating the distance matrix, we applied a weight matrix $[1, 0.83, 1]$ to match the real resolution of the dMRI.

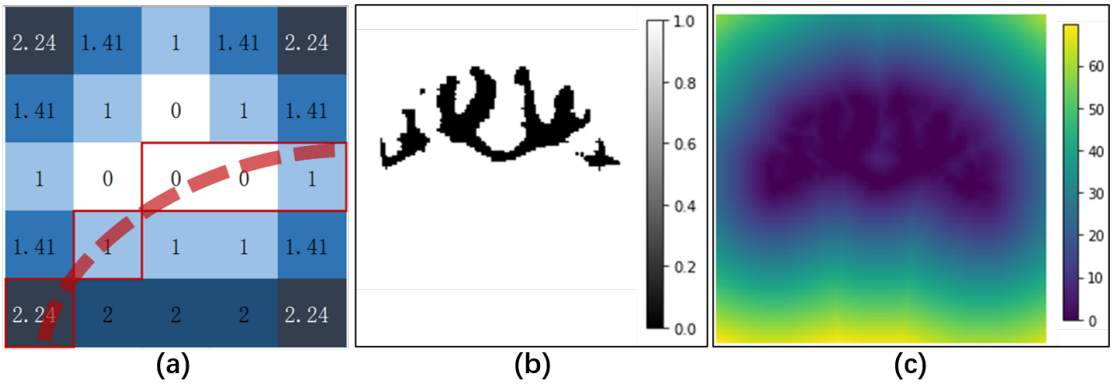

Figure 4.1: **(a)**: A toy example to demonstrate the relation with mask region, distance matrix and streamline distance score. **(b)** A example of sliced mask volume. **(c)** The distance matrix of **(b)**

Algorithm 1 Distance score

Intput:

streamlines, affine transfer matrix

Compute:

1. Download tracts masks from **Tract**
2. Create the mask volume by the union of CC_3, CC_4, and CC_5
3. Apply affine transformation to the mask, make streamlines position and the mask in the same coordinate.
5. calculate distance transformation matrix **Dis**
6. for streamline **s** in streamlines:

    score $= 0$

    for point **P** in streamline $\mathbf{s}[i]$:

        $x, y, z = \mathbf{P}[j]$

        score $+=$ $\mathbf{Dis}[x, y, z]$

Output:

Distance scores

4.3.2 Cluster Confidence Index

The Cluster Confidence Index(CCI) is a method to measure the similarity of a streamline within a group of streamlines [34]. It could be a metric to find out by the assumption that pathways represented by many streamlines following roughly the same trajectory have higher confidence than those streamlines that follow pathways alone.

The CCI is calculated using the minimum average direct-flip(MDF) distance, which measures the Euclidean distance between corresponding points on two selected streamlines to quantify the similarity between streamlines[35]. The CCI of a streamline could be considered as the weighted sum of the MDF distance pairwisely of the streamline and all of the other streamlines among the dataset. A toy model of a streamline dataset containing three streamlines are shown in Figure 4.2.

$$CCI_i = \sum_j \frac{1}{MDF\,(S_i, Sj)^K} \forall MDF\,(S_i, Sj) < \theta \tag{4.1}$$

In equation 4.1, S_i stands for a streamline of a group. K and. θ are two control parameters, default $K = 1$, default $\theta = 5mm$ A high values of K will decrease the contribution of a dissimilar streamline to the CCI. The maximum MDF distance θ could make the calculation faster and not fall in to the confidence index inflating when number of streamlines is large.

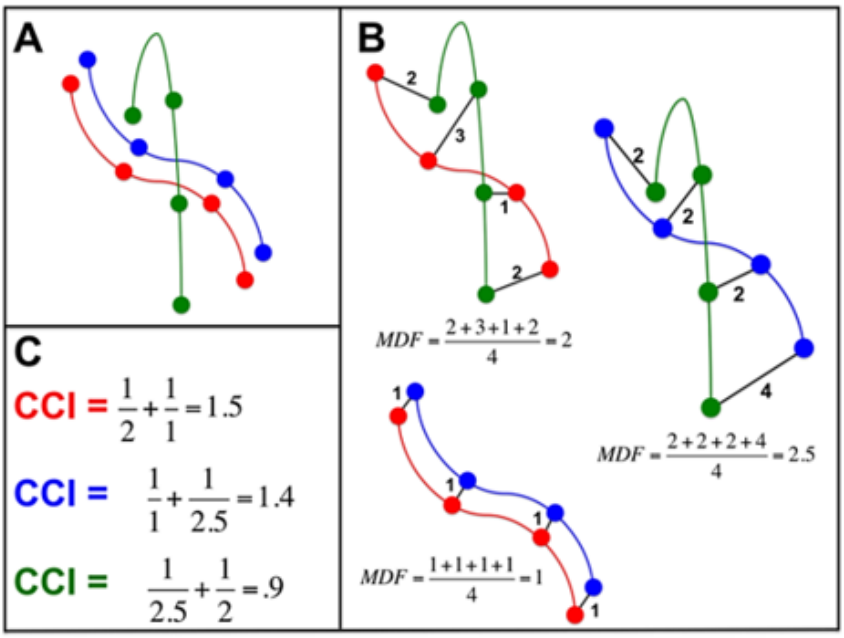

Figure 4.2: Demonstration CCI calculation:(A) There are two streamlines following a similar trajectory (red and blue) and one that does not (green). (B) Calculation of the pairwise minimum average direct-flip (MDF) distance between streamlines subsampled to four points. (C) Calculation of CCI (K = 1, θ = 10), red and blue streamlines with similar CCI values and green with a lower CCI. [34].

4.3.3 Label Generation

An intuitive idea to mark the anomalous streamlines is to set thresholds for CCI and distance score $thre_{cci}$ and $thre_{dt}$. For a streamline, if the CCI is lower than $thre_{cci}$ or the distance score higher than the $thre_{dt}$, there is a high probability that this streamline is the anomalous one.

However, setting thresholds to CCI and distance score have obviously drawbacks: for distance scoring, this method could not find anomalous streamlines within the mask region; for the CCI, the index is only calculated within the neighbors, see the setting of θ in equation 4.1, the density of streamlines distribution effect the CCI value a lot. Besides calculating the MDF of two streamlines, the two streamlines should sub-sampled to the same number of points, which violate the Nyquist sampling theorem, and detailed variation is eliminated. To compensate for the drawbacks above, a third method is introduced.

The **Isolation Forest** algorithm is an anomaly detection method based on ensemble learning. It could achieve linear time complexity and relatively high anomaly detection accuracy. Isolation Forest consists of isolation trees. An isolation tree is a random binary tree where each node connects two child nodes or directly a leaf node. A randomly sampled subset of data is used to construct the isolation tree to ensure the difference between the different trees. To construct an isolated tree, a random feature is selected, and a segmentation value is randomly chosen to recursively segment the dataset until any of the following conditions are met:

(1) The tree reaches a restricted depth.
(2) Only one sample at the node.
(3) All sampled features of the nodes are the same

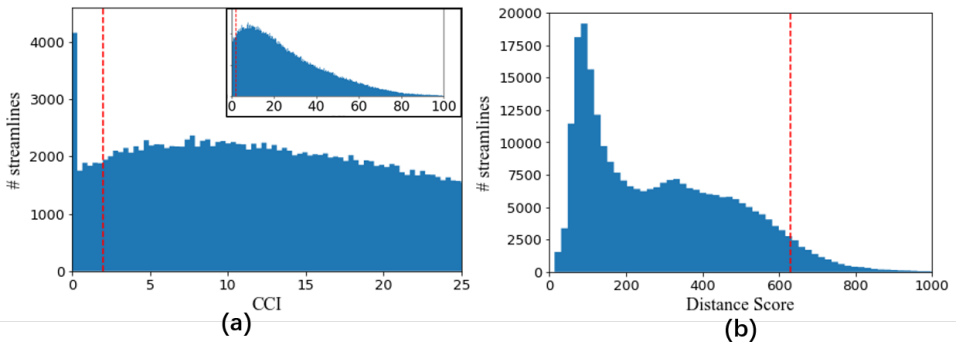

Figure 4.3: **(a)**: histogram of the CCI w.r.t. number of streamlines, the small one in the top right is the overall histogram. **(b)**: histogram of the distance score w.r.t. number of streamlines.

By multiple times of experiment, we proposed a label generation method of algorithm 2, for each of the three methods above, we only select the worst performance parts as anomalous streamline, then take the union of anomalous streamlines of the three methods. As for setting the values of thresholds, a pair of values with good performance is: $thre_{cci} = 2$ and $thre_{dt} = 630$.

Algorithm 2 Generating Labels

Intput:

streamlines of subjects, $thre_{cci}$ and $thre_{dt}$

Compute:

label: 1: normal streamline, 0: anomalous streamline

1. for all streamlines of all subjects:
2. extract diagonal values of co-variance matrix of streamline
3. apply isolation forest with extracted features, get anomaly detection result **IF**.
4. for each subject *subj* in Subjects:
5. remove identical streamlines for *subj*[i]
6. calculate cluster confidence index **CCI**
7. calculate distance score **Dis**
8. for streamline **S** in Streamlines of *subj*[i]
9. cci_tmp = 0 if **CCI**[j] $< thre_{cci}$ else 1
10. dis_tmp = 0 if **Dis**[j] $> thre_{dt}$ else 1
11. final_label = 1 if cci_tmp + dis_tmp +IF[i][j] > 2 else 0

Output:

manually generated labels

5 Deep Autoencdoers

This chapter introduced the structure we explored for the detection of the anomalous fiber with deep autoencoders. We started with a deep fully connected autoencoder, then we changed the neuron units of some layers to the recurrent units. In this thesis, we call the deep autoencoder with recurrent units the deep autoencoder since we put more workload on them and they are more efficient in dealing with sequential data. After that, we built two subtypes of the deep autoencoder: undercomplete autoencoder and bidirectional autoencoder. For each deep autoencoder, we tried with RNN, LSTM and GRU as the recurrent unit. After the deep autoencoders, we also built the sequence to sequence based models: sequence to sequence with attention, ensemble sequence to sequence model.

5.1 Deep Autoencoder Model

5.1.1 Fully connected autoencoder

We would like to start with building a fully connected autoencoder, the structure is depicted in figure 5.1**(a)**, the estimated output size of each layer is marked in the blocks, B_N stands for batch normalization. We could observe that the input and out put of each layer are rank two vectors, the model is worked by computing the dot product between the input and a kernel. For example in a layer the input is 3×2000, the estimated output is 3×5000, within this layer, it create a kernel with size of 2000×5000.

