東北大學 NORTHEASTERN UNIVERSITY

毕业设计（论文）

GRADUATION DESIGN (THESIS)

论文题目	**Design and Implementation of Aided Diagnosis System of Alzheimer’s Disease Based on Evolving Graph Neural Network**
学院名称	计算机科学与工程学院
专业名称	计算机科学与技术
学生姓名	郭恩铭
指导教师	杨晓春 教授

二〇二三 年 六 月

学号____20194723____　　　　　　　　　　　　　　密级______________

东北大学本科毕业论文

Design and Implementation of Aided Diagnosis System of Alzheimer’s Disease Based on Evolving Graph Neural Network

学 院 名 称：计算机科学与工程学院

专 业 名 称：计算机科学与技术

学 生 姓 名：郭恩铭

指 导 教 师：杨晓春　教授

二〇二三年六月

Design and Implementation of Aided Diagnosis System of Alzheimer's Disease Based on Evolving Graph Neural Network

by Guo Enming

Supervisor: Professor Yang Xiaochun

Northeastern University

June 2023

基于时序图神经网络的阿尔茨海默症辅助诊断系统设计与实现

作者姓名：　　郭恩铭

指导教师：　　杨晓春　　教授

单位名称：　　计算机科学与工程学院

专业名称：　　计算机科学与技术

东 北 大 学

二〇二三年六月

Solemn Statement

The graduation thesis submitted by me is the result of independent research work under the guidance of the tutor. All data and pictures are true and reliable. As far as I know, the research results of this thesis do not include the content of copyright enjoyed by others, except as indicated in the text. Other individuals and groups that have contributed to the research work involved in this paper have been clearly identified in the text. The intellectual property rights of this thesis are attributed to the training unit.

Signature: 郭恩铭 Date: 2023.5.22

ABSTRACT

Alzheimer's Disease (AD) is a severe neurodegenerative disorder and a global memory crisis. The disease has a high incidence rate among the elderly population and is irreversible and incurable, posing a significant threat to memory function. With a large number of patients affected, there is an urgent need for an early-stage Alzheimer's Disease aided diagnostic platform that is driven by computer technology and large-scale data, integrating the characteristics of brain network structures.

In this study, the fMRI images from the publicly available ADNI database were used as the original training set. The dynamic brain network was constructed using the DPARSF toolbox based on Matlab. In the classifier construction phase, two evolving graph neural network methods were employed to effectively integrate the temporal information of the brain network, achieving the construction of evolving graph neural networks. Compared to static graph neural networks, this approach achieves higher accuracy in aided diagnosis.

Furthermore, this study designed and implemented a system called Aided Alzheimer's Diagnosis (AAD), which provides a model training function based on evolving graph neural networks and supports multi-user login with different identities. The system includes a parameter tuning panel that allows administrator users to adjust different parameters for optimizing model training. The system also supports visualization and export functions for training and testing records, facilitating data integration and analysis for users. The AAD system was developed using Python and utilized PyMongo for connecting the backend functionalities with the database. The frontend interface adopted ttkbootstrap, and MongoDB was chosen as the database. The system instantiates the following main functional components: (1) a graphical user interface for accessing platform services, (2) a visualization panel for adjusting parameters and visualizing training records, and (3) visualization of aided diagnostic results and testing records.

The AAD system constructed in this study provides researchers with an efficient aided diagnostic system for Alzheimer's Disease. Finally, through validation with specific cases, the accuracy and effectiveness of the AAD system have been demonstrated.

Key words: Alzheimer's Disease; Construction of Dynamic Brain Network; Evolving Graph Neural Network; ttkbootstrap; MongoDB

摘 要

阿尔茨海默症（Alzheimer's Disease, AD）是一种严重的神经退行性疾病，也是全球性的记忆危机。该疾病在老年人群中的发病率高且不可逆转，无法治愈，对记忆功能造成重大威胁。患者数量庞大，迫切需要一种基于计算机和大规模数据驱动、结合脑网络结构特征的早期阿尔茨海默症辅助诊断平台。

本研究利用 ADNI 公开数据库中的 fMRI 图像作为原始训练集，并运用了基于 Matlab 的 DPARSF 工具包构建了动态脑网络。在分类器构建阶段，采用了两种时序图神经网络方法，有效地融合了脑网络的时序信息，实现了时序图神经网络的构建。与静态图神经网络相比，该方法可获得更高的辅助诊断准确率。

此外，本研究还设计并实现了名为 AAD（Aided Alzheimer's Diagnosis）的阿尔茨海默症辅助诊断系统。AAD 系统提供了基于时序图神经网络的模型训练功能，并支持多用户身份的登录。其中包括调参面板，可供管理员调节不同参数以优化模型训练。系统支持训练和测试的记录可视化展示与导出功能，便于用户整合和分析数据信息。AAD 系统采用 Python 语言开发，利用 PyMongo 实现后端功能与数据库的连接，前端界面采用 ttkbootstrap，数据库采用 MongoDB。系统实例化了以下主要功能组件：（1）图形用户界面，用于访问平台服务；（2）可视化参数调节面板和训练记录可视化；（3）辅助诊断结果可视化和测试记录可视化。

本研究构建的 AAD 系统为临床诊断和健康监测人员提供了一个高效的阿尔茨海默症辅助诊断系统，最后，通过实际案例的测试，验证了 AAD 系统的准确性和高效性。

关键词：阿尔茨海默症；动态脑网络构建；时序图神经网络；ttkbootstrap；MongoDB

Content

Chapter 1 Introduction

1.1. Research Background

In 1906, Alois Alzheimer put forward the concept of "Alzheimer's disease", so later generations named it after him, that is, Alzheimer's disease. Alzheimer's disease is a serious neurodegenerative disease, and it is also a memory crisis that plagues the whole world. According to WHO estimates, more than 55 million people (8.1% of women and 5.4% of men in the over-65s) currently have dementia. It is estimated that this number will rise to 78 million by 2030 and 139 million by 2050. In China, Alzheimer's disease also has a great impact on the health of the elderly. According to China Alzheimer's Disease Report 2021, there are 15.07 million dementia patients aged 60 and above, of which 9.83 million are AD patients.

In recent years, functional imaging has developed rapidly, which also provides a prerequisite for studying the brain regions of AD patients. Functional Magnetic Resonance Imaging (fMRI) technology has become a method to observe the structure and function of the brain, and is becoming more and more mature and perfect. In fMRI images, there is connection and functional correlation between different brain regions of human brain. Constructing and analyzing brain network can better describe the active state of brain and the interaction between various brain regions, and calculate the signal correlation between different brain regions to represent the functional connection between them. With the development of research in recent years, the domain knowledge of graph theory has been introduced into the study of brain functional network, which makes the complex brain be studied in depth as an abstract model. However, the common brain network construction methods are mainly based on static brain network, which often ignores the information in time dimension and cannot make full use of the high time resolution of fMRI technology. Therefore, the existing research is mainly based on the construction of evolving brain network, and then shows the changing characteristics of dynamic connection of brain and reflects the information of brain activity in a period of time.

Graph neural network is a kind of deep learning model for modeling and analyzing graph data. Unlike the traditional deep learning model, which aims at regular data such as vectors or matrices, the input of graph neural network is graph data, in which nodes represent entities and edges represent the relationships between entities. Its advantage is that it can process graphic data with arbitrary topological structure and extract useful information from it. Traditional deep learning models can only process input of fixed size, while graph neural network can adaptively process graph data of arbitrary size and shape. In addition, graphical neural networks have some interpretability, which can explain the importance of nodes and edges. In the field of graph classification, the outstanding point is that it can capture the global structure and local features of graph data, so it has higher classification accuracy and generalization ability.

1.2. Research Significance

Graph neural network is a deep learning model for modeling and analyzing graph data, which has the advantage of adapting to graph data processing with arbitrary topological structure and extracting useful information from it. Evolving graphs represent the dynamic evolution process of brain network and can reflect the time sequence characteristics of brain activity. In the aided diagnosis of Alzheimer's disease, the method based on evolving graph neural network can make full use of the advantages of high spatial and temporal resolution of functional magnetic resonance imaging, model and analyze the dynamic evolution process of brain network, and effectively improve the diagnostic accuracy and early warning ability of diseases, which is of great significance to the treatment and management of Alzheimer's disease. Therefore, the design and implementation of the aided diagnosis system of Alzheimer's disease based on evolving graph neural network is proposed.

1.3. Relevant Research Status at Home and Abroad

Aided diagnosis of Alzheimer's disease has been widely studied at home and abroad. In the diagnosis of Alzheimer's disease based on machine learning method, Li Cai et al [1] proposed the classification prediction of Alzheimer's disease based on machine learning, in which machine learning method was used to classify and predict Alzheimer's disease with brain structure magnetic resonance, age, gender, years of education and MMSE scale score as characteristics. Jo et al [2] uses deep learning method to construct convolution neural network to train and classify important features extracted from medical images for diagnosis. However, feature extraction in related fields lacks research on brain network structure, and cannot explore the correlation between changes in brain connections and Alzheimer's disease.

Graph kernel method has gradually become a research hotspot in the diagnosis of Alzheimer's disease. It is a measure of graph similarity, so it can express the topological attributes of brain network and describe the structural information of brain network more comprehensively. Jiebiao et al [3] proposed a graph kernel method to calculate the similarity of brain networks to realize disease diagnosis. With the continuous progress and development of research, new kernels are constantly constructed to gradually optimize the algorithm [4-5].

The domain knowledge of graph theory is introduced into the study of brain functional network, which makes the complex brain be studied in depth as an abstract model. Cao et al [6] once proposed to use the discriminative subgraph patterns of brain network as a marker to diagnose bipolar disorder. Discriminative subgraphs can not only reflect the topological structure of brain network, but also more easily capture the connection patterns in the network of lesion areas and reflect the differences between normal people and AD patients, which can be used as biological markers for aided diagnosis of Alzheimer's disease. The algorithm concerning mining discriminative subgraphs is also in the process of continuous research and development [7-9]. Although the method can better reflect the topological structure of brain network, the existing research results are mainly based on extracting the discriminative subgraphs of static brain network [10-12], and there are also some problems such as high complexity of graph mining algorithm and limited training ability for large dataset.

In recent years, more and more studies have integrated the longitudinal data of brain network into the research, and obtained the dynamic feature information of brain network by sampling different cases at different time points. Nguyen et al [13] proposed to use minimal RNN model to fill in time series information and explore the time series changes of important features of brain network. Because resting fMRI can reflect the functional cognitive state of subjects in a period of time, the construction of static brain network often ignores the information in time dimension, and fails to make full use of the high temporal resolution of fMRI technology. On the contrary, the dynamic brain network constructed in different time periods can show the dynamic connection changes of the brain, and then reflect the information of brain functional activity in each time period.

Graph neural network can retain the topological connection characteristics of brain network, and effectively fuse multi-modal features and model the correlation between samples. Parisot et al[14] initially applied GCN (Graph Convolution Neural Network) to integrate image and non-image features to complete the prediction of lesions. Mansu Kim [15] and others put forward an interpretable GNN model, which makes full use of the longitudinal data of dynamic brain network to judge the influence of important features of brain network on auxiliary diagnosis. The existing research has not fully combined evolving brain network with multiple graph neural networks, so an aided diagnosis method of Alzheimer's disease based on evolving graph neural network is proposed.

1.4. Research Content

The main research contents focus on the preprocessing of fMRI, the construction of evolving brain networks, the aided diagnosis of Alzheimer's disease based on evolving neural network, and the comparative experiments combined with the existing related research results.

Finally, the aided diagnosis system of Alzheimer's disease is realized by building a full-stack platform.

Firstly, through the fMRI downloaded from ADNI (Alzheimer's Disease Neuroimaging Initiative) public database, the preprocessing process of the original image data is completed, which mainly includes time layer correction, head motion correction, spatial standardization, smoothing, de-linearity drift and filtering, etc.

After the completion of image preprocessing, the construction of an evolving brain network was initiated. It is necessary to extract BOLD signal from fMRI image after data preprocessing. Since the original data has acquired multiple fMRI in a period of time, it is necessary to introduce sliding window technology to segment the extracted BOLD signal after completing BOLD signal extraction. After completing the above steps, a correlation matrix is constructed to represent the correlation between different brain regions. After correlation matrix is thresholding, finally a binary matrix is constructed to show the connection relationship between brain regions.

The node features of the constructed brain network are extracted, and the node features of adjacent time slices of the constructed evolving brain networks are fused to complete the feature fusion. After that, different graph neural network models are used for model training. Optimize parameters to improve the accuracy of aided diagnosis. In the contrast experiment, the aided diagnosis effect of evolving brain network is compared with that of static brain network.

Finally, the model is used to classify Alzheimer's cases and normal cases, and the front-end interface is made based on python GUI library ttkbootstrap, and MongoDB is used to store data. Finally, the whole course design completes a complete, user-friendly and robust application.

1.5. Article Structure

The structure of the entire article is organized as follows:

Chapter 1 serves as the introduction, providing an overview of the research background, which includes the global prevalence of Alzheimer's disease and the favorable conditions offered by advancements in computer science for intelligent medicine. It highlights the research significance and presents the current research status both domestically and internationally. Additionally, it briefly introduces the system and organizational structure of this paper.

Chapter 2 primarily focuses on providing a technology overview of the AAD system. It covers the Python library functions employed for model training, as well as the programming languages used for frontend and backend development. These knowledge and technologies form the fundamental basis for implementing the AAD system.

Chapter 3 mainly delves into the construction process of the dynamic brain network. This section details how fMRI data is processed into computationally manageable brain network data.

Chapter 4 extensively explains the construction process of the temporal graph neural network. The chapter introduces two types of temporal graph neural networks and also discusses the analysis of experimental performance. This section provides the technical support for the entire diagnostic assistance system.

Chapter 5 focuses on the requirements analysis of the AAD system. It describes the requirements of different types of users, including ordinary users, advanced users, and administrator users. The chapter encompasses both the functional and non-functional requirements of the AAD system, providing a theoretical basis for the system design discussed in Chapter 6.

Chapter 6 presents the system design of the AAD system, with particular emphasis on interface design and database design. This chapter lays a solid foundation for the system implementation discussed in Chapter 7.

Chapter 7 covers the implementation of the AAD system. Building upon the system design outlined in the previous chapter, this section describes the implementation and testing of the AAD system. It provides a detailed description of the application technology used for each module of the AAD system, along with environment configuration. The chapter also elaborates on the implementation details of each module, presents interface screenshots, and discusses functional design in conjunction with the interface.

Chapter 8 entails the system testing of the AAD system. The purpose of this chapter is to assess the robustness and accuracy of the AAD system. It employs functional testing and integrity testing to evaluate the system.

Chapter 9 concludes the article with a summary and outlook. This chapter summarizes the AAD system and looks ahead to potential future upgrades for the system, laying the groundwork for further improvement in subsequent work.

Chapter 2 Technology Overview

In this chapter, technologies used in the scientific research and development process are briefly introduced by discussing relative advantages and current application prospect.