The structure is organized from the idea of autoencoder in chapter 3.2, the input is firstly embedded to a higher dimension space , then the model tried to learn the reconstruction from this space and project it back to the original space. As a matter of fact, to find a good setting of structure and layer size, each configuration is trained and validated, the structure in figure 5.1**(a)** performance the best so far in fully connected autoencoders.

5.1.2 Deep autoencoder

The deep autoencoders is referring the ones with recurrent units as described in the beginning of this chapter.

Figure 5.1**(b),(c)** demonstrated the structure of the deep autoencoder model. The yellow block stands for a fully connected layer, the green one is a Recurrent neuron layer. From the input to the recurrent layer, we embedded the input data in to a high dimensional space. Steps between the embedding module and projecting module could be considered as the real autoencoding process, then we project the high dimensional data back to the position format. This embedding-projecting process is a basic operation in natural language processing tasks, so we borrowed this idea in our task. The embedding-projecting is also exist in the fully connected autoencoder and has been proved worked. The number in bracket is the number of neurons of that layer. For the deep autoencoder, the operation of batch normalization is essential, the model could hardly start to learn without it.

The input is the position of streamline $s_i = (x_i, y_i, z_i)$, and the number of recurrent times is length of the streamline, in our experiment the value is 550. The expected output value is the same as input value. The idea of this model is trying to reconstruct the input from the expanded version of input.

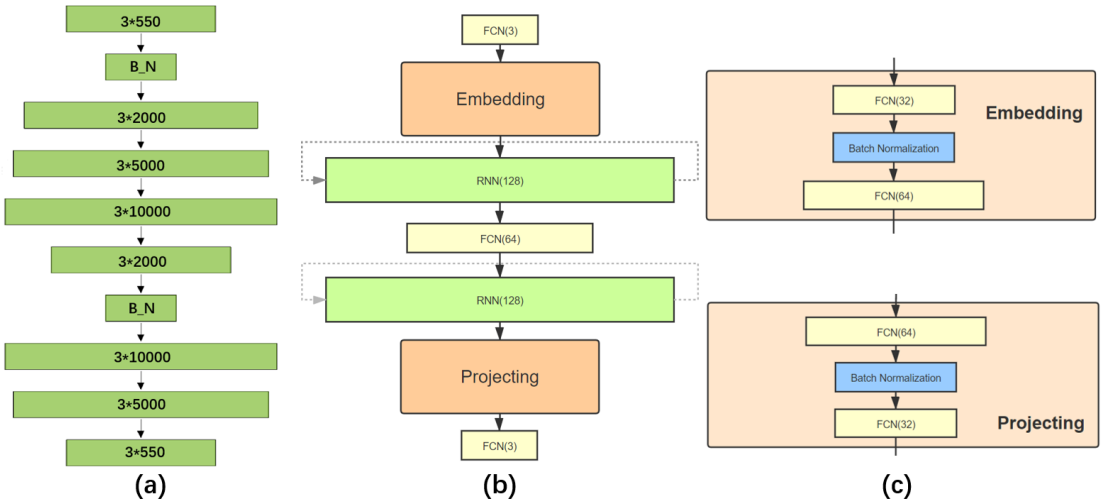

Figure 5.1: **(a)**: Structure of deep fully connected autoencoder. **(b)**: The structure of deep autoencoder. **(c)**: the detailed structure in embedding and projecting module.

5.1.3 Undercomplete Bidirectional Autoencoder

Aiming to improve the regularization capacity of the model, we built the undercomplete autoencoder and bidirectional autoencoder. In the undercomplete autoencoder, there is a bottle neck structure in the middle of the whole model, this action might push the model to learn a compact expression of the dataset, and the anomalous data is hardly to kept in the compact expression.

In autoencoders we wish the model to learn how the position information in back contribute to position data in front part. So we applied bidirectional reccurent unit instead of the normal unit as well.

Figure 5.2 demonstrated the structure of the two models, we could see that the overall structure is the same, the inner structure of the embedding and projecting module are the same. In the middle we could see there are more layers to compact the data into lower dimensional format.

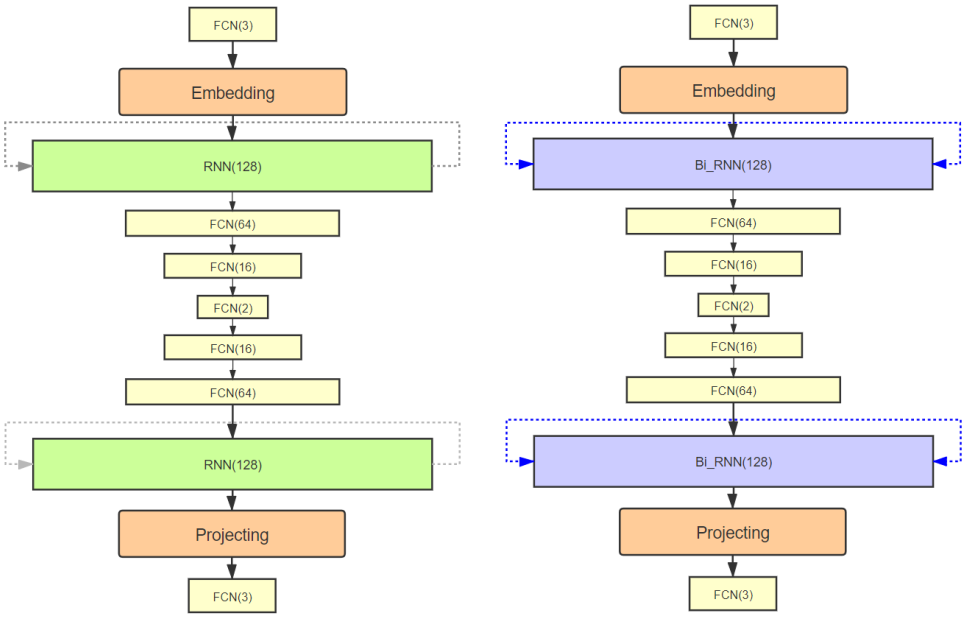

Figure 5.2: left:. Structure of undercomplete autoencoder model. **right:** Structure of bidirectional autoencoder model.

5.2 Seq2seq models

5.2.1 Sequence to sequence model

The structure of models built above is not good at compact expression since the model process one position data each time, the dimensionality is three, leaves

it small space to learn the compact expression. As a result we tried to train the Seq2Seq model and Seq2Seq model with attention mechanism. Unlike the autoencoders mentioned above, the seq2seq need to process all the position data of a streamline, pass the context vector to the decoder for reconstruction. So the context vector could be considered as a the compact expression of one curve , which could be defined in arbitrary length.

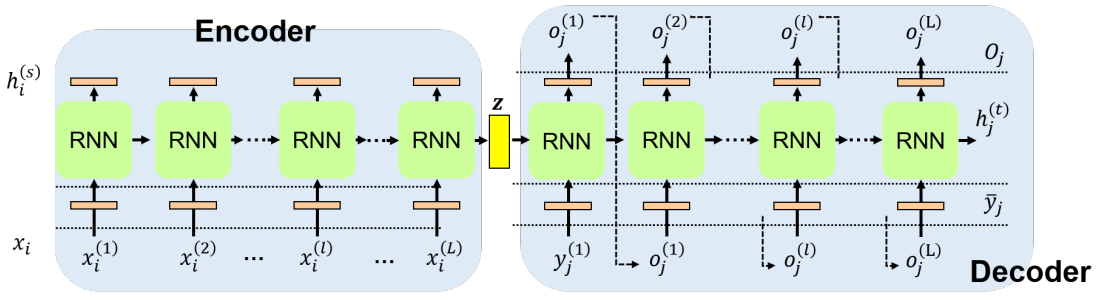

Figure 5.3: Structure of sequence to sequence model, orange block: embedding and projecting module, **z**: context vector.

The structure of the seq2seq model is demonstrated in figure 5.3, there also exist the embedding and projecting module. For the encoder and the decoder, the number of sub-RNN models is the number of vertexes of a streamline, which is 550 in our case. In experiments, the training speed is rather slow compared to methods above.

5.2.2 Attention mechanism

Adding the attention mechanisms will highly improve the capacity of models since it aims to found which parts of input contribute more for the reconstructing procedure. The structure of the seq2seq model with attention mechanisms is demonstrated in figure 5.4, the orange modules of embedding and projecting before and after RNN is the same as seq2seq. Besides, we followed equation 3.14-3.18 to build the attention mechanisms, the added attention mechanism is depicted in blue dotted line. The gray square indicates the score calculation, which is equation 3.14,3.16 or equation 3.15,3.16 depend on choice, the yellow block indicate equation 3.17.

The added structure calculate the score from outputs of encoder and hidden values of that state. Then we pass the score to a softmax function to generate the attention situation as the context vector. The context vector will concatenate with the input of next state and they are passed to the recurrent layer of next state.

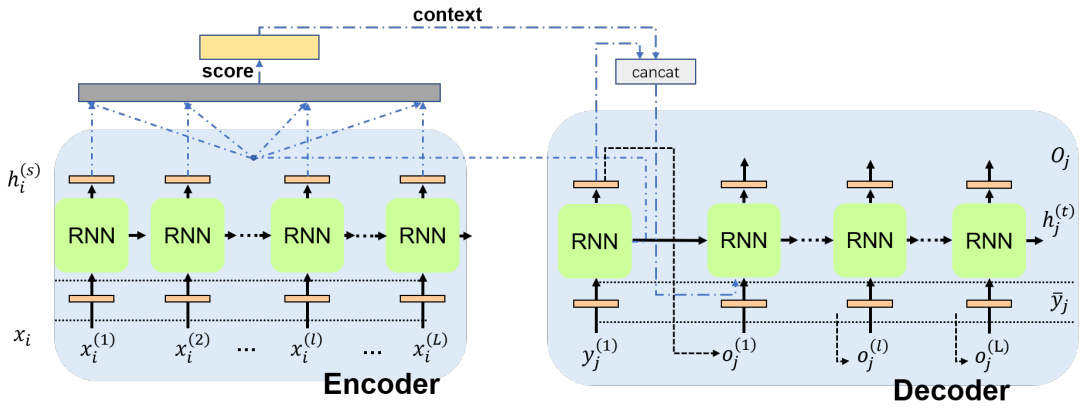

Figure 5.4: Structure of sequence to sequence model with attention mechanisms.