2.1. DPARSF

DPARSF (Data Processing Assistant for resting-state fMRI) is a Matlab-based brain imaging data preprocessing tool, which is mainly used to preprocess and analyze resting-state functional magnetic resonance imaging (resting-state fMRI) data. DPARSF is based on SPM12 (Statistical Parametric Mapping Software Package). By using the algorithm and toolbox of SPM12, DPARSF realizes the preprocessing and analysis of resting fMRI data. The software can process resting fMRI data of various data formats including DICOM, NIfTI and AFNI, and perform preprocessing steps such as slice orientation correction, realignment, time series processing, denoising, linear trend removal, normalization and spatial smoothing. In addition, DPARSF can also calculate the functional connections between the whole brain and regions, and provides various visual tools to display and analyze data.

Currently, DPARSF is extensively employed in neuroscience and psychology research to investigate the pathological mechanisms underlying various neurological disorders (such as stroke, Alzheimer's disease, schizophrenia, etc.), the processes of brain development and aging, as well as the brain network connectivity among different individuals and groups.

The fundamental operational principle of DPARSF involves multiple preprocessing steps of the original fMRI data. These include removing the effects of head motion, denoising, slice timing, normalization, smoothing, among others. Subsequently, further statistical analysis and functional connectivity analysis are performed. DPARSF offers numerous functions and options, allowing users to tailor their data processing and analysis, such as customizing processing flow and selecting specific regions of interest for functional connectivity analysis, among others.

2.2. ttkbootstrap

The ttbootstrap technology is applied to the front-end design of AAD system. The ttkbootstrap is a Python-based GUI framework built on top of tkinter and ttk libraries. It aims to simplify the development process of user interfaces by providing modern and highly customizable design options.

ttkbootstrap offers a rich set of predefined styles and themes, allowing developers to easily create user interfaces with a professional look. These styles and themes cover various controls such as buttons, labels, text boxes, dropdown lists, enabling developers to quickly choose an appearance that suits their application's style.

In addition to providing predefined styles, ttkbootstrap also supports full customization capabilities. Developers can customize the appearance of controls by modifying colors, fonts, sizes, and other properties according to their needs. This flexibility allows developers to create user interfaces that align with their brand image or application theme.

ttkbootstrap also provides powerful layout managers, allowing developers to easily achieve complex user interface layouts. It supports various common layout methods such as grid layout, pack layout, and absolute layout, enabling developers to flexibly arrange and organize interface elements.

Another important feature is the support for event handling and callback functions. Developers can respond to and process user interactions on the interface by binding events and writing corresponding handling code. This enables developers to implement logic control for interactive actions such as button clicks, mouse movements, and keyboard input.

Overall, the ttkbootstrap framework offers a simple, flexible, and customizable way to build user interfaces. Its predefined styles, themes, and layout managers make interface design easy and professional. Whether it's rapid prototyping or building complex applications, ttkbootstrap is a powerful and worthwhile tool to explore.

2.3. MongoDB

MongoDB is a popular document-based, distributed NoSQL database, which is used as the database for the AAD system. MongoDB documents are stored in BSON (Binary JSON) format, which can support complex data types and nested documents, and has high flexibility and extensibility. MongoDB supports automatic fragmentation, which can realize horizontal scaling and improve system throughput and availability.

MongoDB is also an opensource project that runs on a variety of operating systems, including Windows, Linux, and MacOS. It provides drivers for many programming languages, such as Java, Python, Ruby, Node.js, and so on. In MongoDB, data is stored in one or more collections, each containing one or more documents. Collections can be regarded as analogies of tables in relational databases. The main advantages of MongoDB are as follows:

(1) Highly flexible data model

MongoDB employs a document-based data model that is highly adaptable, capable of accommodating complex data types and nested documents, and can effortlessly scale and adjust data. Furthermore, it features dynamic querying, which provides the flexibility to query data in a variety of ways.

(2) Scalability and high availability

MongoDB provides various functions such as automatic sharding and replica set, enabling easy horizontal scaling and high availability. This feature makes it capable of handling high concurrency and large amounts of data, ensuring robustness and reliability in critical applications.

(3) High performance

MongoDB is known for its high performance and ability to handle highly concurrent access requests. One of the key factors contributing to its performance is the use of memory-mapped file technology, which allows data to be stored in memory and significantly improves reading performance.

(4) Easy Learning Curve and User-Friendly

MongoDB has a simple and easy-to-learn syntax, making it effortless for developers to use. They can utilize graphical interface tools like MongoDB Shell or MongoDB Compass to manage and query data with ease.

In summary, MongoDB is a highly flexible, scalable, and performant database, designed to meet the demands of big data and high-concurrency application scenarios, with the added advantage of high availability.

2.4. Pymongo

Pymongo is a Python library and the official MongoDB driver for Python, which is a excellent Python library that allows developers to connect to and interact with MongoDB databases. It provides a convenient way to work with MongoDB using Python code, enabling seamless communication and data manipulation.

It provides a high-level interface for interacting with MongoDB databases, allowing developers to perform various operations such as querying, inserting, updating, and deleting data. Pymongo is used for integrating MongoDB with Python applications, allowing developers to leverage the power and flexibility of MongoDB in their projects. It provides a rich set of features and functionalities to work with MongoDB's document-oriented database model.

Focus on the operational functions, Pymongo offers a wide range of features, including support for CRUD (Create, Read, Update, Delete) operations, aggregation pipelines, indexing, transactions, and grid file system for storing large files. It also supports advanced features like sharding and replica sets for scaling and high availability.

In comparison with other database drivers, some advantages of using Pymongo include its simplicity, as it provides a clean and intuitive API for interacting with MongoDB. It also offers excellent performance, allowing for efficient data retrieval and manipulation. Pymongo's flexibility is another advantage, as it supports various data types and provides powerful query capabilities. Additionally, being the official MongoDB driver, Pymongo is well-maintained, reliable, and constantly updated to support the latest MongoDB features.

In summary, Pymongo is a powerful and versatile Python library for working with MongoDB. It simplifies the integration of MongoDB into Python applications, providing a user-friendly interface and a range of features for efficient data management and manipulation.

2.5. NetworkX

NetworkX is an open-source Python library used for creating, manipulating, and analyzing complex networks. And in AAD system, Networkx library provides a very convenient programming environment for brain feature extraction in brain network analysis. It offers a simple and flexible way to represent and handle network structures and provides a variety of algorithms and tools for exploring the structure, characteristics, and dynamics of networks. NetworkX supports the creation and handling of different types of networks, including directed graphs, undirected graphs, weighted graphs, and more. It provides a comprehensive set of functions and methods for adding nodes and edges, calculating network metrics, traversing network structures, finding shortest paths, performing community detection, and more. Additionally, NetworkX offers visualization tools to aid in the visual representation of networks.

NetworkX is widely used in various fields of research and application, including social network analysis, biological network analysis, transportation network analysis, network science research, and more. It is a powerful and flexible tool that can be applied to networks of various sizes and types, ranging from small-scale networks to large-scale complex networks.

Networkx has lots of advantages providing users better programming environment. One of the advantages of NetworkX is its user-friendly interface and clear documentation, enabling users to quickly get started and understand its usage. It also offers excellent scalability, allowing users to develop custom algorithms and extensions according to their specific needs. Furthermore, NetworkX benefits from a strong community support, providing users with extensive documentation, examples, and discussions for assistance and support.

Developers and scientists currently apply NetworkX in a wide range of tasks, including social network analysis, recommendation systems, propagation models, path planning, community detection, graph theory research, and more. It serves as a powerful tool for researchers and developers to analyze and understand the structure and dynamics of complex networks.

In summary, NetworkX is a powerful and flexible Python library for creating, manipulating, and analyzing complex networks. It provides a rich set of functionalities and algorithms suitable for studying and understanding network structures and dynamics. Whether in academic research, data analysis, or practical applications, NetworkX is a valuable tool for exploring and comprehending the characteristics of complex networks.

2.6. Graph Neural Network Algorithm Library

2.6.1. Pytorch Geometric

PyTorch Geometric is a Python library used for handling graph data. Built on top of the PyTorch deep learning framework, it provides a rich set of graph neural network models and data processing tools. The main purpose of PyTorch Geometric is to simplify the preprocessing and modeling of graph data for graph machine learning and graph deep learning tasks. In the contrast experiment of graph neural network, we introduce the static brain network method to carry out comparative experimental analysis. The experiment is based on Pytorch Geometric library to complete the network building and programming. Here is a detailed explanation of PyTorch Geometric:

(1) Graph Data Processing: PyTorch Geometric offers efficient tools and functions for loading, preprocessing, and transforming graph data. It supports various graph data formats, including edge lists, adjacency matrices, and feature matrices, allowing users to easily handle and represent graph structures and features.

(2) Graph Neural Network Models: PyTorch Geometric implements many classical and advanced graph neural network models, such as Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), GraphSAGE, Graph Isomorphism Network (GIN), and more. These models, based on deep learning techniques, can effectively learn and represent complex relationships and patterns in graph structures.

(3) Graph Representation Learning: PyTorch Geometric provides a range of graph representation learning methods for mapping graph data to low-dimensional vector spaces. These methods include DeepWalk and Node2Vec based on random walks, as well as GraphSAGE and Graph Convolutional Networks (GCN) based on graph neural networks. By learning compact graph representations, they can be easily applied to tasks such as graph classification, node classification, graph generation, and more.

(4) Graph Data Visualization: PyTorch Geometric also offers tools for visualizing graph data, allowing for intuitive displays of graph structures. Users can visualize graph data as graphs

or charts by plotting nodes and edges, facilitating better understanding and analysis of patterns and features in the graph.

In summary, PyTorch Geometric is a dedicated Python library for handling graph data. It provides a rich set of graph neural network models and data processing tools, enabling users to easily perform graph machine learning and graph deep learning tasks. With PyTorch Geometric, users can quickly load and preprocess graph data, build and train graph neural network models, and visualize graph structures and features. It serves as a powerful and flexible tool for researchers and developers to efficiently process data and model in the graph domain.

2.6.2. Pytorch Geometric Temporal

PyTorch Geometric Temporal is an extension library based on PyTorch Geometric, specifically designed for handling dynamic graph data. It provides a collection of graph neural network models and data processing tools for temporal modeling. The goal of PyTorch Geometric Temporal is to simplify the processing and modeling of dynamic graph data, enabling users to apply it to tasks such as time series prediction, event forecasting, and temporal graph analysis. In the experiment of Graph Neural Network, the experiment is based on Sequential Graph Neural Network, and the Pytorch Geometric Temporal library function is used to build the model. Here is a detailed introduction to PyTorch Geometric Temporal:

(1) Evolving Graph Data Modeling: PyTorch Geometric Temporal offers tools and functions for processing dynamic graph data. It supports handling graph data with temporal information, such as time series graphs, temporal graphs, and event graphs. Users can easily load, preprocess, and transform graph data with temporal dimensions, facilitating the modeling and analysis of temporal features.

(2) Evolving Graph Neural Network Models: PyTorch Geometric Temporal implements various graph neural network models for temporal modeling. These models combine graph neural networks with time series modeling techniques, effectively learning and representing temporal relationships and patterns in dynamic graphs. Some commonly used temporal graph neural network models include Temporal Graph Convolutional Networks (TGCN), Graph Convolutional LSTM (GCLSTM), and more.

(3) Dynamic Graph Data Processing: PyTorch Geometric Temporal provides data processing tools for handling dynamic graph data. It supports operations such as graph data splitting, sliding windows, time series feature extraction, and temporal alignment, allowing users to preprocess and model dynamic graph data conveniently. These tools can handle dynamic graph data with different time steps, resolutions, and intervals.

(4) Visualization of Temporal Graph Data: PyTorch Geometric Temporal also offers tools for visualizing temporal graph data. Users can visualize the temporal patterns and dynamics of graph data by plotting the temporal changes of the graph, temporal features of nodes and edges, facilitating intuitive understanding and analysis of temporal patterns and dynamics in dynamic graph data.

In summary, PyTorch Geometric Temporal is a PyTorch extension library for handling dynamic graph data. It provides graph neural network models and data processing tools to simplify the processing and modeling of dynamic graph data. With PyTorch Geometric Temporal, users can easily load and preprocess dynamic graph data, build and train temporal graph neural network models, and visualize the temporal features of graph data. It serves as a powerful and flexible tool for researchers and developers to process and analyze dynamic graph data, and apply it to tasks such as time series prediction, event forecasting, and temporal graph analysis.

2.7. Summary

This chapter provides a concise overview of the key technologies employed in the AAD system. Leveraging external components and frameworks facilitates agile and effective development. Incorporating third-party benchmark and algorithm libraries enhances the scalability and robustness of the AAD system. These theoretical frameworks establish the groundwork for the analysis and design of the AAD system, paving the way for its successful implementation.

Chapter 3 Dynamic Brain Function Network Modeling

Current research on brain functional networks often involves constructing a correlation connection matrix based on resting-state fMRI data to perform static network analysis. However, the brain is a dynamic structure, and the connections between neurons within the brain change over time. By utilizing a method for constructing a dynamic network, we can better analyze the instantaneous characteristics of each time period within fMRI data. Through the construction and analysis of a dynamic brain function network, we are able to more effectively describe the brain's activity state and the relationships between neurons or regions of interest. This approach allows for a greater understanding of brain dynamic activity and enables more accurate judgments regarding the differences between brain functional networks in different states.

3.1. Image Preprocessing

During the process of generating fMRI data, various factors such as individual differences among subjects and errors of MRI instruments may exert an impact on the resulting fMRI images. In order to mitigate the influence of these extraneous factors, it is necessary to preprocess the fMRI images, rendering the data commensurable and amenable to the extraction of relevant information. Failure to engage in such preprocessing could lead to problematic outcomes, such as instances in which two images that have not been standardized may yield different representations of the same brain region following template matching. By conducting appropriate preprocessing of the data, it is possible to enhance control over the variables in comparative experiments and yield optimal research outcomes.

The DPARSF software is utilized in preprocessing the brain functional image data, which is a tool designed for this specific purpose. The toolkit provides various functions, including slice timing, realignment, standardization, and smoothing, which are widely used. The detailed workflow is illustrated in Figure 3.1.

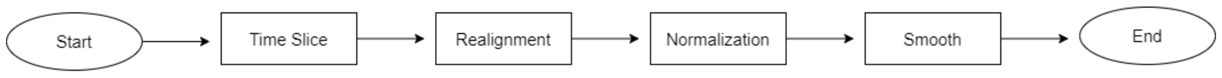


Fig. 3.1 fMRI Images Preprocessing Flowchart

3.1.1. Slice Timing

Since the fMRI images are generated by scanning layer by layer of the instrument, 140 images will be obtained in one TR time, and displaying these 140 images at the same time will form a volume. This means that the images that make up a volume are not acquired at the same time point. The earliest and latest images in these 140 images differ by exactly TR time. This time difference will have a significant impact on subsequent analysis. Therefore, the time layer correction step is used to eliminate the time difference caused by the scanning method of the fMRI instrument, so that the time of each layer within one TR time is the same.

3.1.2. Realignment

During the scanning process, even if the subject's head is fixed, there may be slight head rotation due to two reasons: the scanning process takes too long, and the subject's physiological factors can also cause slight head rotation, which in turn changes the position of voxels and affects the collected signals. Realignment involves aligning each scanned images with the first image. To ensure the accuracy of the research results, researchers select data with smaller head motion for analysis, such as rotation not exceeding 1.0°, and translation not exceeding 1.0 mm. If the allowable range is exceeded, the image needs to be excluded. Figures 3.2 and 3.3 show the SPM software correction reports on realignment at translation and rotation angles. Figure 3.2 is the distance curve of image translation in a time series, and Figure 3.3 is the angle curve of image rotation in a time series. It can be observed that the rotation angle of the subjects does not exceed 0.5, and the translation does not exceed 0.5 mm.

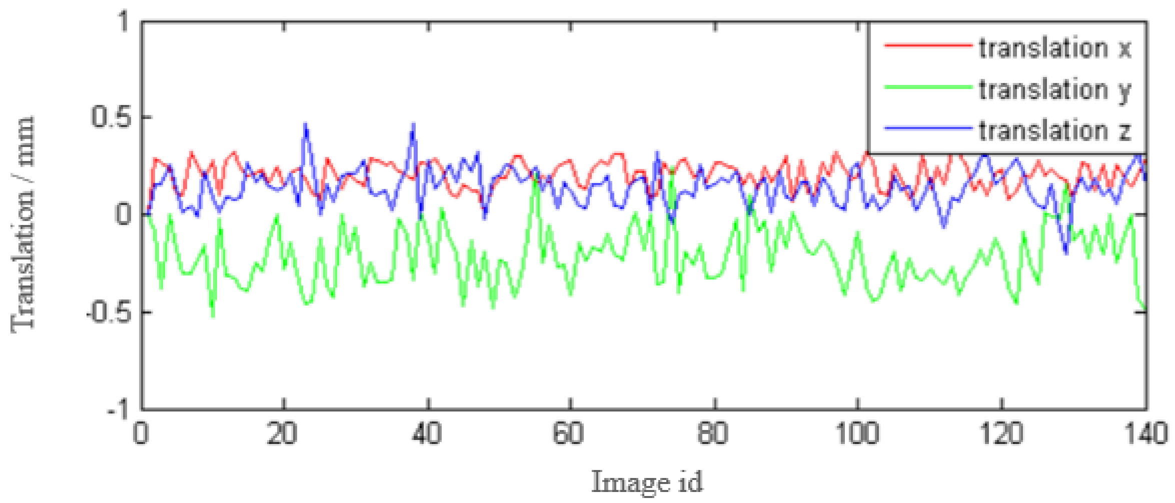


Fig. 3.2 Translation Distance Curve for Head Motion Correction

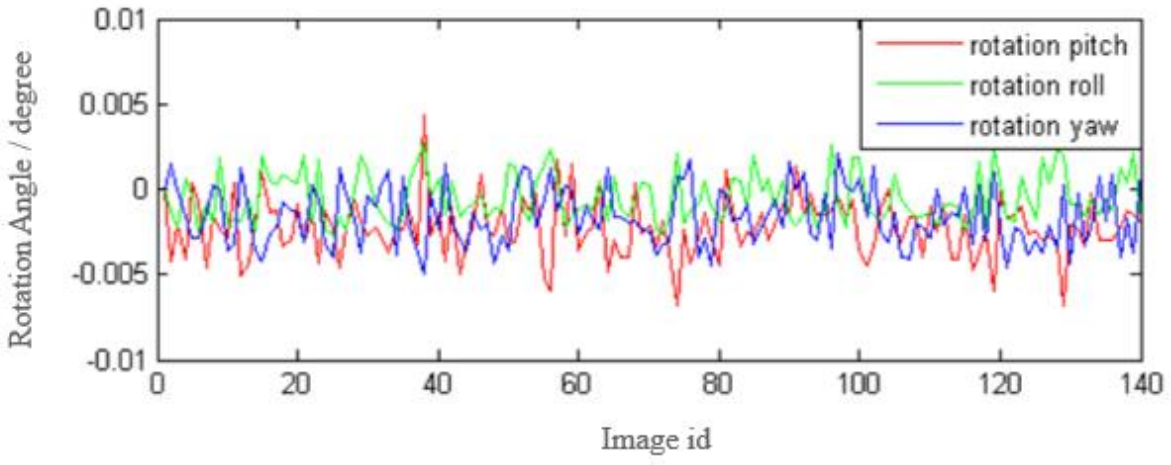


Fig. 3.3 Rotation Angle Curve for Head Motion Correction

3.1.3. Normalization

For the acquired fMRI data, there are rotation phenomena during data storage in the nuclear magnetic resonance instrument, and different patients have different brain sizes and shapes. Therefore, to reduce experimental errors, the data with rotation phenomena are standardized to a unified standard angle, and brain regions with size discrepancies are matched to a unified standard through scaling and stretching. In this way, all fMRI data are obtained, and the same brain region is located in the same area.

3.1.4. Smooth

During the acquisition of fMRI data, the image quality may be compromised due to subject-related factors or limitations of the instrument itself, resulting in strong noise. To enhance the accuracy of the data, a smoothing function is applied to remove noise and improve the signal-to-noise ratio.

3.2. Construction Method of Dynamic Brain Function Network

To construct a dynamic brain function network, it is crucial to have a clear understanding of the conventional process of building static brain functional networks. The construction methodology of the dynamic brain network is predicated on the conventional static brain network.

3.2.1. Division of Region of Interest (ROI)

Traditional brain functional networks are primarily established based on standard templates at the brain region level. For instance, using the Anatomical Automatic Labeling (AAL) template [16] [17], the preprocessed fMRI image is first matched with the AAL template. Then, the 26 regions associated with the cerebellum are removed from the matching results, and the remaining 90 brain regions are selected as the 90 nodes of the brain functional network. By regarding brain regions as nodes, the principle of complex network can be used to analyze the brain functional network, and the node representation of the brain functional network [18] is shown in Figure 3.4. Accordingly, the regions of interest in the brain are extracted.

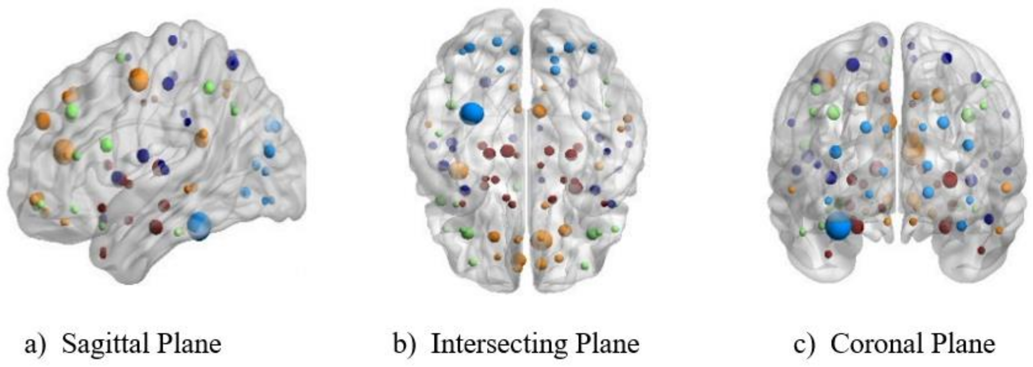


Fig. 3.4 The Representation of the Brain Network Nodes

3.2.2. Extracting ROI Time Series

In general, after the preprocessing stage, a single fMRI scan contains a sequence of fMRI images, which can reveal the changes in blood oxygen level signals over time. Figure 3.5 illustrates this process. Following preprocessing, each voxel in the sequence of fMRI images has a unique spatial position and a corresponding signal value. To obtain the time series of a voxel, we extract its value (indicated by the red dot in Figure 3.5) from each image in the sequence. The time series of a brain region is then obtained by averaging the values of all voxels within that region, which represents the instantaneous value of the brain region at a given time, also known as the BOLD signal $X_i(i = 1,2, \ldots, m)$, m represents the number of different voxels.

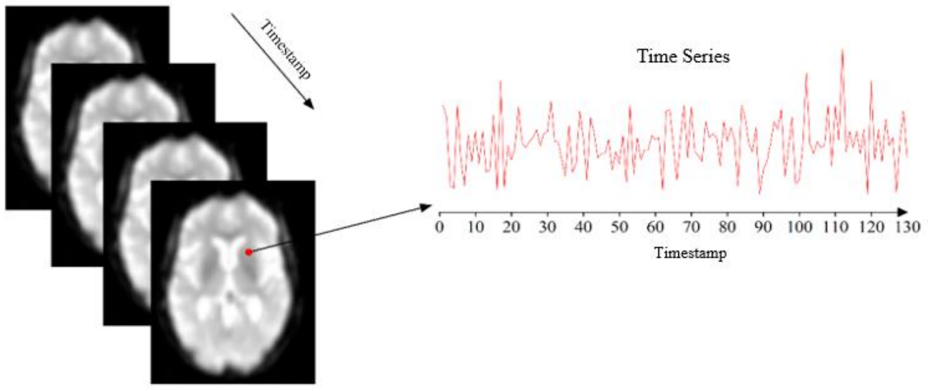


Fig. 3.5 Extract Time Series

3.2.3. Time Series Segmentation

To obtain a dynamic brain function network, the time series of each brain region can be divided into overlapping windows along the time axis using sliding time window technology. Within each window, the time series correlation between the brain regions is calculated, resulting in an adjacency matrix. This process is repeated for each window, providing an evolving analysis of the dynamic brain function network. For the BOLD signal of every brain region $X_i(i = 1,2, \dots, m)$, the time series $X_i(i = 1,2, \dots, m)$ are segmented into shorter time series $Y_{i,w}(i = 1,2, \dots, m; w = 1,2, \dots, n)$, n represents the number of sliding time windows. Through selecting appropriate parameters window size t and sliding step s, for example, in this essay, $t = 30, s = 25$, the whole time series is segmented into n time periods.

$$n = \frac{K - t}{s} + 1 \tag{3.1}$$

In which K represents the length of the time series, with a value of 130, and the total number of sliding time windows, n, is determined to be 5.

3.2.4. Correlating Brain Regions

Using the time series data from various brain regions, the correlation between these regions is calculated. The Pearson correlation method is employed in this study to determine the degree of correlation between two brain regions, which results in a correlation matrix.

The Pearson correlation coefficient [19] is a linear statistical measure used to express the degree of linear correlation between two variables. Specifically, the Pearson correlation coefficient between two variables is defined as the quotient of the covariance and standard deviation between them.

$$\rho_{x,y} = \frac{cov(X,Y)}{\sigma_x \sigma_y} = \frac{E\left[(X - \mu_x)\left(Y - \mu_y\right)\right]}{\sigma_x \sigma_y} \tag{3.2}$$

In which $cov(X, Y)$ represents the covariance between variable X and Y. σ_x and σ_y stand by the standard deviation of X and Y separately.

Utilizing the corresponding $Y_{i,w}(i = 1,2, \dots, m)$ of the w-th window to calculate its connection matrix $R_w(Size: m \times m)$. The correlation matrix of different brain regions can be obtained by this step.

$$R_w(i,j) = \begin{cases} corr\left(Y_{i,w}, Y_{j,w}\right) & i \neq j \\ 0 & i = j \end{cases} \tag{3.3}$$

In which $R_w(i, j)$ represents the element value of row i and column j in R_w and $corr()$ denotes the utilization of algorithm to calculate the Pearson correlation.

3.2.5. Generating Binary Matrix

In this process, a suitable threshold is chosen to assign a value of 1 to nodes that are greater than or equal to the threshold value, while nodes that are less than the determined threshold are assigned a value of 0.

Above all, every subject gets n functional connection matrixes $R_w(w = 1,2,\dots,n)$, and every matrix represents a functional brain network. The n matrixes construct the dynamic brain functional network. Figure 3.6 below illustrates the entire process flow.

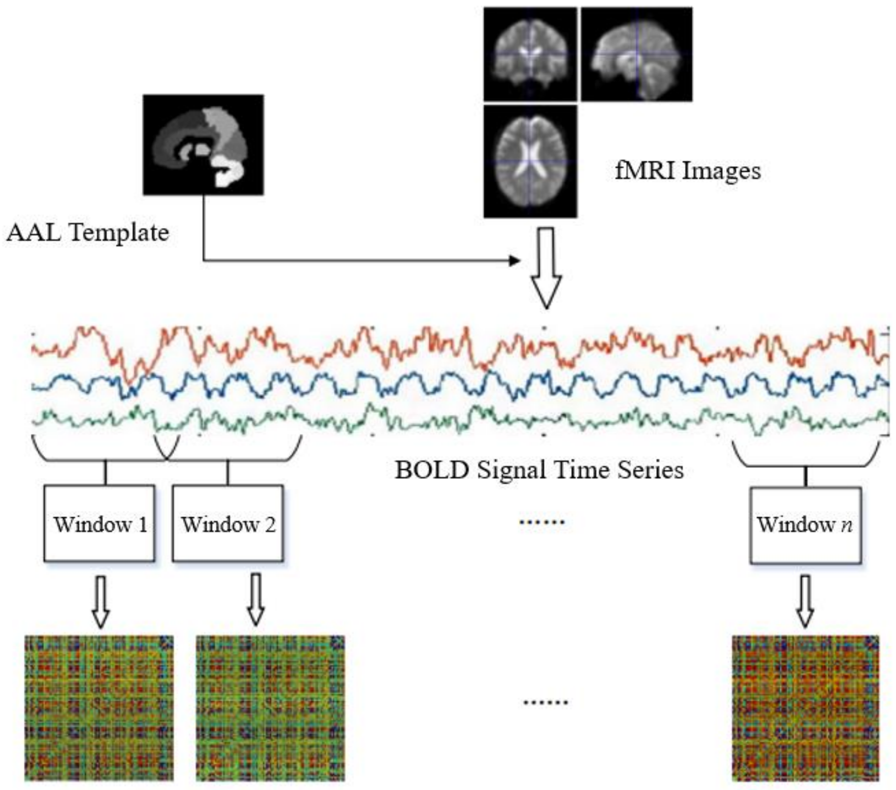


Fig. 3.6 Building dynamic brain function network based on sliding window

3.3. Summary

This chapter presents the process of constructing a dynamic brain functional network, which consists of two main parts. The first part involves preprocessing of fMRI images to correct any errors of the raw data. The second part involves constructing the dynamic brain network, which is achieved by processing the time series of each brain region using a sliding time window. This method segments the complete time series into several overlapping time series, which are then used to generate the dynamic brain functional connection network.

Chapter 4 Aided Alzheimer's Diagnosis Based on Evolving Graph Neural Network

Prior to conducting aided diagnosis for Alzheimer's disease (AD), it is essential to extract distinctive information from the subjects. This experiment utilizes complex network theory to analyze the topological structural measurement indicates of the network. By employing the relevant formulas, measurement indicates for all brain regions are computed as characteristic data representing the respective regions. Subsequently, the extracted features are integrated with an evolving graph neural network to accomplish the AD aided diagnosis.

4.1. Analysis of Topological Characteristics of Dynamic Brain Function Network

In this research paper, we employ topological structure metrics to extract brain region features from brain functional connection matrices generated by different sliding time windows of the same test sample. To accomplish the task of feature extraction, we analyze the significant metrics of brain topology. In this section, we will initially make detailed analysis among topological metrics and then select a variety of pre-selected features.