5.2.3 Ensemble Seq2seq

Another effective and popular method to boost the regulation of a deep learning model is ensemble learning. Inspired by the paper[16], we built the ensemble seq2seq model. The structure of the ensemble seq2seq model is demonstrated in figure5.5. There are N seq2seq models, and for each model, the dropout module is added to reduce the learning capacity. As for the context vectors, we concatenate context vectors from the number of N encoders and pass is to a fully connected layer to squeeze it as a ensemble context vector and pass it to the number of N decoders. Since the dropout part is selected randomly, these added dropout operation would increase the generation expression of the ensemble context. In the step of reconstructing, we would take the mean values of the N decoders as the reconstruction.

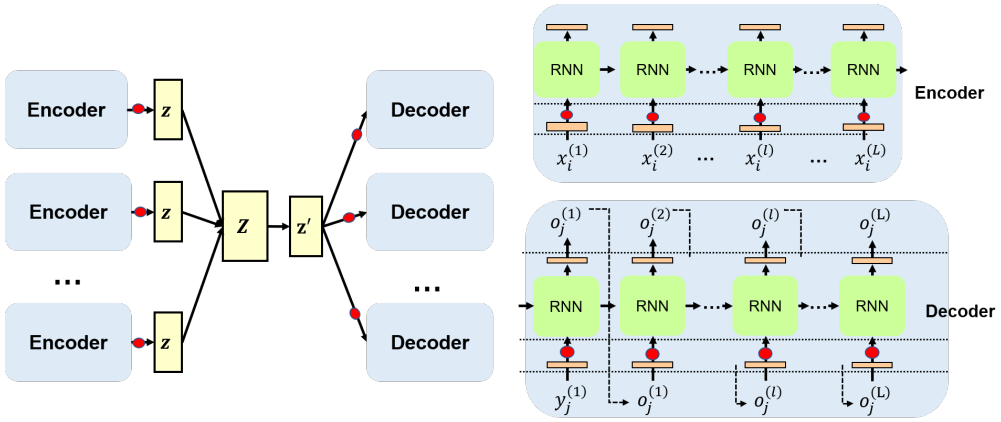

Figure 5.5: Structure of sequence to sequence model, orange block: embedding and projecting module, red dot: dropout operation. **z**: context vector.

The original seq2seq model enables the model output length varies. Unfortunately this feature is only available to discrete data, since the start and end sign of a sentence are embedded in the discrete space. However in this project we are dealing with continuous data, there are methods to tag these signs, but not as effective as in discrete space.

Precisely speaking, the figure 3.4,5.4,5.5 depicted the procedure of generating reconstruction. For a streamline $\mathcal{S}_i = \left\langle \mathbf{s}_i^1, \mathbf{s}_i^2, ..., \mathbf{s}_i^L \right\rangle$ all these seq2seq based models are trained and generating reconstruction as follow: when training, the decoder takes $\mathbf{s}_i^1, \mathbf{s}_i^2, ..., \mathbf{s}_i^L$ as inputs, when generating the reconstruction, the decoder only takes $\mathbf{s}_i^1$ as the initial input, after that step, it takes the decoder output $\mathbf{o}_i^l$ as the decoder input in state $l + 1$.

6 Experiments

6.1 Manually Anomaly Detection

We followed Algorithm 2 to mark the anomalous streamlines and normal streamlines. In that algorithm three methods are detecting anomalous streamlines separately and union the results as final anomalous set. We will demonstrate and analyze the detected results of these three methods separately to evaluate the pros and cons.

Cluster Confidence Index. For each subject, we calculate the Cluster Confidence Index(CCI) of each streamline and classify that a streamline with CCI above $CCT_{thre} = 2$ as a normal one.

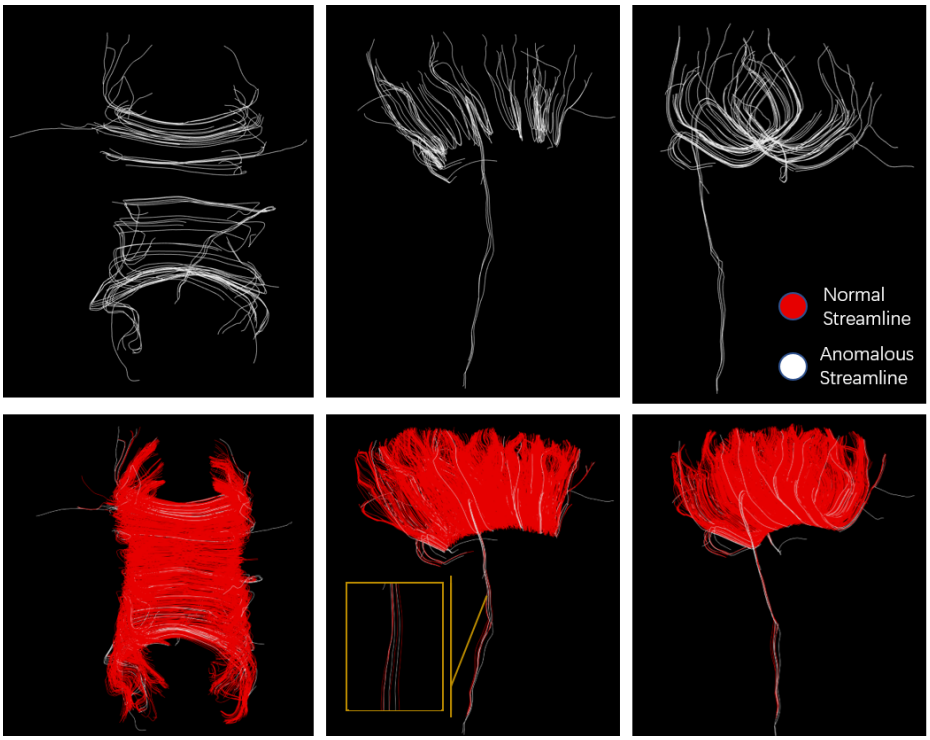

Figure 6.1: Result of method CCI in three directions. **Upside**: detected anomalous streamline. **Downside**: anomalous streamline within in the bundle. Subject ID:144731

Figure 6.1 demonstrated the result of threholding the CCI value, red stands for normal streamline and white stands for anomalous ones. From the image we could tell this method detected some of the anomalous streamlines but the results also contain some false positive. The bundle of should have a roughly U shape, the long streamlines growing below this is U shape is definitely anomalous ones, but the CCI method did not detect them all. The yellow box in the second image downside showed the zoomed in details, four of seven lines are false positive. False positive here means the fiber should be anomaly but marked as normal one.

In fact, the CCI values varies in subjects, setting a strict threshold for anomaly detecting is not a good idea. Figure 6.2 showed that the CCI value distribution differs in subjects. For subject 132118, the number of low CCI streamlines is much less than the subject 144731, this should means that the bundle of subject 132118 have a better performance in streamline arrangement, while the tractography did not prove that, we even see that the streamlines are more messy and disordered than the subject 144731.

besides there are control parameters within the calculation, see equation4.1, setting these parameters also takes energy and reduce the robustness of this method.

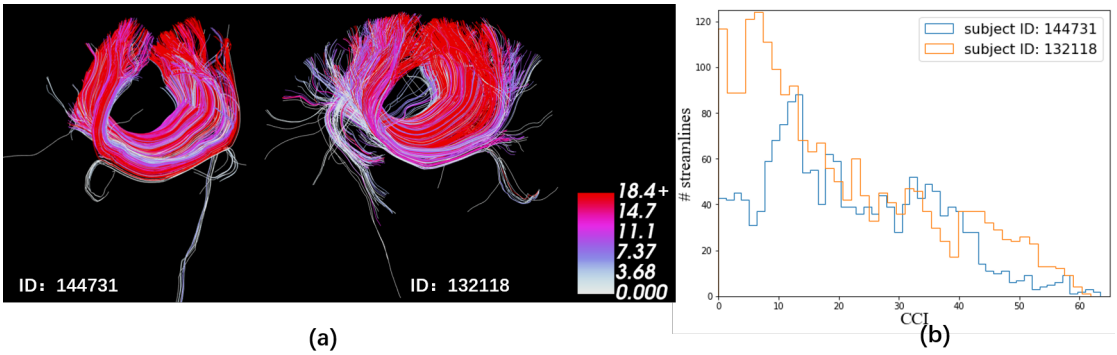

Figure 6.2: **(a)**: tractgraphy of two subjects **(b)**: Histogram of CCI values of the two subjects in (a).

Distance Score. The idea of using Distance score is to detect the streamline out of the bundle region, a high distance score implies that the streamline is farther away from the region. Figure 6.3 demonstrated the detected result in the whole bundle of subject 144731. The yellow marked part is a group of anomalous streamlines, and this group is all detected correctly, much better than the method of CCI. However, it seems that the group of streamlines marked in blue is more likely to be false positive. Overall this method is just used for detecting the specific subset of anomalous streamlines. Besides the bundle volume mask is an approximate one, the result is far away from good.

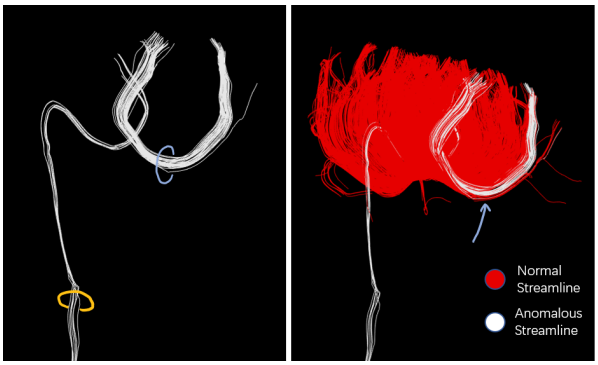

Figure 6.3: Method: distance score. **left**: detected anomalous streamline. **right:** anomalous streamline within in the bundle.

Isolation Forest. Figure 6.4 demonstrated the detecting result under method isolation forest. We marked three typical features in the result. The yellow group is detected correctly, as good as the distance score, better than CCI. For the blue pointed part, the group is very similar to the blue pointed part in figure 6.3, results of distance score. As for the green marked group, these streamlines are not detected by CCI and distance score, we could observe that these streamlines have a shorter length. Since the user could not decide to mark the short curves or not, this is not a very good feature.

Fusion Strategy. From algorithm 2, we could see we set a strict threshold for the three methods and took the union set of the detected anomalous fibers. In fact, the original idea was to set slack thresholds and took the majority voting. However, the intersection of detected anomalous fibers did not improve too much when slacking the thresholds, so we changed our plan. This also implies that the manually generated label set is not an ideal one or the methods only work in detecting one specific kind of anomalous fibers.

6.2 Evaluation Methods

We would like to see the performance of the deep autoencoder models in anomalous streamline detecting, the idea is, for each streamline, pass the position data of streamlines to the model and let the autoencoder reconstruct the streamline, compute the mean square error, a large MSE value implies a high probability to be an anomalous streamline. To evaluate the performance of these binary results, a normal way is to draw the precision and recall curves.