4.1.1. Extraction of Topological Structural Measurement Indicators from Brain Functional Networks

Typically, graph theory is employed to study and describe complex networks. Any complex network can be represented as a graph $G(N, K)$, consisting of a set of nodes N and a set of edges K. Each edge in the edge set K connects two nodes from the node set N. When the edges between nodes in a graph lack values or directions, the graph is referred to as an undirected network. Conversely, if the edges formed by nodes in a graph possess directions, the graph is referred to as a directed network.

The commonly utilized approach for representing graphs is through the use of adjacency matrixes, also known as binary matrixes. Hence, this study employs the adjacency matrix as the basis for constructing the brain functional network for subsequent analysis. According to the principles of complex network theory, each brain region within the brain functional network is considered a node, while the connections between brain regions are regarded as edges. If a network consists of n nodes, its adjacency matrix will have a size of $n \times n$. If an element $K(i, j)$ of the adjacency matrix equals 1, it signifies a direct connection between nodes i and j. Conversely, if $K(i, j)$ equals 0, it implies the absence of a direct edge between nodes i and j, indicating no direct connection between the two nodes. The resulting brain functional network is represented by an undirected and unweighted adjacency matrix.

This research paper describes the node attributes utilized in an undirected and unweighted network $G(N, K)$ as follows:

Local Efficiency: Local efficiency signifies the local capacity of a network to integrate information, encompassing the integration of connections between a given node and its neighboring nodes [20]. It serves as a measure to assess the effectiveness of information exchange within or near the immediate neighborhood of a specific node. A higher local efficiency implies a greater level of interconnectivity and efficient communication within the node's neighborhood, highlighting the resilience and robustness of the local network structure. Conversely, a lower local efficiency indicates suboptimal information flow in the nearby vicinity. The local efficiency for Node i is:

$$E(i) = \frac{1}{N_{G_i}\left(N_{G_i} - 1\right)} \sum_{j \neq k \epsilon G_i} \frac{1}{l_{j,k}} \tag{4.1}$$

Here, G_i denotes a graph consisting of all the neighboring nodes of node i. The symbol $l_{j,k}$ represents the shortest path length, which corresponds to the minimum distance between two distinct neighboring nodes j and k, which are connected to node i.

Degree: Degree serves as an indicator to describe the attributes of nodes and plays a significant role as a local feature. In the context of an undirected network, the degree of a node refers to the count of edges directly connected to that particular node. In a simple graph devoid of self-loops or multiple edges, the degree of a node also represents the number of neighboring nodes directly linked to it within the network. This measure not only elucidates the node's relationships with other nodes but also highlights its position and role within the network G. Therefore, degree can be employed as a criterion to determine whether a node functions as a network hub. The calculation formula is:

$$D_i = \sum a_{ij} \tag{4.2}$$

Where a_{ij} represents the number of direct connections between node i and node j in the network.

Degree Centrality: Degree centrality is a metric used to quantify the number of connections a node has within a network. It is derived by calculating the degrees of individual nodes. Nodes exhibiting elevated degree centrality possess a greater abundance of direct connections, often signifying their status as core nodes with heightened impact and capacity for disseminating information.

In contrast, degree refers to an attribute that characterizes nodes by denoting the quantity of their direct connections. Degree centrality, on the other hand, characterizes the position and influence of nodes within the network structure, serving as an index calculated based on node degrees.

$$DC_i = \frac{D_i}{n-1} \tag{4.3}$$

Where D_i represents degree of node i and n represents the number of nodes in the whole graph.

Betweenness Centrality: Betweenness Centrality is a measure used to assess a node's level of intermediation in a network. If a node serves as a crucial intermediary in the shortest paths between various nodes, its betweenness centrality is higher.

This centrality possesses several significant characteristics. Firstly, it aids in identifying nodes that play vital mediation roles in the network, serving as crucial bridges for information transmission, resource flow, and cooperation. Secondly, it quantifies a node's contribution to network connectivity. Nodes with high betweenness centrality often connect distinct communities or subgraphs, facilitating the dissemination and exchange of information within the network. The calculation formula is:

$$BC_i = \sum_{s \neq i \neq t} \frac{n_{st}^i}{g_{st}} \tag{4.4}$$

Where g_{st} represents the number of all the shortest distance paths between nodes s and t that are not directly connected. n_{st}^i represents the number of paths including node i among the g_{st} shortest distance paths.

Closeness Centrality: Closeness Centrality measures how closely connected a node is to other nodes in a network. It is calculated based on the shortest path length between a node and others, reflecting the node's proximity to them.

Closeness centrality has important properties. Firstly, it assesses the influence and accessibility of nodes in the network. Nodes with high closeness centrality are easily reached by others, enabling their influence to spread quickly across the entire network. Secondly, it gauges nodes' contribution to network connectivity. Higher closeness centrality implies a greater impact on network connectivity when nodes are removed or fail. The calculation formula is:

$$CC_i = \frac{n-1}{\sum_{j=1}^{n} l_{i,j}} \tag{4.5}$$

Here, n denotes the number of nodes in the whole graph G. The $l_{i,j}$ represents the shortest path length between a distinct neighboring node j and i.

PageRank Centrality: PageRank is an algorithm employed by Google for its search function, which assesses the significance of each webpage to provide more precise search results based on relevance and importance when searching for keywords. The PageRank of a webpage is determined through iterative calculations of the PR values of all pages that link to it via hyperlinks. A higher PR value indicates greater importance, while a lower PR value suggests a lack of significance [21]. When a node possesses a relatively high PR value, it signifies a prominent status within the network.

Consider four pages designated as A, B, C, and D, respectively. For example, if pages B, C, and D all contain hyperlinks pointing to page A, then the PageRank value of page A will be determined by summing the PageRank values of pages B, C, and D.

$$PR(A) = PR(B) + PR(C) + PR(D) \tag{4.6}$$

Suppose there exist hyperlinks to page A, excluding those from pages B, C, and D. Additionally, page B can also access page C through hyperlinks, and page D can access pages A, B, and C through hyperlinks. According to the given criteria, each page contributes a total score of 1 when assigning scores. Consequently, page B has a score of 0.5 towards both page A and page C. By extension, the PR value of page D towards page A amounts to only 0.33.

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1} + \frac{PR(D)}{3} \tag{4.7}$$

Finally, multiply by a coefficient, and since pages without outward links pass a PR value of 0, give each page a minimum value:

$$PR(A) = \frac{(1-d)}{n} + d \times \left(\frac{PR(T_1)}{D_{T_1}} + \frac{PR(T_2)}{D_{T_2}} + \cdots + \frac{PR(T_{n-1})}{D_{T_{n-1}}} \right) \tag{4.8}$$

Here, $PR(A)$ represents the PageRank value of node A , $PR(T_i)$ represents the PageRank value of any node except node A. D_{T_i} accounts for the degree of node T_i. The damping factor, denoted as d, represents the probability of a user reaching a page and subsequently continuing to browse backward at any given time.

The PageRank value of each node is derived through the superposition and accumulation of the PageRank values from other nodes. Initially assigning each node a PR value, subsequent iterations are performed to calculate the PR value of each node continuously. Eventually, these nodes' PageRank values reach a state of gradual stability. The importance of nodes is determined by evaluating which nodes possess higher PageRank values.

K-shell: The K-shell technique represents a network analysis methodology within graph theory employed for the purpose of discerning and partitioning the central nodes within intricate networks. The analysis entails a systematic elimination of interconnected edges connecting nodes within the network. This sequential progression hinges on the concept of node degree, namely the count of edges linked to a particular node. Initially, nodes with a degree of 1 are systematically removed, followed by the subsequent elimination of nodes with degree of 2, and so forth, until either no nodes remain or all nodes have been stripped away.

The application of K-shell analysis proves valuable in uncovering the core structure and pivotal nodes within a network. It finds widespread utility in comprehending and studying the topological structures, information dissemination, and node influence intrinsic to complex systems.

The flow of K-shell parsing is shown in Figure 4.1. In the process of defining K-shell, the green node with degree value of 1 and its connected edges are removed, and the Ks value of green nodes is 1; The remaining topological graph composed of blue and red nodes is named 2-core; Then remove the red nodes with degree value of 2, and the Ks value of the red nodes is 2; Finally, the graph formed by the remaining nodes is named 3-core, and so on, the Ks value of the blue nodes can be calculated.

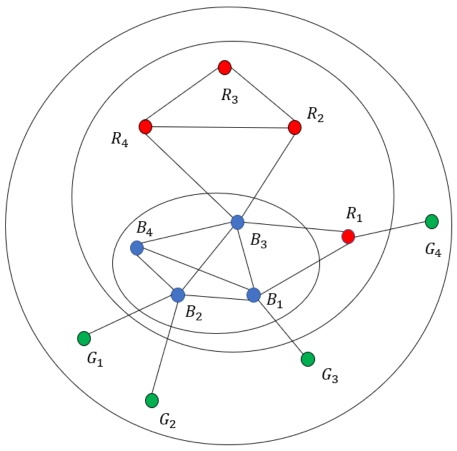


Fig. 4.1 K-shell analysis diagram

4.2. Evolving Graph Neural Network Design

4.2.1. Evolving Graph Convolutional Network

Evolving Graph Convolutional Networks (Evolving GCN) is a neural network architecture specifically designed to handle dynamic and evolving graph-structured data. It addresses the challenge of learning from graphs that change over time, allowing the model to capture temporal dynamics and adapt to evolving graph structures.

At its core, Evolving GCN [22] consists of three main components: graph aggregation, temporal convolution, and memory-augmented neural networks.

Graph Aggregation: In the graph aggregation step, Evolving GCN combines information from the current graph snapshot and the historical graph states, which aggregates node features and neighborhood information from the current time step, along with features and connectivity patterns from previous time steps. By incorporating historical graph information, the model can capture evolving relationships and dependencies between nodes over time.

Temporal Convolution: After graph aggregation, Evolving GCN applies temporal convolutions to capture temporal dependencies in the data. Temporal convolutions involve convolving node features across different time steps, allowing the model to capture temporal patterns and changes in the graph structure. This enables the model to adapt and update node representations as the graph evolves.

Memory-Augmented Neural Networks: Evolving GCN leverages memory-augmented neural networks to store and retrieve historical information from previous time steps. The model maintains a memory bank or recurrent neural network to retain past graph states and incorporate them into the aggregation and convolution processes. This memory mechanism enables the model to utilize historical context and improve its understanding of the evolving graph dynamics.

Training Evolving GCN involves a combination of supervised learning and graph-level loss functions. The model optimizes the parameters of the evolving graph convolutional layers to minimize prediction errors or maximize the graph-level objectives, depending on the specific task (e.g., node classification, link prediction).

Evolving GCN has demonstrated promising results in various applications involving dynamic graph-structured data, such as social network analysis, dynamic knowledge graphs, and temporal recommendation systems. By capturing temporal dependencies, adapting to changing graph structures, and making accurate predictions or classifications in dynamic environments, Evolving GCN enables effective analysis and modeling of dynamic graph domains.

This convolution process is similar to a perceptron but includes an additional step called neighborhood aggregation, which is inspired by spectral convolution. In each layer of the GCN, at time t, the l-th layer takes the adjacency matrix A_t and the node embedded matrix $H_t^{(l)}$ as input. It then uses a weight matrix $W_t^{(l)}$ to update the node embedding matrix, resulting in the output matrix $H_t^{(l+1)}$. Figure 4.2 give a schematic illustration of the whole convolution process of evolving GCN. Mathematically, this can be expressed as follows:

$$H_t^{(l+1)} = GCONV\left(A_t\,, H_t^{(l)}, W_t^{(l)}\right) := \ \sigma\left(\widehat{A_t} H_t^{(l)} W_t^{(l)}\right) \tag{4.9}$$

Where $\widehat{A_t}$ is a normalization of A_t defined as: $\hat{A} = \widetilde{D}^{\frac{-1}{2}}\tilde{A}\widetilde{D}^{\frac{-1}{2}}$, $\tilde{A} = A + I$, $\widetilde{D} = diag\left(\sum_j \widetilde{A_{ij}}\right)$. σ is the activation function for the output layer. The initial embedding matrix is derived from the node features, denoted as $H_t^{(0)} = X_t$. Assuming L layers of graph convolutions, the output layer can have two interpretations for the activation function σ: (1) If σ is considered as the identity function, $H_t^{(L)}$ contains high-level representations of the graph nodes that have been transformed from the initial features. (2) Alternatively, if σ is the softmax function used for node classification, $H_t^{(L)}$ represents the prediction probabilities for each node.

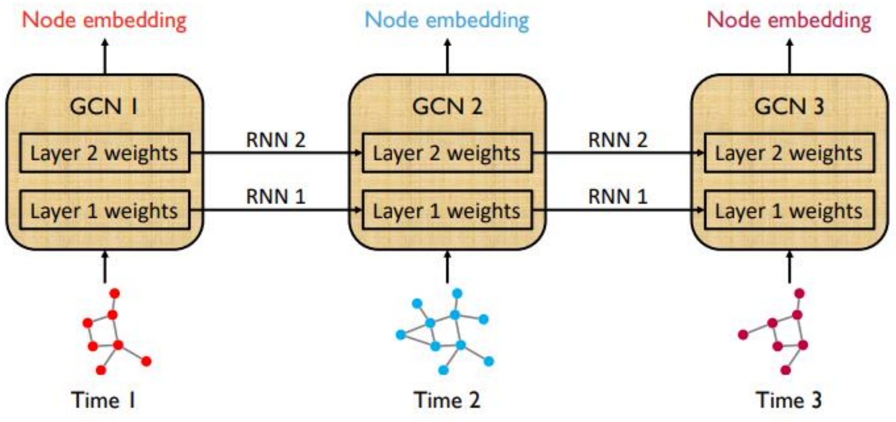


Fig. 4.2 Schematic illustration of Evolving GCN

EGCU-H Weight Evolution: The core of the proposed approach involves updating the weight matrix $W_t^{(l)}$ at time t using both current and historical information. This need can be effectively addressed by employing a recurrent architecture. We can treat $W_t^{(l)}$ as the hidden state of the dynamical system. We use a gated recurrent unit (GRU) to update the hidden state upon time-t input to the system. The input information naturally is the node embeddings $H_t^{(L)}$.The formula is as followed:

$$\overbrace{\underbrace{W_t^{(l)}}_{hidden\ state}}^{GCN\ weights} = GRU\left(\overbrace{\underbrace{H_t^{(L)}}_{input}}^{node\ embeddings}, \overbrace{\underbrace{W_{t-1}^{(l)}}_{hidden\ state}}^{GCN\ weights}\right) \tag{4.10}$$

Through the Evolving GCN network, the input consists of a continuously evolving brain network $G = (G_1, G_2, G_3, \dots, G_s)$, where s denotes the total count of snapshots capturing the dynamic functional brain network, and with the increase of integrated brain network images, our network will continuously integrate the node information of G_i into $W_t^{(l)}$ of the model.

The loss function utilized in this context is the cross-entropy function, denoted as $l(z, y) = -\frac{1}{N}\sum_{n=0}^{N-1}(y_n \log(z_n) + (1 - y_n)\log(1 - z_n))$, which N takes into account the number of individuals, the probability of predicting an individual z_n as a positive example, and the label of the individual y_n.

4.2.2. Evolving Graph Neural Network Based on GraphSAGE Algorithm

The key challenge in incorporating time series information into the process of node feature fusion within a graph neural network model. GraphSAGE (Graph Sample and Aggregation) is a deep learning algorithm designed for graph data, specifically aimed at learning node representations. It achieves this by sampling and aggregating the neighboring nodes of each target node, enabling effective representation learning and prediction tasks on graphs. The fundamental principle revolves around aggregating the neighbor information to generate node representations.

To elaborate, GraphSAGE algorithm selects a subset of neighbors for each node and combines their features to create a new representation for the target node. This aggregation process can be achieved through various techniques such as pooling, averaging, or max-pooling.

In the context of a graph convolution model, one of the primary concerns is how to effectively incorporate temporal aspects, such as time series information, into the node feature fusion process. This entails addressing the challenge of integrating evolving features into the aggregation operation of GraphSAGE, enabling the model to capture the evolving dynamics and exploit them for improved representation learning [23]. Therefore, we should design strategy to aggregate the information of neighbor nodes on neighbor slices and node features on current slices. The overall process of graph neural network is shown as Figure 4.3.

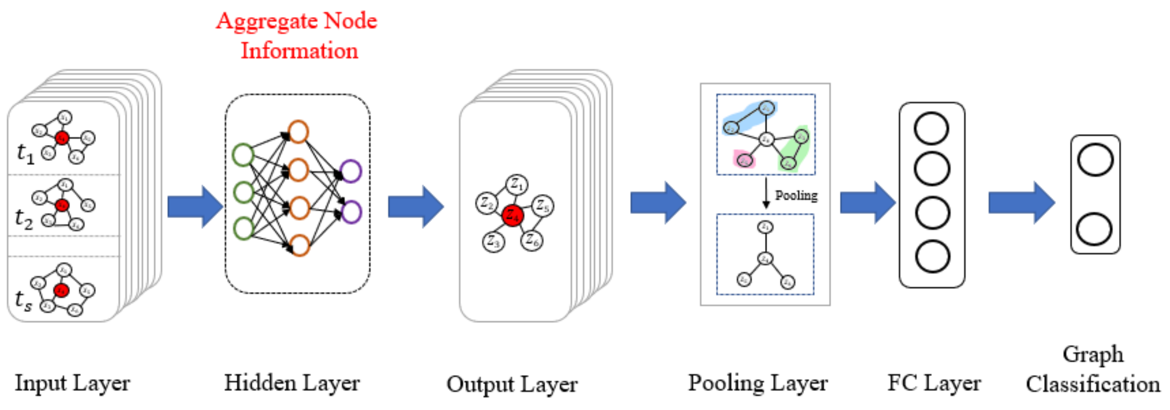


Fig. 4.3 Graph neural network based on GraphSAGE algorithm

An input is given of an evolving brain network $G = (G_1, G_2, G_3, \dots, G_s)$ is input, where s represents the number of snapshots of the dynamic functional brain network. Each snapshot, denoted as $G_m = (V_m, E_m)$, consists of a node set V_m and an edge set E_m. The node features of G_m are represents as $X_m = \{x_{v_m}, \forall v_m \in V_m\}$, and the node feature X denotes the clustering coefficient of all nodes and the brain region to which each node belongs.

As the first step, we sample the neighboring nodes of each node in network G, considering both the previous and next time slices. The snapshot containing the corresponding node is referred to as G_m, while the snapshots before and after G_m, while the snapshots before and after G_m are termed as neighboring snapshots, denoted as G_{m-1} and G_{m+1}, respectively. We proceed to sample the k-order neighboring nodes from these three snapshots. Let's assume the number of nodes to be sampled is n. If n outnumbers the number of available neighbor nodes, we perform sampling with replacement until n nodes are obtained. Conversely, if n is smaller than the number of neighbor nodes, we conduct non-replacement sampling. The sampled nodes from these three snapshots are respectively denoted as $N_{v_{m-1}}$, N_{v_m}, $N_{v_{m+1}}$.

Next, the information from the neighboring nodes and adjacent time slices is consolidated using an aggregation function to facilitate evolving information aggregation. The neighbor nodes of a specific node v in the mth snapshot $G_m$is expressed as:

$$h'_{N_v} = \left\{h_{N_{v_{m-1}}}, h_{N_{v_m}}, h_{N_{v_{m+1}}}\right\} \tag{4.11}$$

Where $h_{N_{v_{m-1}}}$, $h_{N_{v_m}}$, $h_{N_{v_{m+1}}}$represent the neighbor nodes of snapshots at time $m-1$,m and $m+1$ respectively.

The aggregation formula is defined as:

$$h^k_{N_{v_m}} = AGGREGATE\left(h^{k-1}_{N_{v_{m-1}}}, X_{m-1}, h^{k-1}_{N_{v_m}}, h^{k-1}_{N_{v_{m+1}}}, X_{m+1}\right) \tag{4.12}$$

The value of k in this formula corresponds to both the network's layer count and the maximum order of adjacent nodes that each node can aggregate. For instance, if k equals to 2, every node can derive its own embedding representation by considering information from its second-order adjacent nodes at most. Moreover, X_{m-1} and X_{m+1} indicate the node features at time $m-1$ and $m+1$, respectively. $AGGREGATE(.)$ is the aggregation function. After the embedding process, the node features are represented as $X^k_m = \sigma\left(W^k \cdot CONCAT\left(X^{k-1}_m, h^k_{N_{v_m}}\right)\right)$, where $CONCAT(\cdot)$ is the concatenation function. $\sigma(\cdot)$ is the activation function. W^k is the weight coefficient, $\forall k \in \{1, \dots, K\}$ and k is the order of neighbor nodes.

Ultimately, we derive the vector representation of each node within the dynamic functional brain network, which proves useful for subsequent tasks. The convolved functional brain network undergoes pooling, followed by graph classification, enabling the facilitated diagnosis of Alzheimer's disease (AD). The evolving graph convolutional network algorithm is presented in Algorithm 1.

Algorithm1: Evolving graph convolutional network based on GraphSAGE algorithm

Input: The dynamic brain network $G = (G_1, G_2, G_3, \dots, G_s)$; input features $X_m = \{x_{v_m}, \forall v_m \in V_m\}$; sampling hop k; weight matrices W^k,$\forall k \in \{1, \dots, K\}$; activation function σ; aggregation function $AGGREGATE(.)$

Output: Vector representation z_{v_m} for all $\forall v_m \in V$.

```
$X_{v_m}^0 \leftarrow x_{v_m}, \forall v_m \in V$;
$\boldsymbol{for}\ k = 1 \dots k\ \boldsymbol{do}$
    $\boldsymbol{for}\ v_m \in V\ \boldsymbol{do}$
        $h_{N_{v_m}}^k \leftarrow AGGREGATE\left(h_{N_{v_{m-1}}}^{k-1}, X_{m-1}, h_{N_{v_m}}^{k-1}, h_{N_{v_{m+1}}}^{k-1}, X_{m+1}\right)$
        $X_m^k \leftarrow \sigma\left(W^k \cdot CONCAT\left(X_m^{k-1}, h_{N_{v_m}}^k\right)\right)$
    $\boldsymbol{end\,for}$
    $X_{v_m}^k \leftarrow \frac{X_{v_m}^k}{\|X_{v_m}^k\|_2}, \forall v_m \in V$
$\boldsymbol{end\,for}$
$z_{v_m} \leftarrow X_{v_m}^k$
```

The loss function utilized in this context is the cross-entropy function, denoted as $l(z, y) = -\frac{1}{N}\sum_{n=0}^{N-1}(y_n \log(z_n) + (1 - y_n)\log(1 - z_n))$, which N takes into account the number of individuals, the probability of predicting an individual z_n as a positive example, and the label of the individual y_n.

The aggregation function serves the purpose of updating and iterating node information, making its selection crucial for the performance of the evolving graph convolutional network. It encompasses three primary types of aggregators.

Max Aggregator: Considering the features of neighboring nodes and selecting the maximum value as the representation information for the node being updated.

Mean Aggregator: Neighboring node features are averaged to create a representative summary of neighbor node information.

LSTM Aggregator: Neighbor nodes are randomly ordered, and their information is obtained through an LSTM network.

4.3. Performance Analysis

In the process of classification, accuracy, sensitivity, specificity, and ROC curve are commonly used to evaluate the performance of classifiers. Therefore, this study evaluates the proposed method's performance in terms of these aspects. The classification results can be categorized into the following four scenarios:

(1) True Positive (TP): Predicted as positive and actually positive.

(2) False Positive (FP): Predicted as negative and actually positive.

(3) True Negative (TN): Predicted as negative and actually negative.

(4) False Negative (FN): Predicted as positive and actually negative.

In the medical direction, the meanings of the above four situations are shown in Table 4.1 below:

Table 4.1 Meanings of parameters in Evaluation indices

Parameter name	Parameter meaning
TP	The number of cases accurately diagnosed as being diseased.
TN	The number of cases accurately diagnosed as not being diseased.
FN	The number of cases of the disease that were not accurately diagnosed.
FP	The number of cases not having the disease that could not be accurately diagnosed.

Comparison is made between the results of classifying brain functional networks using the proposed method based on evolving graph neural networks and the actual conditions of the subjects' data. The evaluation of the proposed method for aided diagnosis based on dynamic networks is conducted using commonly used evaluation metrics for diagnostic systems. These evaluation metrics include sensitivity, positive predictive value, diagnostic accuracy, negative predictive value, specificity, etc. The specific meanings of these metrics are explained below:

Table 4.2 Evaluation indices

Parameter name	Parameter meaning
Accuracy	$(TP+TN)/(TP+TN+FP+FN)$
Sensitivity	$TP/(TP+FN)$
Specificity	$TN/(TN+FP)$
Positive predictive value	$TP/(TP+FP)$
Negative predictive value	$TN/(TN+FN)$

The two proposed algorithms, Evolving Graph Neural Network based on GraphSage and Evolving Graph Convolutional Network, both demonstrate good performance. Under the training set and testing set ratio of 0.8:0.2, the following experimental results can be obtained as Table 4.3.

Table 4.3 Experimental results

Evaluation	EGNN-GraphSage	EGCN
Accuracy	94.5%	95.3%
Sensitivity	89.6%	90.3%
Specificity	92.4%	91.4%
Positive predictive value	93.5%	91.7%
Negative predictive value	90.3%	88.3%

By comparing the ROC curves of AD (Alzheimer's Disease) and NC (Normal Control) using the Evolving GraphSage neural network based on dynamic brain networks and the normal GraphSage neural network based on static brain networks, we can observe the superior classification performance. Figure 4.4 illustrates the ROC curves for the two methods.

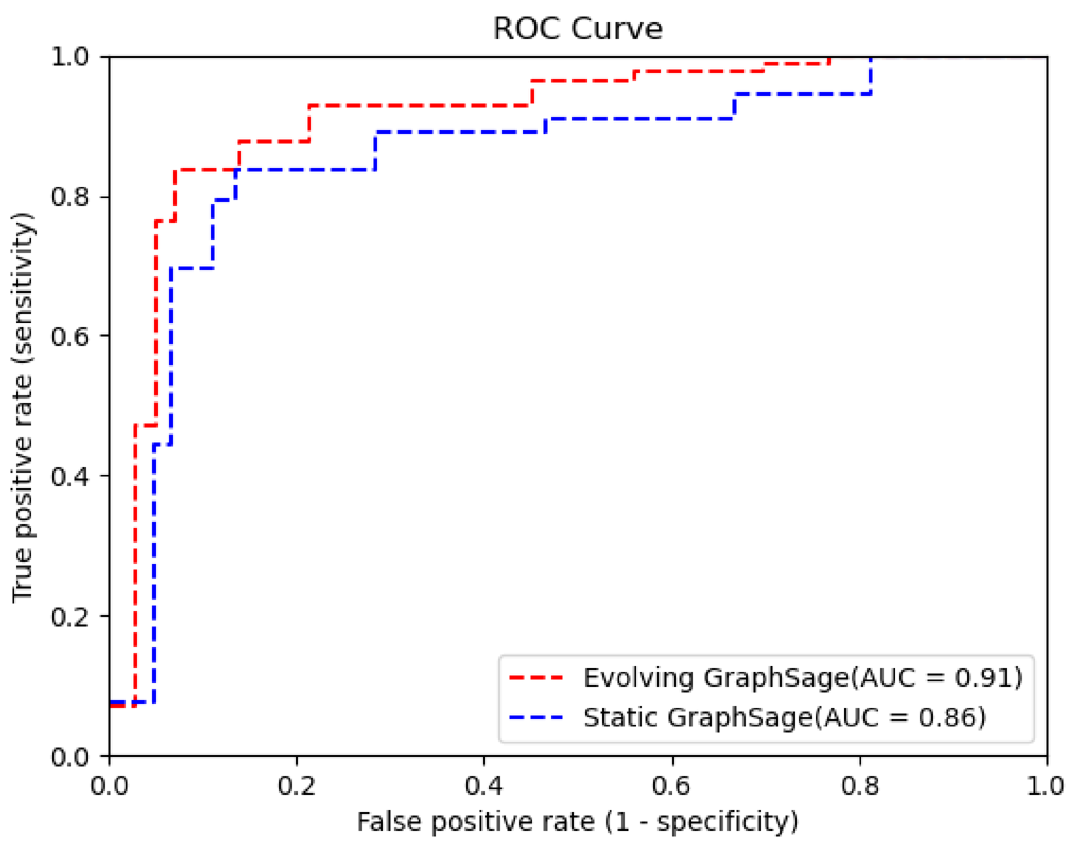


Fig.4.4 Comparison of ROC curves of two methods (1)

In the comparative experiments between AD (Alzheimer's Disease) and NC (Normal Control) data, the Evolving GraphSage model based on dynamic brain networks achieves an AUC value of 0.91, while the GraphSage model based on static brain functional networks achieves an AUC value of 0.86. Both methods use the same test samples and employ a similar aggregate method based on relative mean clustering. The results indicate that the method based on dynamic brain networks provides superior diagnostic performance for assisting in the diagnosis.

By comparing the classification results between AD and C datasets, the EGCN (Evolving Graph Convolutional Network) method, based on constructing dynamic brain networks, achieves an AUC value of 0.92, while the GCN (Graph Convolutional Network) method, based on constructing static brain networks, achieves an AUC value of 0.85. Figure 4.5 illustrates the ROC curves for the two methods.