Figure 6.5 plotted thee F1 score in normal fibers detecting supervised by the manually generated label set under different models. From the plot, we could see

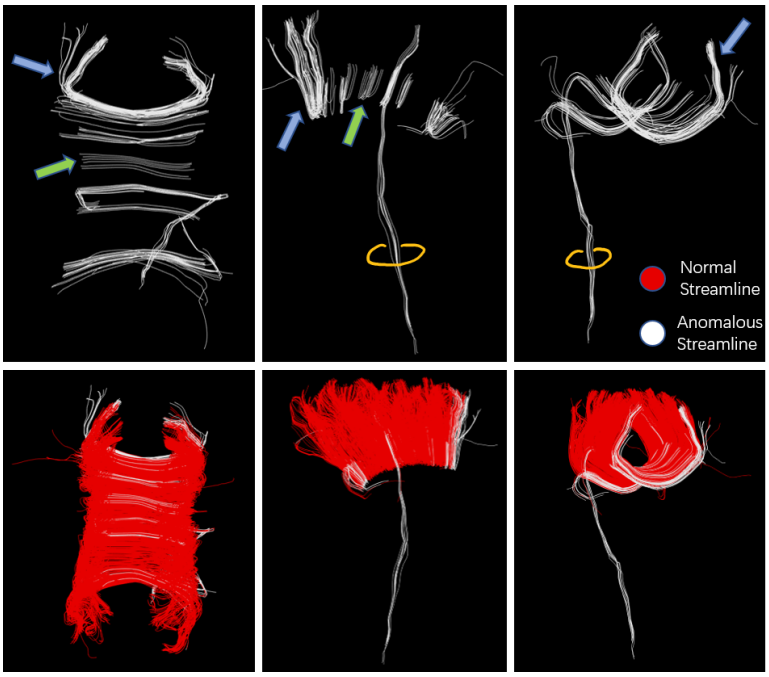

Figure 6.4: Result of method isolation forest. **Upside**: detected anomalous streamline. **Downside**: anomalous streamline within in the bundle.

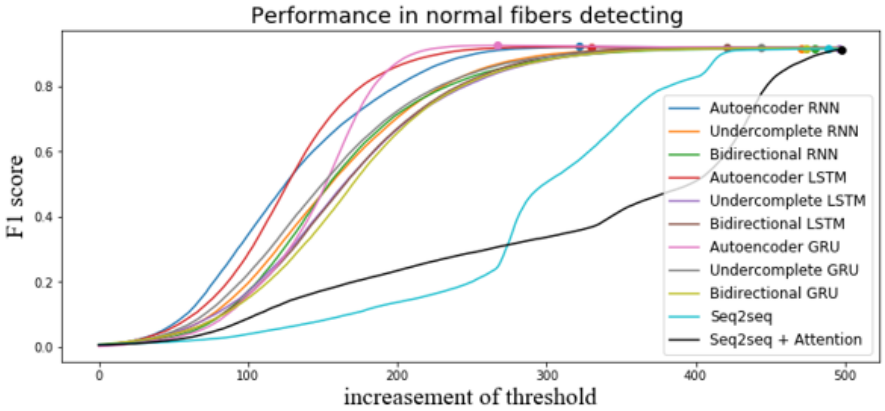

Figure 6.5: Performance of normal fibers detecting under different models.

most of the models except seq2seq based models have a good performance. For a model, if the F1 score curve is generally above another, we could say this model has a better performance than another.

In the referencing manually generated label set, the anomalous fibers only occupy 15%, in figure 6.5 , we could hardly compare some models since the basis value is too large. As a result, we applied the analysis scheme in anomaly detection instead. The definition of the terminology is described in the table 6.1, for instance, true positive here means the correctly detected anomalous fibers. The way to compute precision and recall is shown in equation 6.1,6.2. The F Score is a measurement of performance considering both precision and recall, the F1 score is a situation that $\beta = 1$, means precision and recall have the same weight in measuring, see equation 6.3,6.4.

Table 6.1: Terminology of anomaly detecting

manual label model prediction	**anomalous**	**normal**
anomalous	TP (true positive)	FP (false positive)
normal	FN (false negative)	TN (true negative)

$$Precision = \frac{TP}{TP + FP} \tag{6.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{6.2}$$

$$F\_Score = \left(1 + \beta^2\right) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot (Precision + Recall)} \tag{6.3}$$

$$F1\_Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{6.4}$$

6.3 Deep Autoencoders

6.3.1 Fully Connected Autoencoder

The figure 6.6 demonstrated the anomaly detection result with the fully connected autoencoder. From the results we found the capacity of this model is low, in figure 6.6**(c)**, the reconstruction MSE is in a quite high range, from 33 to 403. And in figure 6.6**(a)(b)**, the detection results of two randomly selected subjects are not good, for example the curve below the main body are anomaly but not detected,

the part in the right of main body are normal curves but was marked as anomaly,

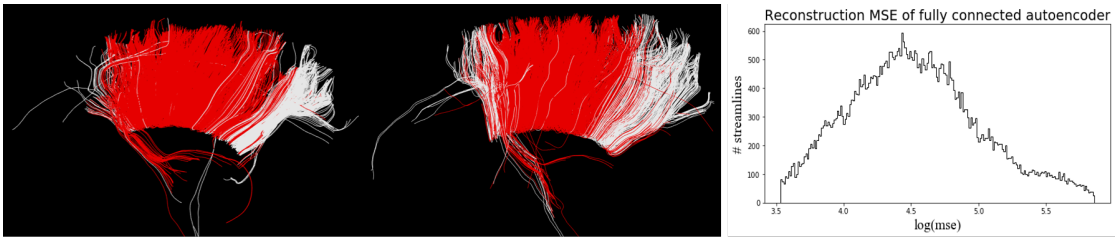

Figure 6.6: Anomaly detection result with the fully connected autoencoder. **(a)** Anomaly detection for subject 161630. **(b)** Anomaly detection for subject 156031. **(c)**Histogram of reconstruction MSE of all subjects in test list.

6.3.2 Deep Autoencoder Models

The table 6.2 listed the configuration of the deep autoencoder. We set a strict threshold MSE_{thre} that for a fiber if the reconstructing MSE is larger than the threshold, it will be marked as an anomalous fiber.

We described the performance of models by analyzing anomaly detecting capacity. The figure 6.7, the first three subplots demonstrated the anomaly detecting performance of each model, in each subplot, the histogram of mean square error in reconstructing fibers under the model is depicted in brown lines, the precision, recall and F1 score values under thresholds are calculated and depicted in green, red, blue, the highest F1 score are also marked in a blue dot. The last subplot of figure 6.7 compares the F1 score performance in three models.

Table 6.2: Configuration of deep autoencoder models

Batch Size	**50**	Activate functon	**Leaky Relu**
Learning Rate	**0.001**	recurrent activation	**Sigmoid**
Loss Function	**MSE**	Recurrent Type	**RNN, LSTM ,GRU**
Optimizer	**Adam**	Initialization	**glorot_uniform**
# Recurrent Unit	**128**		

From the figure 6.7 we could find that the deep autoencoder with GRU recurrent has a better performance, in another word, this model have the most similar

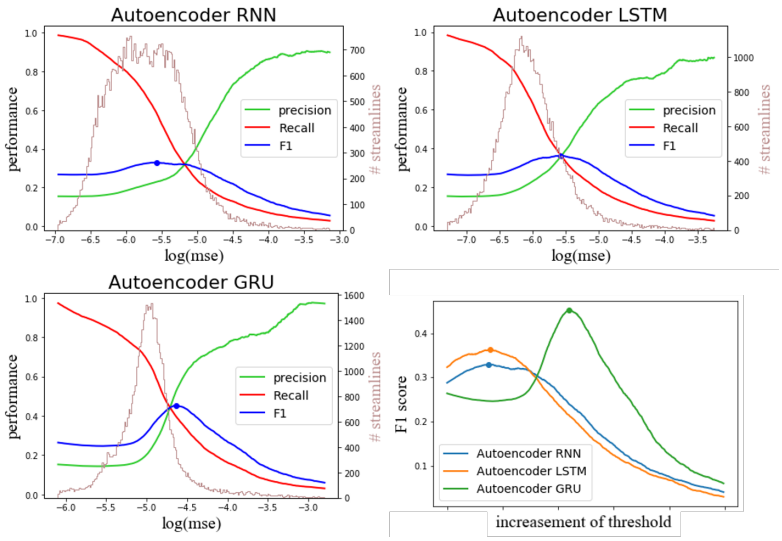

Figure 6.7: Performance of anomaly detecting of three models.

result with our manually generated label set. Besides, we could set the threshold $MSE_{thre} = argmax_{MSE}(F1)$ for better performing.

Figure 6.8 compared the result of our model result with the former manually detected result. They are different in two aspects.

1. The yellow circled group. In figure 6.8, we select a group of fibers in yellow, principally they should be considered as anomalous fibers. In the manually detected result, some fibers in this group are still considered as normal ones, this is due to this method could not be detected by three of the manual detecting methods, for distance sore, this group is very close to the mask volume region, and the fibers are parallel to the mask volume, so the distance score would not be very large. As in the CCI method, this anomalous fibers in same shape gathered in a group, an anomalous fiber could find its companion in neighbor area, thus get a high CCI value.

2. Short fibers. The deep learning model did not consider the short fibers with in the U shape bundle as anomalous ones.

6.3.3 Undercomplete Bidirectional Autoencoder

Followed by the same idea, we built the undercomplete autoencoder and bidirectional autoencoder with the same configuration as 6.2.

The performance of **undercomplete autoencode**r in anomaly detecting is depicted in figure 6.9. Around the three types of recurrent units, RNN showed a better performance. However, through the distributions of reconstructing MSE, if

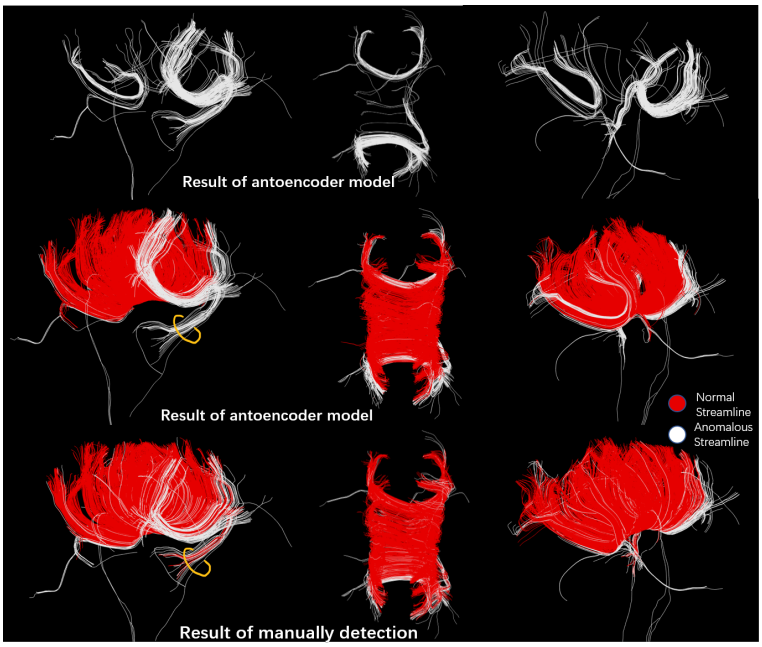

Figure 6.8: Anomaly detecting results of deep autoencoder GRU compared with the manually generated set, subject ID:161630.

we set the threshold at $argmax_{MSE} = exp(-0.41)$, there will be too much fibers considered as outliers, we wish the threshold at the beginning of tail part, so we slack it to $exp(-0.1)$.

Figure 6.10 showed the result of undercomplete autoencoder RNN compared with the manually generated label of a randomly selected subject. From the result we could see that neither of the two techniques could correctly detect all the anomalous fibers under the U shape bundle. Both of them contains some fiber detected as anomaly while are normal ones in fact. In the manually generated label result, the blue arrow pointed anomalous fiber group, might be caused by high distance score as demonstrated in figure 6.4, blue arrow pointed group.

As for the **bidirectional autoencoder**, we only trained in two types of recurrent units, since bidirectional RNN takes quite longer time to train. From figure 6.11 we could see the GRU bidirectional autoencoder has a slightly better performance. For the same reason as in undercomplete autoencoder, the threshold is slacked to $exp(2.2)$

From figure 6.12 we could observe that in yellow circled fiber groups the autoencoder model did better job in anomaly detecting, since most of fibers in these group are marked as outlier. While in the green circled group, neither of them detected the anomalous fibers.

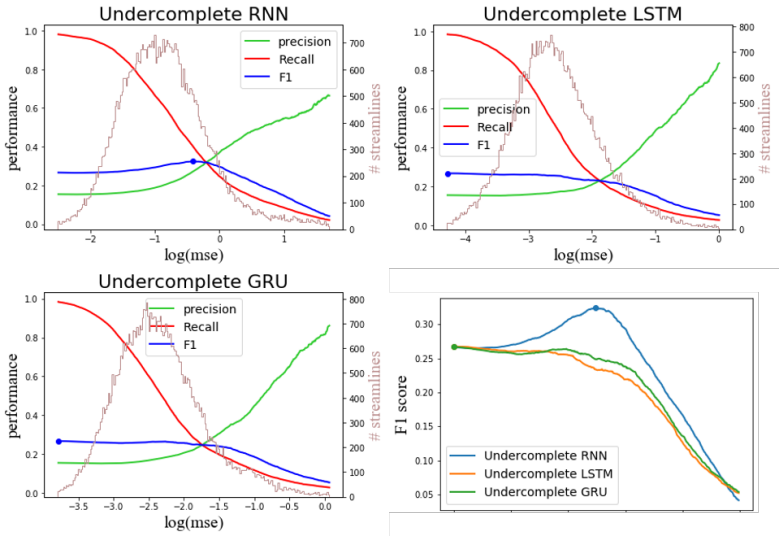

Figure 6.9: Performance of anomaly detecting of three Undercomplete Autoencoder models.

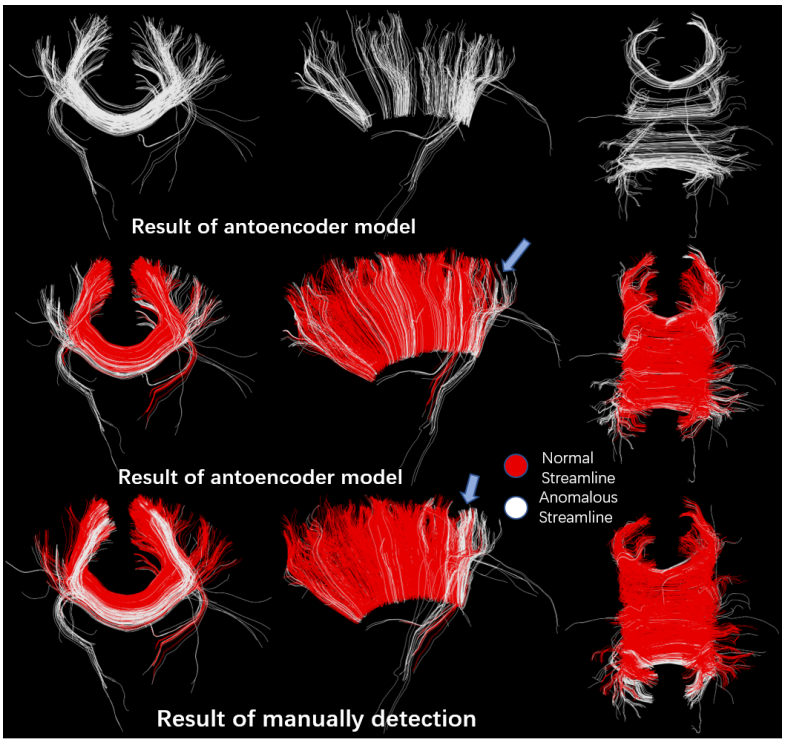

Figure 6.10: Anomaly detecting results of Undercomplete autoencoder RNN compared with the manually generated set, subject ID:156031.

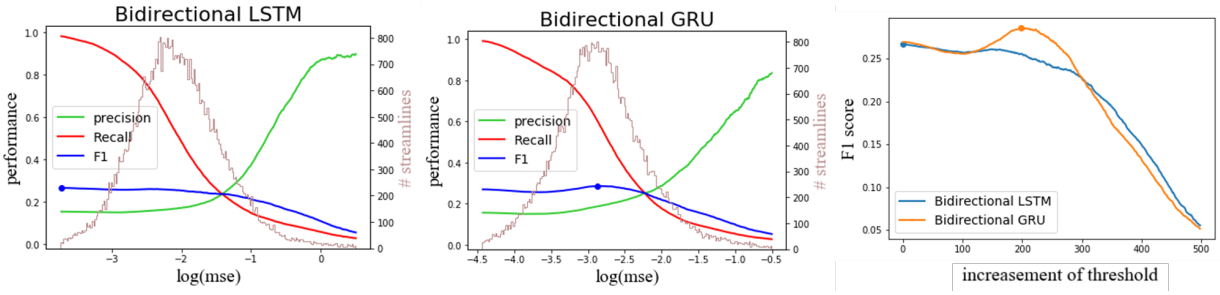

Figure 6.11: Performance of anomaly detecting of two Bidirectional Autoencoder models.

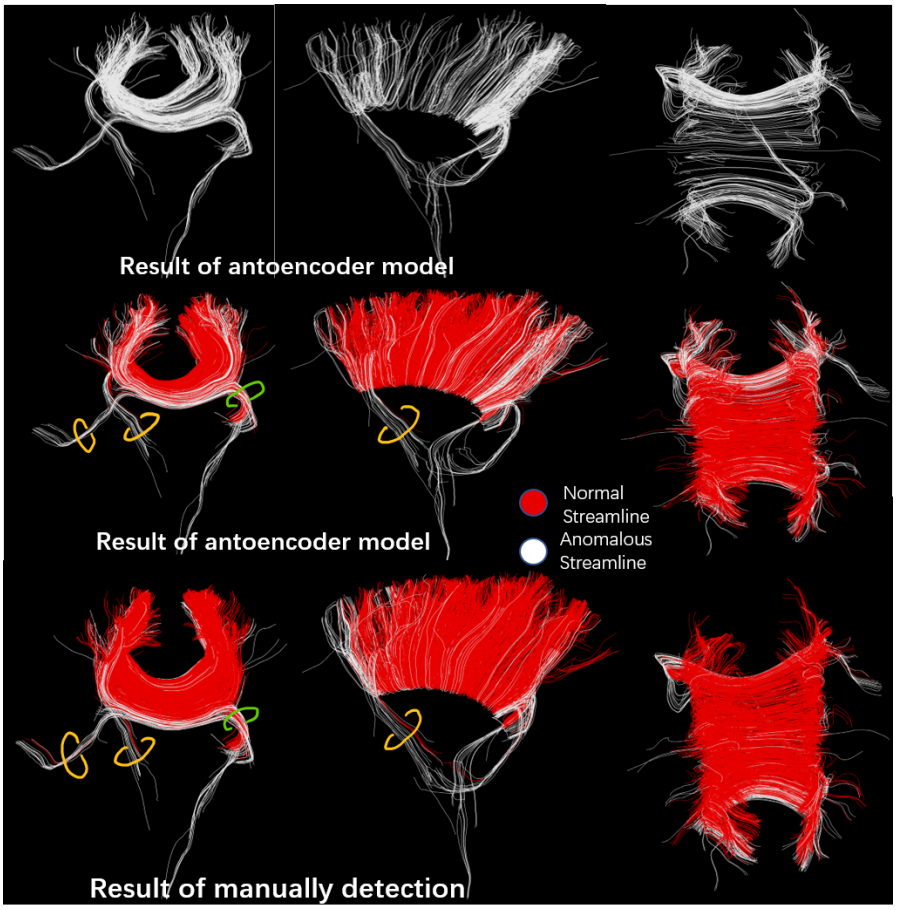

Figure 6.12: Anomaly detecting results of Bidirectional autoencoder GUR compared with the manually generated set, subject ID:156031.

Summery so far The figure 6.13 plotted the performance of anomaly detecting under all models. From the plot, we could roughly conclude that the deep autoencoder models have a better performance than undercomplete and autoencoder bidirectional autoencoder.

This conclusion is stated under the evaluation scheme supervised by the manually generated labels. Compared with the result of manually detection, all the deep autoencoders have better performance in anomaly detection.