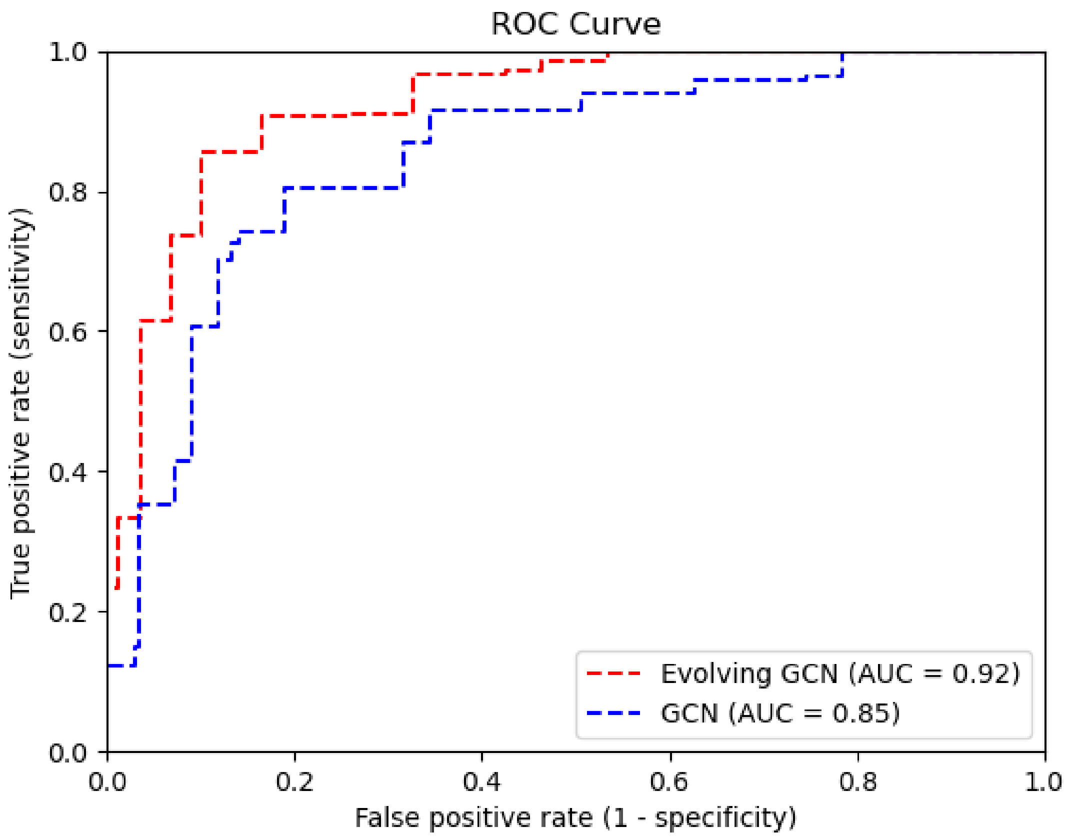


Fig.4.5 Comparison of ROC curves of two methods (2)

The experimental results on the same dataset reveal that the method based on dynamic brain functional networks achieves higher accuracy in assisting diagnosis compared to the static brain network method. When considering other evaluation metrics such as sensitivity and specificity, it is evident that the dynamic approach outperforms the static one. This can be attributed to the fact that dynamic brain networks better capture the temporal topology information of the brain. During the process of node information fusion, the dynamic method obtains the information from different brain nodes at different time points for the same example. In contrast, the static method can only access the information from other brain nodes at a single time point. As a result, the dynamic approach achieves superior performance in aided diagnosis.

4.4. Summary

In this section, we present an in-depth exploration of key indicates in the network metric analysis, thereby offering a solid theoretical foundation for the extraction of node information from brain networks. Subsequently, we delve into two advanced evolving neural network models, providing a comprehensive overview of their associated theoretical algorithms and internal implementation details. Furthermore, we discuss the integration of brain network information across multiple evolving nodes in the experimental phase, elucidating the methodology for accomplishing aided diagnosis through experimental procedures. The experimental results of assisted diagnosis based on evolving brain networks were presented lastly.

Chapter 5 System Requirements Analysis

Requirement analysis is employed to identify the business needs of users, the problems to be addressed, or the objectives to be achieved in the future. It is essential for software developers and users to reach a consensus and clarify the development project. The accuracy, standardization, and comprehensiveness of requirement analysis play a vital role in the subsequent stages of design and development, ultimately determining the quality of the final product. A meticulous and comprehensive requirement analysis serves as a fundamental pillar for successful development, occupying a pivotal position throughout the entire process. Its impact on project success cannot be overstated, as it sets the stage for effective decision-making, streamlined development, and the attainment of desired outcomes.

5.1. Functional Requirements Analysis

The system is mainly divided into three parts: user service module, model training module and aided diagnosis module. The user roles in the system are categorized into three levels: ordinary users with minimum privileges, advanced users with higher privileges, and Administrator users who have full access to all system functionalities, including the model training module and aided diagnosis module. Different user roles have different permissions. As shown in table 5.1.

Table 5.1 Permission Description

ID	User role	Permission Description
1	Ordinary user	Ordinary users have access to the registration function, personal prediction record search function, and aided diagnosis function. Additionally, ordinary users can perform basic functional integration of their prediction records. In summary, ordinary users possess a fundamental understanding of the system's capabilities.
2	Advanced user	Advanced users have the same full access to the model prediction functionality as ordinary users, along with the ability to view the test results of all users and perform comprehensive data integration. In addition, advanced users have the capability to leave messages for the system. In summary, advanced users have all the privileges to utilize the model for predictions and have complete access to prediction data, except for the training of the model.
3	Administrator user	The administrator possesses full privileges to thoroughly evaluate all system functionalities and has access to the

		model training module. Furthermore, the administrator has the capability to integrate and refine the model's prediction capabilities by analyzing training results under different parameters.

New users are required to utilize the registration function to create an account and log in before they can access the AAD system for model training. During the registration process, users need to provide a unique username, password, confirmed password, email address, gender, birthday and other areas of interest relevant to this system. Administrator users have the authority to manage registered user information and view users' login logs.

After logging in, ordinary users can access the aided diagnosis module, which leads them to the main panel. The main panel provides a feature to import test cases. Users can browse through all files in a specific folder on their local machine that have a particular file format. Once they have selected a test case file, they can choose the model and proceed to the next step. In the subsequent interface, the system retains all the user-selected information. Upon clicking the prediction button, the system initiates the training process. As a binary classification model, the system predicts the percentage likelihood of a patient being normal or having a disease. Furthermore, within the main interface, ordinary users can navigate to the records section to query all prediction records and perform data integration. The system incorporates fuzzy and precise searching methods. It allows filtering, sorting, deletion, alignment, and movement of individual data entries within the table. Additionally, users can export the entire visualization area of the interface as a CSV file. Through this design process, ordinary healthcare professionals can utilize the system for aided diagnosis of patients. They can leverage the database and frontend interface to accomplish medical information integration tasks.

In addition to the capabilities of ordinary users, advanced users have the functionality to leave messages within the system. All messages are stored in the data system, and system developers can utilize a dedicated database visualization platform for querying purposes. Furthermore, advanced users have full auditing privileges for prediction cases, enabling them to conduct comprehensive analysis by integrating and examining prediction results from both ordinary users and themselves.

Administrator users have full access to all system functionalities, except for the messaging feature. The introduction mainly focuses on the model training module. When administrator users choose the model training functionality, they can browse through a specific folder on their local machine to locate all files ending with a specific format. These files serve as the raw data for model training and can be imported into the system. Afterwards, administrators enter the model parameter tuning panel. Within this panel, administrators can select different model training options, feature combinations, iteration counts, and ratio of the test set. Once all the necessary information is selected, the model training process can commence. Besides, the database visualization panel then integrates and presents all the model training information in a unified manner. Administrators can utilize the domain sorting feature to select the optimal model based on specific criteria. The database visualization interface also enables sorting, alignment, filtering, deletion, and the ability to move individual data entries within the table. Moreover, administrators can export the entire visualization area of the interface as a CSV file.

In summary, the system effectively achieves complete functionality, user-friendly interface operations, and support for multiple users with different levels of permissions.

5.2. Function Description

The functions available to ordinary users, advanced users and administrator users are shown in the tables below. They are designed in User Manager Module, Message Manager Module, Model Training Module and Aided Diagnosis Module.

5.2.1. User Manager Module

For User Manager Module, use case diagram is designed and shown in Figure 5.1.

Because of various identities, we design different login jump logic according to different user names. After completing registration, users can enter different portals. The use case description for Login is shown in Table 5.2.

Table 5.2 Use case for Login

Term	Description
Use Case ID	01
Use Case Name	Login
Actor	ordinary users, advanced users and administrator users
Summary	User logs in the system.
Pre-conditions	User has registered an account in the system
Basic Flow	1. User enters the username and password 2. User presses login button. 3. The system check whether the username is existed. 4. According the identity of the user, the system will present different portal. 5. User finishes the Login process.

Exceptional Flow	3a. User input wrong username or password.
Post-condition	The user logged in to the system. The main portal shows welcome information uniquely concerning different users.

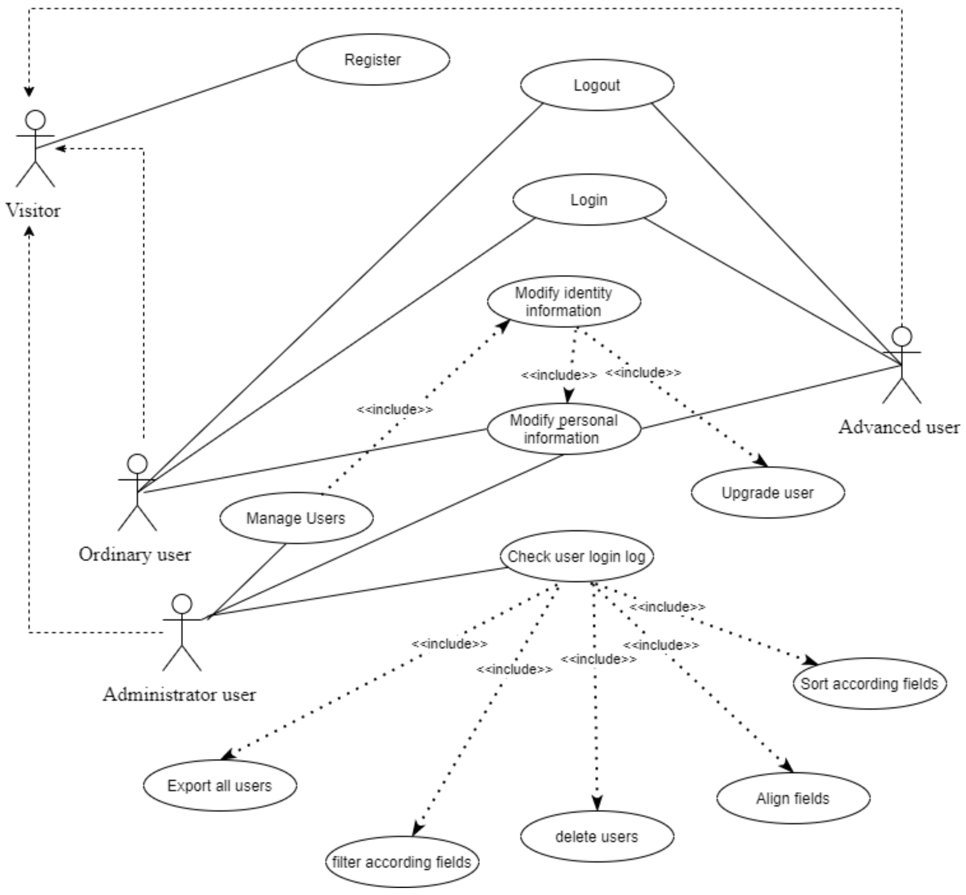


Figure 5.1 User manager use case diagram

When the visitor enters the system for the first time, he has no account and needs to register. The use case description for Registration is shown in Table 5.3.

Table 5.3 Use case for Registration

Term	**Description**
Use Case ID	02
Use Case Name	Registration
Actor	visitors
Summary	Visitor creates an action to use the account when they first enter the system and click the Register button.
Pre-conditions	Visitor has logged in to the system.

Basic Flow	1. Visitor clicks “Register” button. 2. Visitor comes into the registration portal. 3. Visitor fills in the corresponding registration information according to the form to complete the registration. 4. The front and back ends judge whether the filled information meets the requirements. 5. Visitor finishes the registration progress.
Exceptional Flow	4a. Visitor type a repeated username. System gives out warning and user can continue type different username. 4b. The user's password does not meet the required format. System gives out warning and user can continue input different passwords. 4c. The password entered by the user is different twice. System gives out a warning window and user can continue input different passwords.
Post-condition	Upon successful registration, the visitor is redirected back to the login interface, where the registered username information is automatically populated in the username input box.

The system provides different users to fill in their own information to change the function, the administrator user can modify any user's information at will. The use case description for Modify personal information is shown in Table 5.4.

Table 5.4 Use case for Modify personal information function

Term	**Description**
Use Case ID	03
Use Case Name	Modify personal information
Actor	ordinary users, advanced users and administrator users
Summary	Different User modifies user’s personal information
Pre-conditions	User has logged in to the system.
Basic Flow	1. User clicks on “Modify” button. 2. User inputs new information. If user want to modify password, he needs to input origin password. 3. User submits the information. 4. System will check whether the username and password meet the requirements. 5. System modifies user’s information. 6. User finishes the process.
Exceptional Flow	4a. Visitor type a repeated username. System gives out warning and user can continue type different username.

	4b. The user's password does not meet the required format. System gives out warning and user can continue input different passwords. 4c. The password entered by the user is different twice. System gives out a warning window and user can continue input different passwords.
Post-condition	Information of target user is modified.

Administrator is a specific administrator set at the beginning of the system, which has the ability to manage all users. You can modify the permissions of some users into the capabilities of managers. The use case description for Manage users is shown in Table 5.5.

Table 5.5 Use case for Manage users function

Term	**Description**
Use Case ID	04
Use Case Name	Manage users
Actor	administrator users
Summary	Administrator user can query the personal information filled in when all users register in the system.
Pre-conditions	Administrator use has logged in to the system.
Basic Flow	1. The administrator user enters the correct password information and logs in to the system successfully. 2. The front-end calls the database information, and displays all the user information completely in the interface. 3. Administrator users can select users and then jump to the modification portal to modify users' information.
Exceptional Flow	1a. Administrator user types a repeated username. System gives out warning and user can continue type different username.
Post-condition	Administrators enter the visual interface of user database to manage user data comprehensively.

5.2.2. Message Manager Module

For Message Manager Module, use case diagram is designed and shown in Figure 5.2.

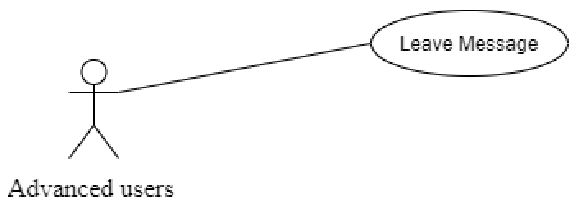


Figure 5.2 Message Manager use case diagram

Advanced users can provide information suggestions for the system, and will enter the database of the system, so as to facilitate the query of administrators. The use case description for Leave message is shown in Table 5.6.

Table 5.6 Use case for Leave message function

Term	Description
Use Case ID	05
Use Case Name	Leave message
Actor	advanced users
Summary	Advanced user writes a message in the message area and send the message to the database.
Pre-conditions	Advanced users successfully log in to their corresponding main portal interface, and then click the corresponding button keys to come to the Message interface.
Basic Flow	1. User clicks "Message" button. 2. User texts the message in the text field. 3. User submits the message. 4. System puts the message in the database. 5. User finishes the process.
Exceptional Flow	3a. User input message which length is over the maximum limitation. System gives out warning and return to the main page.
Post-condition	New message is input into database. System developers and administrator users can query the relevant Message

5.2.3. Model Training Module

For Model Training Module, use case diagram is designed and shown in Figure 5.3.