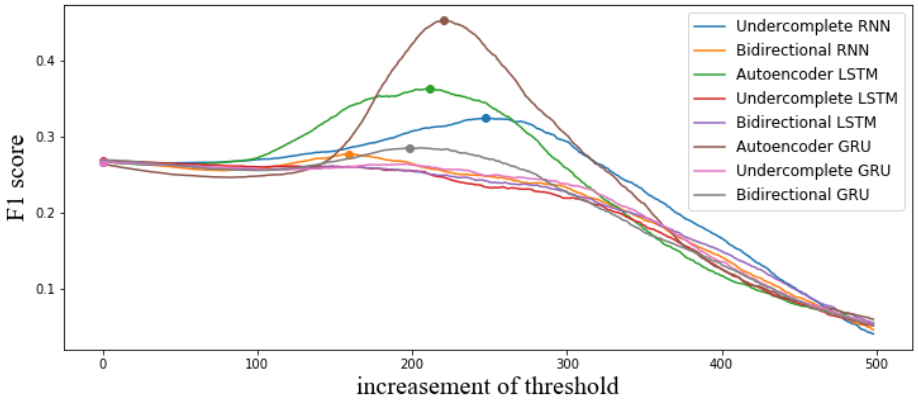

Figure 6.13: Performance of anomaly detecting of all models.

Until here we have presented performance of all the subtypes of deep Autoencoder. Form the demonstrated detection results and the F1 score plot in figure 6.13, we found the deep autoencoder with GRU recurrent units has the best performance. Among all the subjects in test list, we found the fiber bundle of subject 156031 is a very difficult sample since the fibers are fluffy at the border, and there are a lot short anomalous fibers under the U shape but very close to the main body.

Figure 6.14 demonstrated the reconstruction MSE of each fiber under deep autoencoder GRU in subject 156031. The ambiguity range of MSE is between $exp(-4.5)$ to $exp(-3.5)$, from the image we could see the curves of MSE higher than $exp(-3.5)$ are in white, standing for detected anomaly, the curves of MSE lower than $exp(-3.5)$ are in orange, standing for detected normal curves.

From chapter 4.2 we learned there are four types of anomalous curves: 1.Fibers transit out of region of interest; 2. Abnormal trajectory within the target tracts region; 3.Short fibers; 4.Fibers at border of target tracts region. Type 1 anomaly are detected correctly if we observe the curves below the U shape, they are in color of white and blue. For type 2 anomaly we did no find counterexample here. All of the deep autoencoder could not detecte the type 3 anomaly since they are embedded in the bundle volume, but we could filter them our simply by calculating

length of fibers. As for the type 4 anomaly, we could see the ambiguity parts are in color from pink to green while the main body is in orange, so the users could decide the threshold themselves. In conclusion the deep autoencoder did really good job in anomalous curves detection.

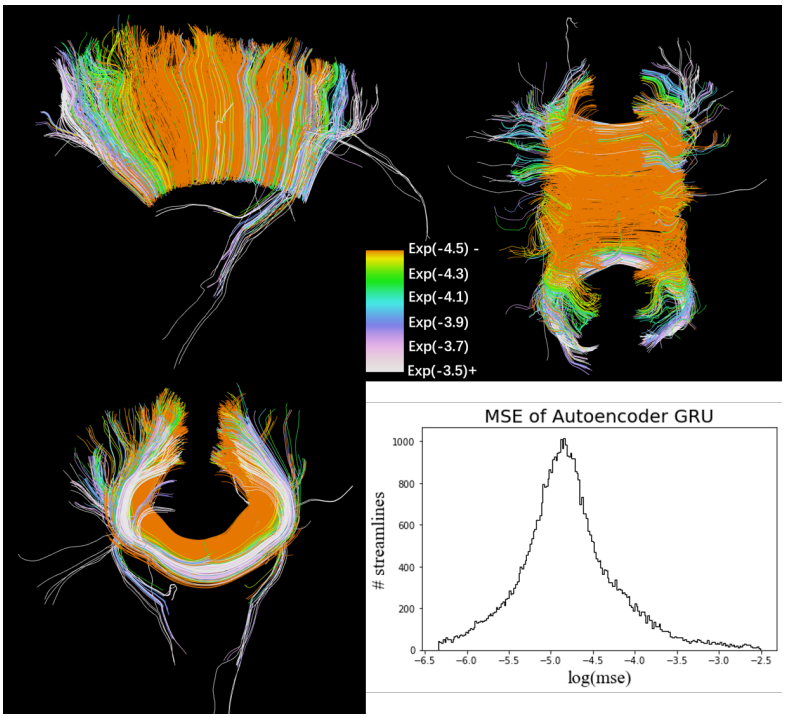

Figure 6.14: The reconstruction MSE of each fiber under deep autoencoder GRU, subject ID: 156031.

6.4 Compact Expression

In the models described above, they all have small training error and reconstructing error, we even need to take the logarithm of reconstructing MSE to rescale and expand the interval for further analysis. To get a more generalized result, we would like to use models to learn a more compacted and more generalized expression of the input data. A Sequence to sequence based model enables it to compress the input into arbitrary size, so we also trained the seq2seq, seq2seq+Attention, ensemble Seq2seq models. The configuration is listed in table 6.3, the batch size for the seq2seq, seq2seq with Attention, ensemble Seq2seq model are 64,17,80, set to maximize the GPU memory to get full speed training.

However, the performance of these seq2seq based models are not satisfying. In figure 6.15 we found the the models showed a very strange trend in precision, recall

and F1 score, after setting the threshold at $exp(7), exp(1)$ the detected results showed that the model could not to our task at all.

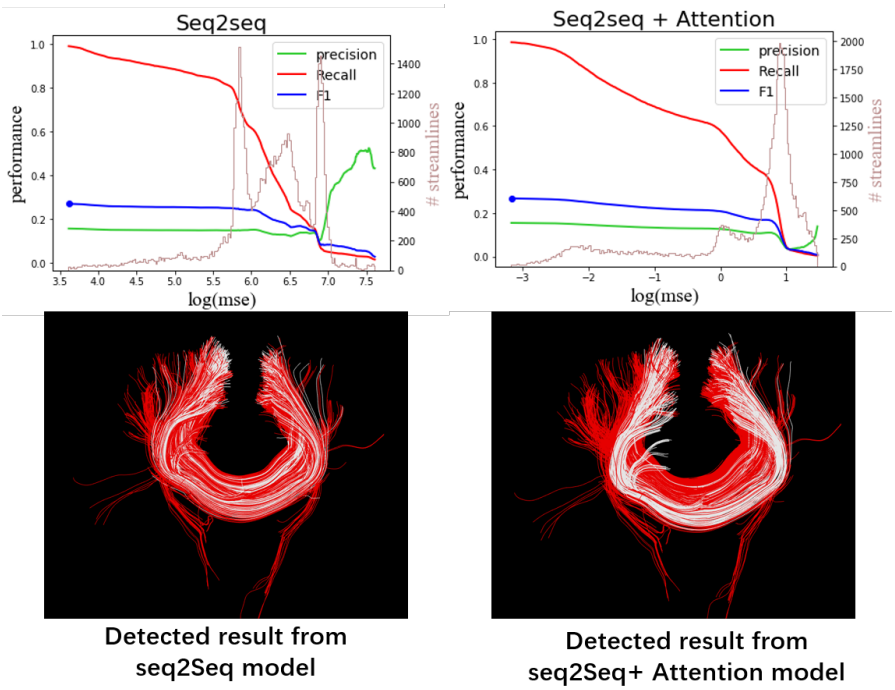

Figure 6.15: Up: Performance of anomaly detecting of two seq2seq based models. **Down:** Detected result of two seq2seq based models.

Table 6.3: configuration of seq2seq based models

Batch Size	**64/17/80**	Activate functon	**Leaky Relu**
Learning Rate	**0.001**	recurrent activation	**sigmoid**
Loss Function	**MSE**	Recurrent Type	**GRU**
Optimizer	**Adam**	Initialization	**glorot_uniform**
# Recurrent Unit	**128**		

From the reconstructing MSE distributions, we could see the MSE is in a very large cardinal number, which means the models are in an underfitting situation. At the same time, the training history showed the training loss could not further go down. To figure out what might go wrong and what could we do to improve the performance, we did some research and compared with the working NLP mode configurations, the possible reasons are as follow:

1. Too small recurrent units size. In table 6.3 the size of recurrent units is only 128, which means for a Seq2seq model, the input is encoded from 500×3 into 128, the encoded size is two small for reconstructing. In a experiment, we change to the max possible value, 500, the training loss remains at the same level.

2. Too long sequence length. In our model, the sequence length is 550, while in NLP tasks the sequence length is no longer 20. We tried subsampled the length to 200, but the improvement of the model is still not too clear.

3. Too small embedding and projecting size, in the NLP task the input should be embedded to the size of the number of words, while in our task, the embedding size is only 64.

Overall the seq2seq is originated for the NLP task to process discrete data, we believe it could work for our task, but it will take time to find good configurations and compensate mechanism. Table 6.4 tells us that the seq2seq based models take much longer time to train.

As for compact expression, this is a trade-off with high reconstruction MSE, from our experiments, we found high reconstruction MSE is not good for anomalous detecting since this will cause MSE distributions differs too much among subjects, as a result, not good for find threshold.

6.5 New Seq2seq Models

Since the seq2seq based models could not finish this task, we decided to have a new start, we removed all the embedding and projection modules of the seq2seq based models, for comparison, these simple models are called the **new** seq2seq based models. We did multiple experiments in the new seq2seq model and found the key part that limits the train speed and capacity: the number of recurrent units and number of vertexes each sequence. So we down-sampled the number of vertexes of streamlines from 550 to 50 and calculated the train errors and validation errors with different numbers of recurrent units, the results are plotted in figure 6.16.

From the figure 6.16 we could tell why our old models not working, in the previous configuration, the number of vertexes of streamlines is 550, and the number of recurrent units is 128, the $\frac{\text{of units}}{\text{of vertexes}}$is only 0.23, in this case, the training error could reach to a very low level, but the validation error remains in high level, this happens because the training and generating reconstruction are different.

Because of the computation resource limitation, we built the new seq2seq based models with the dataset that the number of vertexes is down-sampled to 50, the

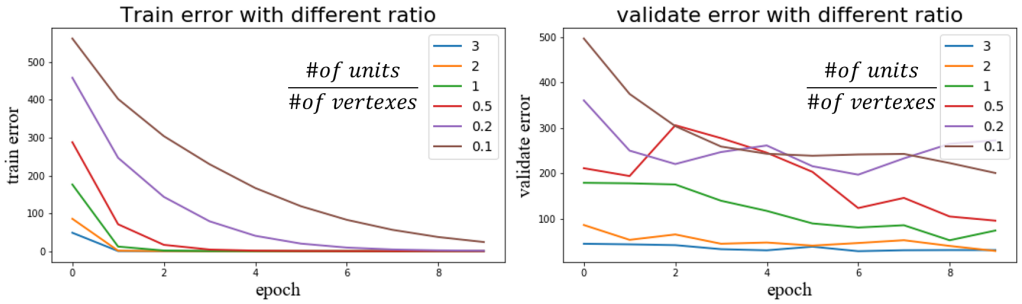

Figure 6.16: Train errors and validation errors trend with different ratios when number of vertexes each fiber is 50.

embedding and projection modules are totally removed. As for the number of recurrent units, we set it to 150 in the new seq2seq and the new seq2seqAttention. In the ensemble seq2seq, the number of vertexes is 20, the number of recurrent units is 10, the number of sub-autoencoder N is 5 the dropout is 0.25.