The first step of training model is to find the position of training data in PC. The use case description Search training dataset is shown in Table 5.7.

Table 5.7 Use case for Search training dataset function

Term	Description
Use Case ID	06
Use Case Name	Search training dataset
Actor	administrator users
Summary	Administrator user selects datasets in a specific format in the PC.
Pre-conditions	Administrator user successfully logs into the system and accesses the main portal.

Basic Flow	1. Administrator user fills in the input box of "RawPath" with the absolute path of the original data. 2. Administrator user sets the format requirements of the file. 3. Administrator user can click the "search" button to check whether the selected folder is correct. 4. Administrator user fills in the input box of "StorePath" with the absolute path of the original data.
Post-condition	Administrator user can click the "Import" button to start the data Import work.

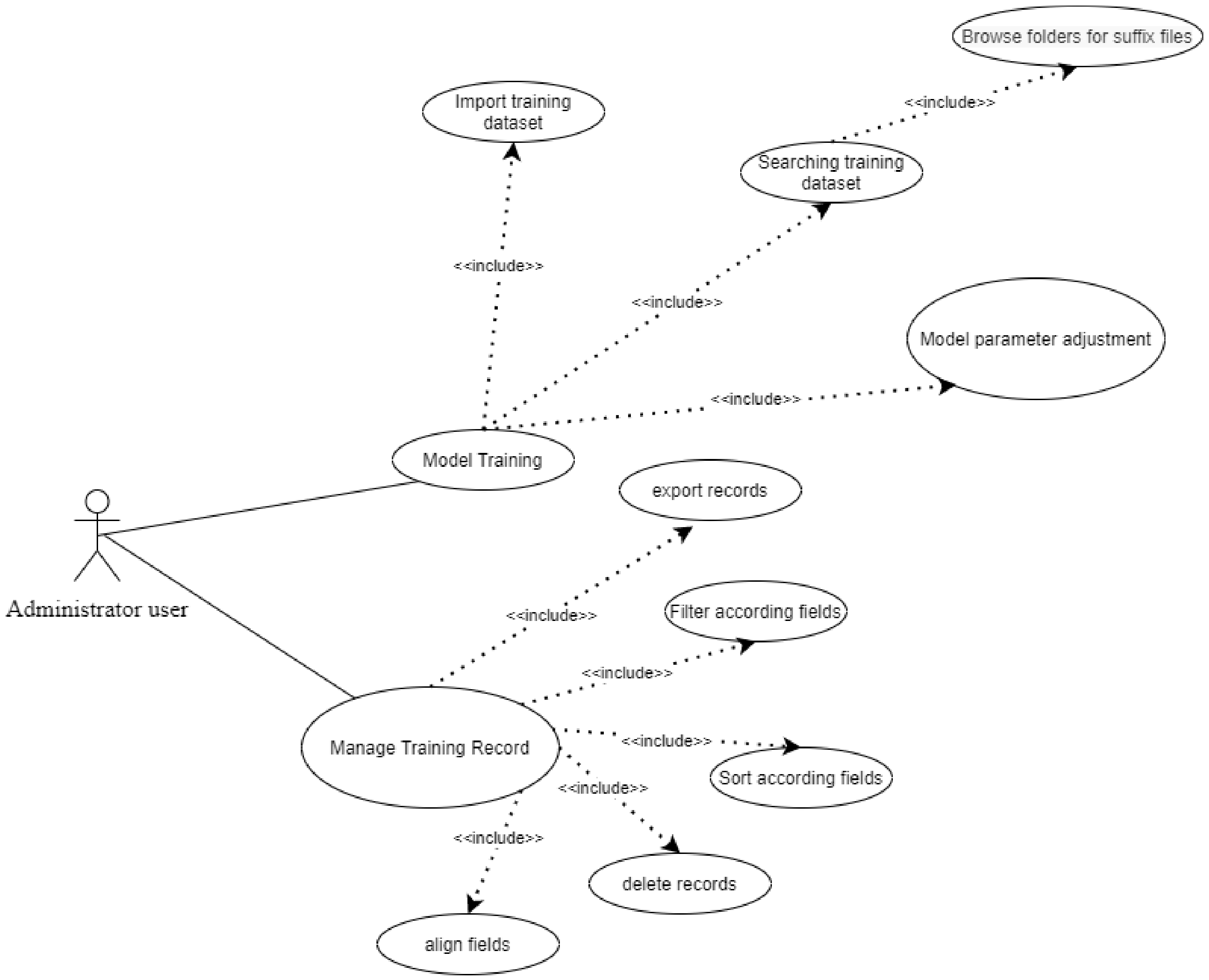


Figure 5.3 Model Training use case diagram

To import data, we need to call the data import function defined by Python, and use the library function InMemoryDataset of pytorch_geometric to complete the initial data processing. The use case description for Import training dataset function is shown in Table 5.8.

Table 5.8 Use case for Import training dataset function

Term	**Description**
Use Case ID	07
Use Case Name	Import training dataset
Actor	administrator users
Summary	Administrator user utilizes python library functions to complete the processing of the original data, and stores the data in the specified path.
Pre-conditions	Administrator user has logged in to the system.
Basic Flow	1. Administrator user completes the search for the training data. 2. Administrator user clicks the “Import” button. 3. After the system judges that the data import is correct, it can activate the "Next" button and enter the model parameter adjustment interface. 4. Administrator user finishes the process.
Exceptional Flow	3a. When the system query data has not been imported successfully, an error warning message will be displayed after clicking the "Next" button.
Post-condition	Administrator user can have access to the model parameter adjustment interface.

Model parameter adjustment portal is the most important interface for model training. By configuring different parameters, the training accuracy of the model changes. So that administrators can train the best model. The use case description for Model parameter adjustment is shown in Table 5.9.

Table 5.9 Use case for Model parameter adjustment function

Term	**Description**
Use Case ID	08
Use Case Name	Model parameter adjustment
Actor	Administrator users
Summary	Administrator user adjusts different parameters to complete model training.
Pre-conditions	Administrator user has logged in to the system.

Basic Flow	1. Administrator user completes the importing dataset operation and enters the parameter adjustment interface. 2. Administrator user adjusts all parameter information according to the information on the panel. 3. Administrator user clicks “Confirm” button. 4. The system automatically trains the dataset with the selected model. 5. Administrator user finishes the process.
Exceptional Flow	4a. When administrator user does not enter the model information correctly, omit the model, omit the feature, etc., the system will give a warning message.
Post-condition	The system will jump to the prediction result display interface.

The previous model training information of the administrator user will be stored in the database. By entering the model training record interface, the administrator can query the results of each model training. The use case description for Manage training record function is shown in Table 5.10.

Table 5.10 Use case for Manage training record function

Term	**Description**
Use Case ID	09
Use Case Name	Manage training record
Actor	administrator users
Summary	Administrator user manages the model training information to obtain the optimal value of the model parameters.
Pre-conditions	Administrator user has logged in to the system.
Basic Flow	1. Administrator user enters the main interface. 2. Administrator user clicks the "Record" button, and then enters the database display portal. 3. Administrator user can perform many different record operation functions.
Post-condition	Administrator user can carry out model training, analysis and integration functions.

5.2.4. Aided Diagnosis Module

For Aided Diagnosis Module, use case diagram is designed and shown in Figure 5.4.

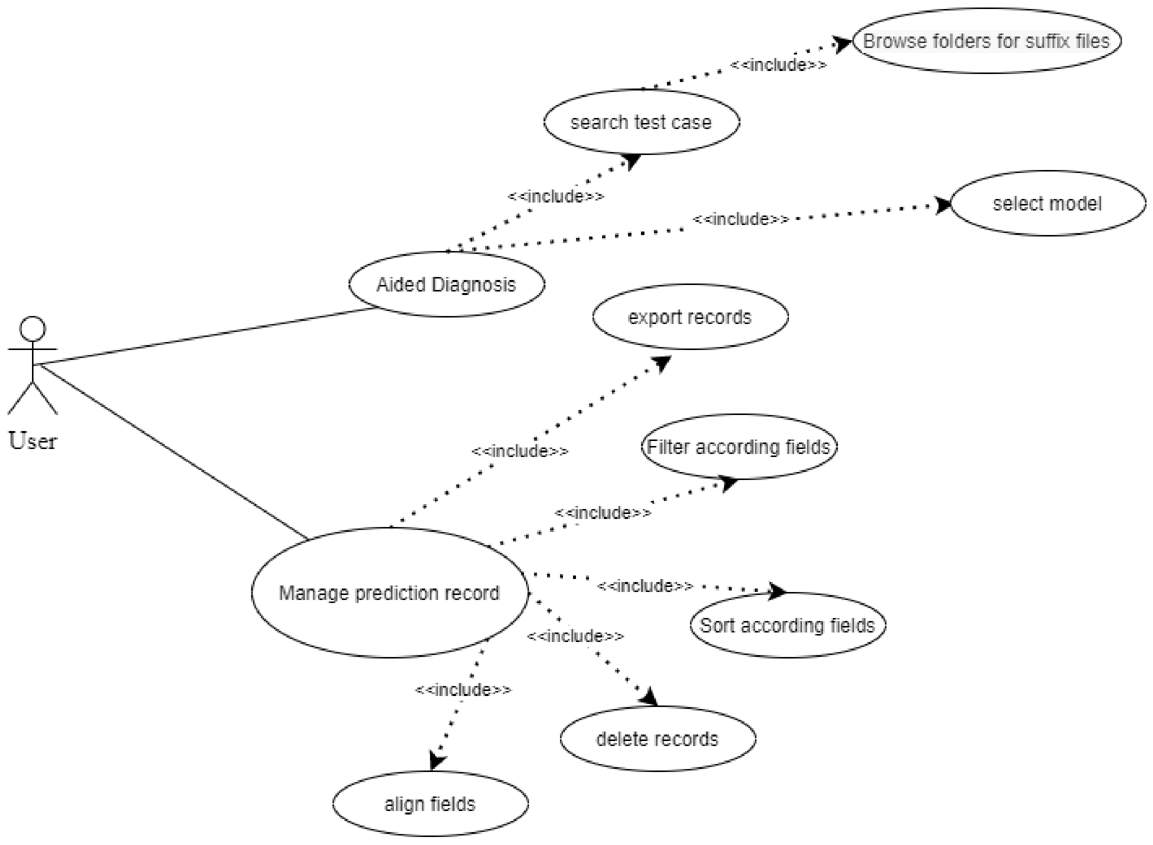


Figure 5.4 Aided Diagnosis Module use case diagram

Prediction is the process of inputting test cases into the system and using the trained model to realize aided diagnosis. The use case description for Aided Diagnosis is shown in Table 5.11.

Table 5.11 Use case for Aided Diagnosis

Term	**Description**
Use Case ID	10
Use Case Name	Aided Diagnosis
Actor	ordinary users, advanced users and administrator users
Summary	User passes the test samples to the diagnostic system, and the system gives the predicted results in percentage form.
Pre-conditions	User has logged in to the system.
Basic Flow	1. User fills in the path information of the test case on this machine in the portal. 2. User finds the folder in the specified path and selects a certain test case. 3. User clicks the "Next" button to enter the following interface.

	4. The system utilizes model to perform aided diagnosis on the selected test cases and gives the prediction results in the form of percentages. 5. User finishes the process.
Exceptional Flow	3a. If user does not select a test case, the system will give an error warning message indicating that there is no test case at present.
Post-condition	The system will enter the prediction result display interface to display the prediction information.

All prediction records of a certain user will be stored in the database. By entering the prediction record interface, the user can query the results of each case prediction result. The use case description for Manage prediction record function is shown in Table 5.12.

Table 5.12 Use case for Manage prediction records

Term	**Description**
Use Case ID	11
Use Case Name	Manage prediction records
Actor	ordinary users, advanced users and administrator users
Summary	User can have an access to authorized prediction records.
Pre-conditions	User has logged in to the system.
Basic Flow	1. User enters the main interface. 2. User clicks the "Record" button, and then enters the database display portal. 3. User can perform many different record operation functions.
Post-condition	User can carry out model training, analysis and integration functions.

5.3. Non-functional Requirements Analysis

5.3.1. Security Requirements

The digital landscape is intricate, encountering frequent instances of unauthorized entry and malicious infiltration. To ensure safeguarding of data integrity, the AAD system necessitates robust measures encompassing authentication, stringent access controls, and granular user permissions. These security imperatives address the multifaceted challenges prevalent in the network ecosystem, mitigating risks of data loss and fortifying against potential threats.

5.3.2. Reliability Requirements

The AAD system should be capable of handling errors or exceptions. Due to the user's operation against the process, the system should give reasonable reminders. This requires adding a variety of warning and prompt windows in the design process. The system should also be able to avoid data loss, which requires the use of a good and stable database as a support.

5.3.3. Scalability Requirements

The AAD system is designed to exhibit exceptional scalability. It incorporates a robust technical architecture and adheres to standard data interfaces, enabling seamless integration of new functionalities. Advanced algorithm libraries are utilized within the system to ensure optimal performance. Additionally, the AAD system aims to simplify data exchange with existing systems, reducing complexities and enhancing interoperability. It emphasizes the need for a flexible and adaptable framework that can effortlessly accommodate future enhancements and foster smooth collaborations with other systems.

5.3.4. Ease of Use Requirements

User experience is a critical aspect to consider in system design. The intuitiveness of the user interface significantly influences user satisfaction. The system should provide a streamlined and accessible user experience. It should cater to the needs of both regular users and administrators or maintenance personnel, ensuring a smooth onboarding process and efficient operations. AAD system is very simple in interface design, and uses a variety of error prompt operations, which is convenient for users to complete self-correcting experience.

5.3.5. Maintainability Requirements

AAD system need to reduce maintenance costs, improve maintenance efficiency, and can quickly adapt to changing needs, which means high maintainability.

5.4. Summary

This section delves into the comprehensive examination of the AAD system's prerequisites. Beginning with an exploration of the use case concept, it proceeds to provide an intricate elucidation of the software module's functional diagram. The document expounds upon the indispensable non-functional demands of the system, encompassing areas such as dependability, extensibility, user-friendliness, and serviceability. These deliberations serve as a guiding compass and the foundational bedrock for the subsequent stages of system design, ensuring a meticulous and professional approach to the development process.

Chapter 6 System Design

System design entails the process of devising a novel system that effectively aligns with the intended objectives and purpose, drawing upon the principles and methodologies of system science based on the outcomes of system analysis. It encompasses several critical design components, with a specific focus on interface design and database structure. This chapter expounds upon these two facets, laying the foundation for the subsequent chapters that will delve into the detailed design and implementation of system modules.

6.1. Interface Design

6.1.1. Login Interface Design

During the user login process, it is essential to capture the password and username details at the frontend, followed by a database query to validate the accuracy of the provided information and retrieve the corresponding authority details associated with it. The interface statement is shown as Figure 6.1.