The figure 6.17 depicted the anomaly detection performance of the three models. From the reconstruction MSE histogram, we found the models are working with really high MSE, this also implies that there is much space to increase the capacity. The figure 6.18 demonstrated the anomaly detection result of the three new models. Combined with the figure 6.176.18, we found the new seq2seqAttention has the best performance, the four types of anomaly except the short fibers are all detected. The new seq2seq model met a problem in the border anomaly detection, the group pointed by the red arrow could hardly be removed when restricting the threshold. The new seq2seq model is the worst here, it could even detect the group pointed by the yellow arrow, the type one anomaly, fibers transit outside of ROI. This might be due to the very low number of vertexes in curves and no time to adjust the hyper-parameters.

6.6 Summary

Estimated features for the anomalous detection From the experiments we did, we found there are two estimated features for the deep autoencoder.

1. Bell shape of the reconstruction MSE histogram. We listed 13 types of models in this thesis, and we could find the performance is not directly related to the range of the MSE but the shape of the histogram. One example is the seq2seq based models, their reconstruction MSE is very high due to a special way of reconstruction, but they could still finish the task. A counterexample is demonstrated in figure 6.19 , both of the two models have the same MSE

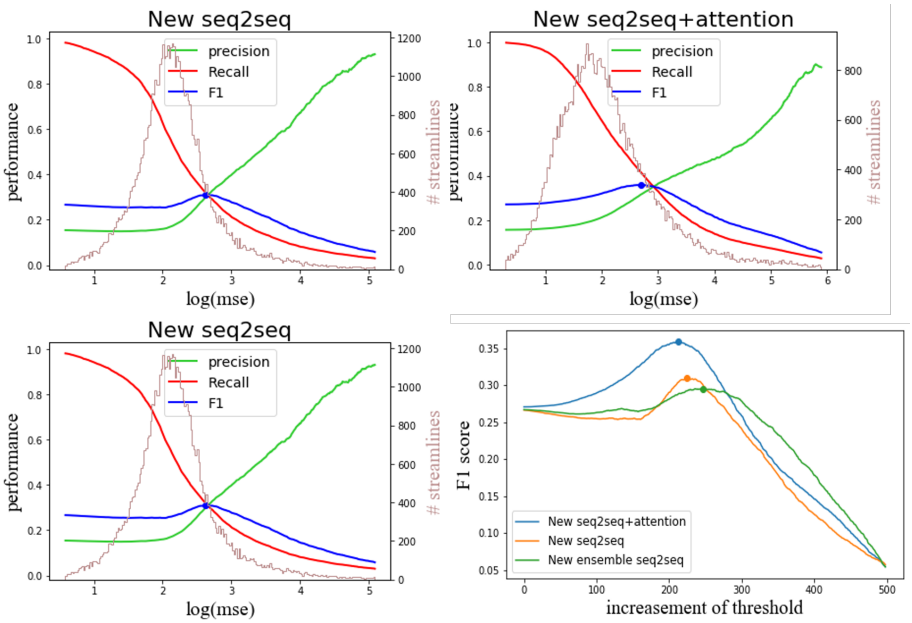

Figure 6.17: Performance of anomaly detecting of three new seq2seq models.

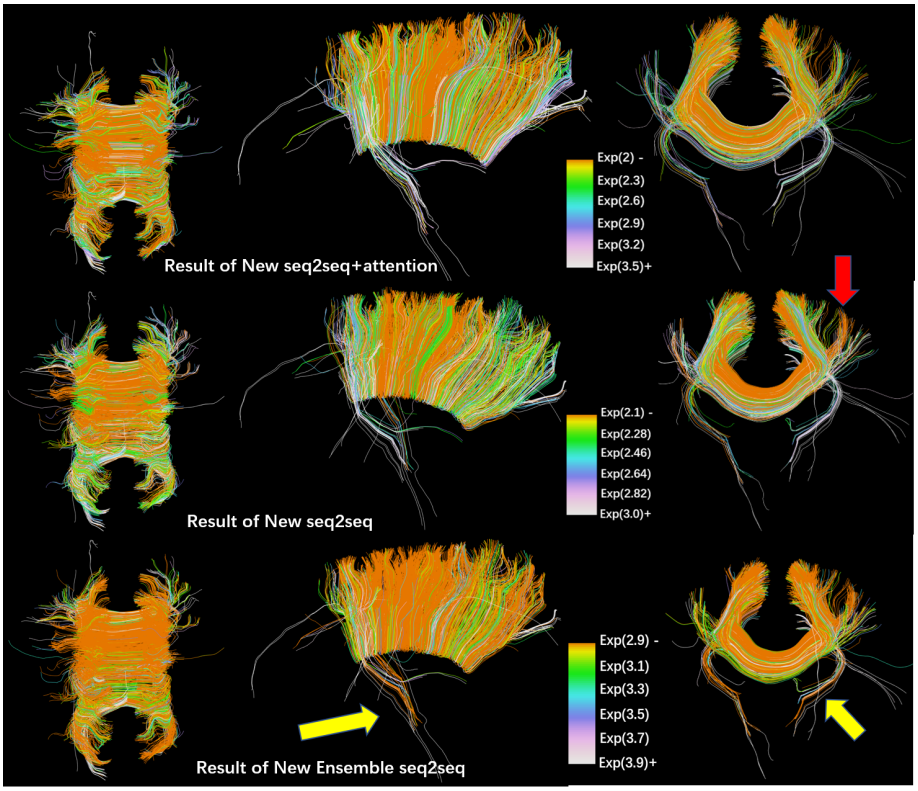

Figure 6.18: Train errors and validation errors trend with different ratios when number of vertexes each fiber is 50.

range but performance different, the model one could not finish the anomaly detection. A non-bell shape histogram shape appeared several times in deep autoencoder LSTM and seq2seq+Attention (both old and new).

2. Invariable MSE histogram trend of different subjects. In this task, since we calculate the reconstruction MSE and set a threshold to it, we would like that the threshold could be a global one, and the user does not need to adapt it in different subjects. A model generates a similar reconstruction MSE histogram trend among test subjects is more preferred. From the listed models, each of the seq2seq based models did perfectly in this feature, a deep autoencoder with the same reconstruction MSE also did a good job in it.

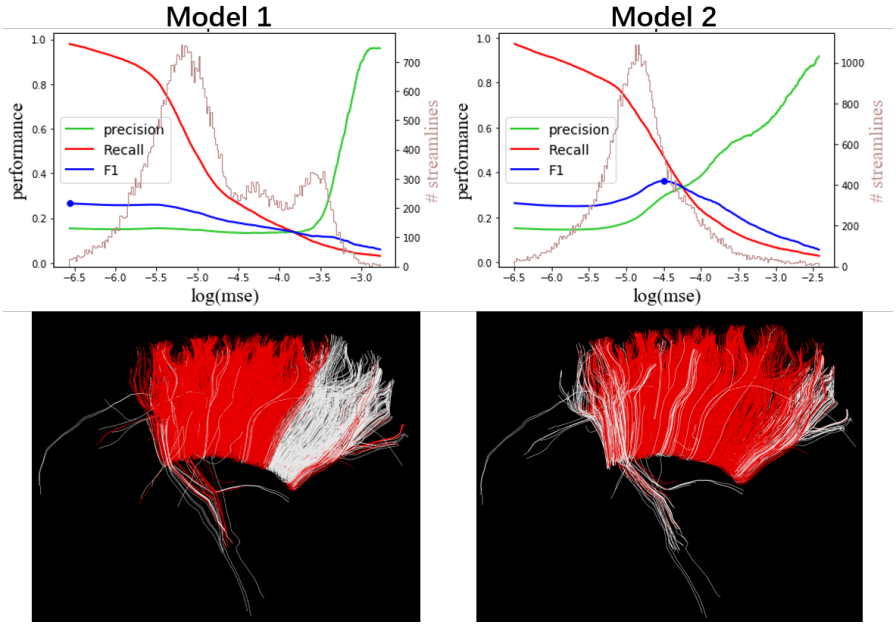

Figure 6.19: Results of two models with the same range of reconstruction MSE but different shape of histogram.

Estimated Recurrent types Among the three recurrent types in this task, the GRU is more preferred. LSTM sometimes has a non-bell shape of reconstruction MSE histogram, whereas the recurrent neurons took a longer time to train than the other two.

Deep Autoencoder Fusion We did experiments to combine the good detection results to boost the performance. From figure 6.13, we selected three models: Autoencoder RNN, Autoencoder LSTM, Autoencoder RNN as candidates. Since

the reconstruction MSE range differs in models, we set the percentile value of MSE in each histogram as the threshold, generated the detection. The final result is generated by majority voting of the three results. This method did a better job of keeping normal fibers in the main body. Figure 6.20 demonstrated on the result of the anomaly removal by the threshold percentile 80.

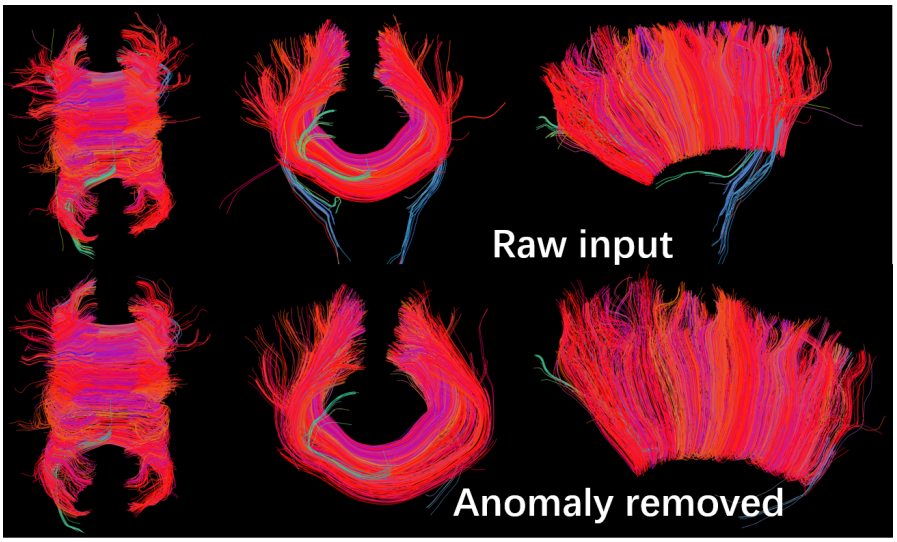

Figure 6.20: Results of anomaly removal by deep autoencoders fused method, compared with the raw input. The color is coded only for position .

6.7 Implementation

We implemented this project with python, all the preprocessing and postprocessing steps are finished automatically, for visualizing streamline the library nibabel is used. We built the deep autoencoder models with tensorflow 2.2, to boost the training speed, the package tensorflow-addons 0.11.2 was utilized. in table 6.4 we listed the GPU type, training time each epoch, GPU memory, and batch size for each model.

Table 6.4: Implementation details

model	GPU	time per epoch	memo	B_S
Deep Autoencoder	GTX 780	574s	3Gb	50
Undercomplete	GTX 780	586s	3Gb	50
Bidirectional	GTX 780	1152s	3Gb	50
Seq2seq	GTX 780	22032s	3Gb	64
Seq2seqAttention	NVIDIA Titan X	92594s	12Gb	17
Ensemble Seq2seq	NVIDIA Tesla T4	77511s	16Gb	80
New Seq2seq	NVIDIA Tesla T4	804s	16Gb	60
New Seq2seqAttention	NVIDIA Tesla T4	1256s	16Gb	60
New Ensemble Seq2seq	NVIDIA 2080ti	2384s	12Gb	60

7 Conclusion and Future Work

The task of anomalous fibers detecting is a new topic in dMRI processing under deep learning methods. Unlike other anomaly detection methods, this task requires the models to deal with continuous sequence data. In this report, firstly, we proposed a simple manual method base on three measurements. Each measurement detects each kind of anomalous fibers. We call it the manual method since there are a lot of control parameters to set.

For deep learning models, we built deep autoencoders with different types of recurrent units and different mechanisms, then measure the performance with the manually generated labels. Massive evidence showed the autoencoder models did a much better job, and in the worst case, the autoencoder models perform the same as the manual method. There are four types of anomalous curves, the deep learning models could detect all of them except the short curves, but this would not affect the performance since all the deep models could not detect them, and we can filter them out just by calculating the fiber length.

We planned to learn a compact encoded expression of the inputs fibers data as well, so we trained under sequence to sequence model, seq2seq with attention, ensemble seq2seq model. Unfortunately, these ideas are not working. So we tested the capacity of this model concerning its configuration and built simple seq2seq models with down-sampled curves. Since we did not have enough time on diving in the seq2seq models, this leaves us further work for exploring:

1. We need to add structures to improve the capacity of the new sequence to sequence models. They have shown good generalization capability. It is possible to modify this model to have better performance.

2. If an anomalous curve is detected in the bundle, the reconstruction from autoencoder might imply the correct trajectory.

3. The ensemble seq2seq model is simplified from the paper [16], we could change the RNN structure to the skip-RNN structure they proposed to see how the performance change.

4. In [15] they found embedding the contextual information could improve the

performance. We could also embed the information on subjects to see how the performance change.

There are also leaves enough space for fine-tuning, for example the depth of network, the dropout mechanism, the activate function, the optimizing method and the batch size.

References

[1] N. Zheng, "Fiber tracking for brain white matter," Ph.D. dissertation (cit. on p. 1).

[2] S. Mori, B. J. Crain, V. P. Chacko, and P. C. M. V. Zijl, "Three-dimensional tracking of axonal projections in the brain by magnetic resonance imaging," *Annals of Neurology*, vol. 45, no. 2, 1999 (cit. on p. 1).

[3] M. Lazar, D. M. Weinstein, J. S. Tsuruda, K. M. Hasan, K. Arfanakis, M. E. Meyerand, B. Badie, H. A. Rowley, V. Haughton, and A. Field, "White matter tractography using diffusion tensor deflection," *Human Brain Mapping*, vol. 18, no. 4, pp. 306–321, 2010 (cit. on p. 1).

[4] J. Wasserthal, P. Neher, and K. Maier-Hein, "Tractseg - fast and accurate white matter tract segmentation," *NeuroImage*, vol. 183, Aug. 2018 (cit. on pp. 2, 3, 15).

[5] J. Wu, W. Zeng, and F. Yan, "Hierarchical temporal memory method for time-series-based anomaly detection," *Neurocomputing*, S0925231217313887, 2016 (cit. on p. 1).

[6] P. C. Austin, J. V. Tu, J. E. Ho, D. Levy, and D. S. Lee, "Using methods from the data-mining and machine-learning literature for disease classification and prediction: A case study examining classification of heart failure subtypes," *Journal of Clinical Epidemiology*, vol. 66, no. 4, pp. 398–407, 2013 (cit. on p. 1).

[7] S. Ramaswamy, R. Rastogi, K. Shim, and T. Korea, "Efficient algorithms for mining outliers from large data sets," Apr. 2000 (cit. on p. 3).

[8] K. D. Sequeira, *Anomaly-based data mining of intrusions*, 2002 (cit. on p. 3).

[9] A. Mccallum, K. Nigam, and L. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," *Proceeding of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Mar. 2000 (cit. on p. 3).

[10] J. B. D. Cabrera, L. Lewis, and R. K. Mehra, "Detection and classification of intrusions and faults using sequences of system calls," *ACM SIGMOD Record*, 2001 (cit. on p. 3).

[11] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," in *Acm Sigmod International Conference on Management of Data*, 2000 (cit. on p. 3).

[12] M. Khatami, T. Schmidt-Wilcke, P. C. Sundgren, A. Abbasloo, B. Sch?Lkopf, and T. Schultz, "Bundlemap: Anatomically localized classification, regression, and hypothesis testing in diffusion mri," *Pattern Recognition*, S0031320316302849, 2016 (cit. on p. 3).

[13] L. M. Manevitz and M. Yousef, "One–class svms for document classification," *Journal of Machine Learning Research*, vol. 2, no. 1, pp. 139–154, 2002 (cit. on p. 3).

[14] Y. Xia, X. Cao, F. Wen, G. Hua, and J. Sun, "Learning discriminative reconstructions for unsupervised outlier removal," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015 (cit. on p. 3).

[15] T. Kieu, B. Yang, and C. S. Jensen, "Outlier detection for multidimensional time series using deep neural networks," in *2018 19th IEEE International Conference on Mobile Data Management (MDM)*, 2018 (cit. on pp. 3, 47).

[16] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, "Outlier detection for time series with recurrent autoencoder ensembles," in *Twenty-Eighth International Joint Conference on Artificial Intelligence IJCAI-19*, 2019 (cit. on pp. 3, 25, 47).

[17] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science (New York, N.Y.)*, vol. 313, pp. 504–7, Aug. 2006 (cit. on p. 5).

[18] J. Masci, U. Meier, D. Ciresan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International Conference on Artificial Neural Networks*, 2011 (cit. on p. 5).

[19] H.-I. Suk, S.-W. Lee, and D. Shen, "Latent feature representation with stacked auto-encoder for ad/mci diagnosis," *Brain Structure and Function*, vol. 220, Mar. 2013 (cit. on p. 5).

[20] S. Ruder, "An overview of gradient descent optimization algorithms," Sep. 2016 (cit. on p. 6).

[21] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org` (cit. on pp. 6, 7).

[22] A. Graves, "Supervised sequence labelling," 2012 (cit. on p. 7).

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997 (cit. on p. 7).

[24] dprogrammer, *Rnn, lstm gru*, `http://dprogrammer.org/rnn-lstm-gru` (cit. on p. 8).

[25] J. Chung, C. Gulcehre, K. H. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *Eprint Arxiv*, 2014 (cit. on p. 8).

[26] Z. Yang, Z. Hu, Y. Deng, C. Dyer, and A. Smola, "Neural machine translation with recurrent attention modeling," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 2016 (cit. on p. 10).

[27] S. N. Sotiropoulos, S. Jbabdi, J. Xu, J. L. Andersson, S. Moeller, E. J. Auerbach, M. F. Glasser, M. Hernandez, G. Sapiro, and M. a. Jenkinson, "Advances in diffusion mri acquisition and processing in the human connectome project.," *Neuroimage*, vol. 80, no. 20, pp. 125–143, 2013 (cit. on p. 13).

[28] M. Ankele, L. H. Lim, S. Groeschel, and T. Schultz, "Versatile, robust, and efficient tractography with constrained higher-order tensor fodfs," *International Journal of Computer Assisted Radiology Surgery*, vol. 12, no. 8, pp. 1–14, 2017 (cit. on p. 13).

[29] L. Jonasson, P. Hagmann, X. Bresson, R. Meuli, O. Cuisenaire, and J. P. Thiran, "White matter mapping in dt-mri using geometric flows," *Lecture Notes in Computer ence*, vol. 2809, no. 2809, pp. 585–596, 2003 (cit. on p. 14).

[30] M. Bogu?á, S. Pajevic, P. J. Basser, and G. H. Weiss, "A model for noise effects on fibre tract trajectories in diffusion tensor imaging: Theory and simulations," *New Journal of Physics*, vol. 7, pp. 24–24, 2005 (cit. on p. 14).

[31] E. Takahashi, G. Dai, R. Wang, K. Ohki, G. D. Rosen, A. M. Galaburda, P. E. Grant, and V. J. Wedeen, "Development of cerebral fiber pathways in cats revealed by diffusion spectrum imaging," *Neuroimage*, vol. 49, no. 2, pp. 1231–1240, 2010 (cit. on p. 14).

[32] V. Wedeen, T. Reese, J. Dou, and R. Chesler, "Mapping fiber orientation spectra in cerebral white matter with fourier-transform diffusion mri," *Proceedings of the 8th Annual Meeting of ISMRM, Denver, Colorado, USA*, Jan. 2000 (cit. on p. 14).

[33] T. Saito and J. I. Toriwaki, "New algorithms for euclidean distance transformation of an n-dimensional digitalized picture with applications," *Pattern Recognit*, vol. 27, no. 11, pp. 1551–1565, 1994 (cit. on p. 15).

[34] K. M. Jordan, B. Amirbekian, A. Keshavan, and R. G. Henry, "Cluster confidence index: A streamline-wise pathway reproducibility metric for diffusion-weighted mri tractography," *Journal of Neuroimaging*, 2017 (cit. on pp. 16, 17).

[35] G. Eleftherios, B. Matthew, C. M. Morgado, G. B. Williams, and N. S. Ian, "Quickbundles, a method for tractography simplification," *Frontiers in Neuroence*, vol. 6, no. 6, p. 175, 2012 (cit. on p. 17).