Table 6.1 Login Interface Design Table

Name	Login		
Statement	Used for user login operations		
Request Protocol	TCP/IP		
Request Method	POST		
Request Format	JSON		
Return Format	JSON		
Description of the request parameters			
Name	Type	Constraint	Statement
username	varchar	Yes	The username of the user.
password	varchar	Yes	The password of the user.
Description of the return parameters			
Name	Type	Statement	
authority	varchar	The authority information of the user	

6.1.2. Register Interface Design

For visitor who is not registered initially, it is necessary to complete the registration operation in advance before entering the system smoothly. The design of this interface requires a lot of request data information. The interface statement is shown as Figure 6.2.

Table 6.2 Register Interface Design Table

Name	Register		
Statement	Used for visitor registration operation		
Request Protocol	TCP/IP		
Request Method	POST		
Request Format	JSON		
Return Format	JSON		
Description of the request parameters			
Name	Type	Constraint	Statement
username	varchar	Yes	The username of the visitor.
password	varchar	Yes	The password of the visitor.
email	varchar	No	The email of the visitor.
sex	int	Yes	The gender of the visitor.
interest	array	No	The interest point of the visitor.
birthday	datetime	Yes	The birthday of the visitor.
authority	varchar	Yes	This is automatically set as “normal”.
Description of the return parameters			
Name	Type	Statement	
None	None	None	

6.1.3. Training Model Interface Design

Training model refers to the record information generated after the user calls the model training. By recording the results of model training, it is convenient for administrator user to query the performance of the model. The interface statement is shown as Figure 6.3.

Table 6.3 Training Model Interface Design Table

Name	Training model
Statement	Used for recording administrator training record
Request Protocol	TCP/IP
Request Method	POST

Request Format	JSON		
Return Format	JSON		
Description of the request parameters			
Name	Type	Constraint	Statement
username	varchar	Yes	The username of the user.
request_Id	varchar	Yes	The unique id of the request of training model.
selected_model	varchar	Yes	The model information of the request.
is_shuffle	int	Yes	The data organization of the request.
device	varchar	No	The selected device of the request.
batch_size	int	Yes	The batch size of the request.
test_ratio	double	Yes	The test ratio of the request.
epoch	int	Yes	The epoch of the request.
selected_feature	array	Yes	The selected features of the request.
optimizer	varchar	Yes	The optimizer information of the request.
datetime	datetime	Yes	The request datetime
test_accuracy	double	Yes	The test accuracy result of training
Train_accuracy	double	Yes	The training accuracy result of training
Description of the return parameters			
Name	Type	Statement	
None	None	None	

6.1.4. Training Record Visualization Interface Design

In order to visualize the training record, each time you enter the interface, you need to define an interface that requests all the training records in the database. By implementing this interface, all the data can be displayed on the front end of the design. The interface statement is shown as Figure 6.4.

Table 6.4 Training Record Visualization Interface Design Table

Name	Training Record Visualization
Statement	Used for requesting all training records
Request Protocol	TCP/IP
Request Method	POST
Request	JSON

Format			
Return Format	JSON		
Description of the request parameters			
Name	Type	Constraint	Statement
None	None	None	None
Description of the return parameters			
Name	Type	Statement	
selected_model	varchar	The model information of the request.	
is_shuffle	int	The data organization of the request.	
device	varchar	The selected device of the request.	
batch_size	int	The batch size of the request.	
test_ratio	double	The test ratio of the request.	
epoch	int	The epoch of the request.	
selected_feature	array	The selected features of the request.	
optimizer	varchar	The optimizer information of the request.	
datetime	datetime	The request datetime	
test_accuracy	double	The test accuracy result of training	
Train_accuracy	double	The training accuracy result of training	

6.1.5. Aided Diagnosis Interface Design

After each completion of aided diagnosis by the user, the system will automatically invoke the interface to submit the training information to the database, enabling seamless access for subsequent queries. This streamlined process facilitates medical users in storing cases as information and integrating them for comprehensive analysis purposes. The interface statement is shown as Figure 6.5.

Table 6.5 Aided Diagnosis Interface Design Table

Name	Aided Diagnosis
Statement	Used for documenting the aided diagnosis records
Request Protocol	TCP/IP
Request Method	POST
Request Format	JSON
Return Format	JSON

Description of the request parameters			
Name	Type	Constraint	Statement
request_Id	varchar	Yes	The unique id of the request.
username	varchar	Yes	The username of the user making this request.
case_path	varchar	Yes	The file path stored in local personal computer.
selected_model	varchar	Yes	The model information of the diagnosis request.
datetime	datetime	Yes	The request datetime.
Normal Case Probability	varchar	Yes	The normal probability of prediction result.
Alzheimer's Case Probability	varchar	Yes	The Alzheimer's probability of prediction result.
Description of the return parameters			
Name	Type	Statement	
None	None	None	

6.1.6. Test Record Visualization Interface Design

Different users will have different test record permissions. We use this interface to effectively control the permissions of each user to view cases. By implementing this interface, user can conveniently query all the test cases of aided diagnosis, and make the follow-up feedback work better. The interface statement is shown as Figure 6.6.

Table 6.6 Test Record Visualization Interface Design Table

Name	Test record visualization		
Statement	Used for requesting test records		
Request Protocol	TCP/IP		
Request Method	POST		
Request Format	JSON		
Return Format	JSON		
Description of the request parameters			
Name	Type	Constraint	Statement
username	varchar	Yes	The username of the user
Description of the return parameters			
Name	Type	Statement	

username	varchar	The username of the user making this request.
case_path	varchar	The file path stored in local personal computer.
selected_model	varchar	The model information of the diagnosis request.
datetime	datetime	The request datetime.
Normal Case Probability	varchar	The normal probability of prediction result.
Alzheimer's Case Probability	varchar	The Alzheimer's probability of prediction result.

6.2. Database Design

The database stores all data in the system. This section will describe the data tables in native personal computer and Mongo database in detail. There are five tables used in the MongoDB, which are user table, user log table, training record table, aided diagnosis record table and message table.

6.2.1. Brain Network Data

As the brain network is constructed using the Matlab software engineering package, the exported original data are stored in files with the ".mat" format. Mat files are capable of storing various types of data, including numerical arrays, structures, character arrays, logical value arrays, and functions, among others. Consequently, a significant amount of brain network information data will be stored in mat files, and the detailed information is shown in table 6.7. This part of data is stored locally, not in the database.

Table 6.7 Matlab File Description

Field name	Data type	Description
graph	array	The brain network is represented by a 90x90 binary matrix, which effectively captures the binary connectivity pattern of the network.
num	int	The information denotes the count of nodes representing the whole brain.
id	int	Test case number
label	int	The information denotes the status of the test case.

6.2.2. User Related Table

The User Related Table comprises crucial data essential for user registration, login, personal information modification, and other related functions. The username serves as the primary key, ensuring uniqueness, and is employed to differentiate between various user accounts. Additionally, the username plays a vital role in establishing connections between the user table and other associated tables. The authority field denotes the user's permissions, distinguishing between three distinct user types as previously mentioned: ordinary users, advanced users, and administrator users. Specific field descriptions of user table are shown in table 6. 8.

Table 6.8 User database table

Field name	Data type	Constraint description
username	varchar	Primary key, not null
password	varchar	Not null
email	varchar	
gender	int	Not null
birthday	datetime	Not null
interest	array	
authority	varchar	Not null

In order to better record the time when the user logs in and exits the system, the user log is also set up. Login logs offer a critical security measure by capturing information about who accessed a system or application, including the date, time of the login. These logs can help identify unauthorized access attempts or comparative activities. Specific field descriptions of the user log table are shown in table 6. 9.

Table 6.9 User log database table

Field name	Data type	Constraint description
username	varchar	Not null
login_time	datetime	Not null
logout_time	datetime	Not null
login_id	varchar	Primary key, not null
authority	varchar	Not null

6.2.3. Training Record Table

The Training database is specifically structured to maintain a comprehensive record each time the administrator user trains the model. Each training record is uniquely identified by a request_id, while the username signifies the identity of the administrator user responsible for the test. The additional information entails a descriptive account of the model parameters during the training process. By diligently storing every training record in the database and implementing a sophisticated front-end visualization module, it empowers the administrator user to capture and document optimal parameter values effectively. The database table of training record is shown in table 6.10.

Table 6.10 Training record table

Field name	Data type	Constraint description
username	varchar	Not null
request_Id	varchar	Primary key, not null
selected_model	varchar	Not null
is_shuffle	int	Not null
device	varchar	Not null
batch_size	int	Not null
test_ratio	double	Not null
epoch	int	Not null
selected_feature	array	Not null
optimizer	varchar	Not null
datetime	datetime	Not null
test_accuracy	double	Not null
Train_accuracy	double	Not null

6.2.4. Aided Diagnosis Record Table

The records of aided diagnosis hold immense significance, as they provide valuable information accessible to users with varying privileges. Additionally, different permissions enable the execution of statistical analysis across distinct categories of test cases. The request_id serves as the unique identifier for each aided diagnosis record, while the username facilitates the differentiation of users responsible for completing the aided diagnosis. Furthermore, pertinent details about the selected model are meticulously logged. Given the direct manipulation of local data in aided diagnosis, the absolute path of the local data is also stored to ensure comprehensive documentation. The database table of aided diagnosis is shown in table 6.11.

Table 6.11 Aided diagnosis record table

Field name	Data type	Constraint description
request_Id	varchar	Primary key, not null
username	varchar	Not null
case_path	varchar	Not null
selected_model	varchar	Not null
datetime	datetime	Not null
Normal Case Probability	varchar	Not null
Alzheimer’s Case Probability	varchar	Not null

6.2.5. Message Table

The Message Table serves as a repository for the insightful recommendations shared by advanced users with the system, offering valuable insights to administrators and system developers for the purpose of enhancing and refining the auxiliary diagnosis system holistically. The database table of message is shown in table 6.12.

Table 6.12 Message table

Field name	Data type	Constraint description
username	varchar	Not null
message_id	varchar	Primary key, not null
datetime	datetime	Not null
message	text	Not null

6.3. Summary

This chapter presents the system design of the AAD system, offering a comprehensive overview of its design principles and methodologies. It delves into various facets such as interface design, database structure, and more, providing valuable insights and direction for the subsequent implementation phase of AAD system. Furthermore, the chapter lays the groundwork for realizing the envisioned capabilities and functionalities of the system.

Chapter 7 System Implementation

The system has been implemented using existing technology and deployed in the actual environment, taking into consideration the system design. This section begins by providing a detailed overview of the application technology utilized in each module of the AAD system. Following that, the system's environment configuration is described. Subsequently, a comprehensive explanation of the code implementation and logical structure of each module within the AAD system is presented.

7.1. Technologies in AAD System

The implementation of AAD components utilizes various technologies, as illustrated in Figure 7.1. These technologies have diverse applications in their respective functionalities, offering a broad spectrum of capabilities. The AAD core, responsible for managing database connections, is developed using the Python programming language and PyMongo. Additionally, it seamlessly integrates support for numerous identity authentication and authority authentication methods, further enhancing its versatility and adaptability. Moreover, the utilization of these technologies empowers the AAD system with robust and flexible functionalities, ensuring optimal performance and security.

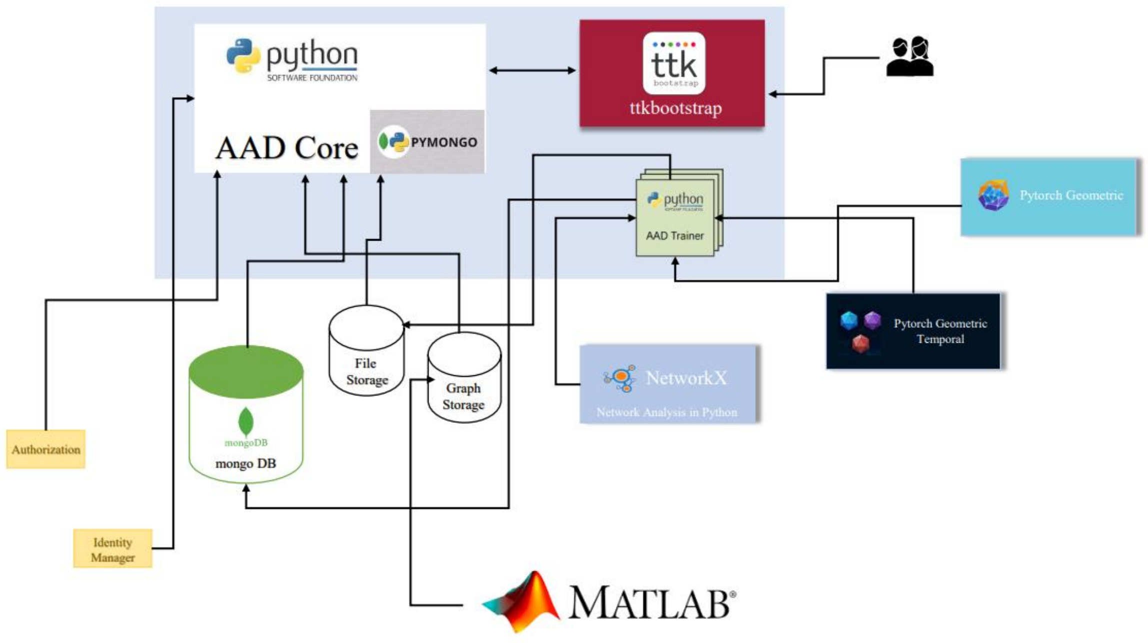


Figure 7.1 AAD system implementation diagram

For data persistence, MongoDB is utilized to store user data and system resources in the AAD system. MongoDB is chosen for its advanced capabilities in handling large volumes of user data and code data, while meeting the requirements of high access rates and low response times. It offers scalability and flexibility to accommodate the dynamic nature of the AAD system. In terms of system resource usage, where data storage requirements are small but access rate and response time are crucial, MongoDB proves to be an ideal choice. It efficiently stores and retrieves system resource data, ensuring high performance and quick response times. For storing result data like diagrams and files, local file storage is employed. This approach is suitable for the AAD system's needs, as it requires minimal storage and has low access rates for result data. Local file storage allows for efficient handling and retrieval of these data types within the system.

The AAD Trainer component utilizes several Python libraries, including the NetworkX library for network analysis, PyTorch Geometric library for graph neural networks, and the recently introduced PyTorch Geometric Temporal library for handling temporal networks. When the frontend sends a request to the AAD Core component, it parses the instructions to invoke the library functions implemented with PyTorch for model training tasks. Apart from model training, the InMemoryDataset function provided by PyTorch Geometric is employed to process local brain network image files, transforming them into a format that PyTorch can handle. Additionally, the NetworkX library is utilized for extracting features from brain networks. Once the AAD Trainer component completes the model detection process, it sends the obtained accuracy and relevant parameter information to be stored in the MongoDB database. Storing resulting accuracy and parameter information in the MongoDB database ensures easy access and retrieval for further analysis and evaluation.

The frontend of the AAD system is developed using the ttkbootstrap framework, which offers a comprehensive set of tools and components for building modern and visually appealing user interfaces. With ttkbootstrap, the frontend interface of the AAD system is enhanced with a wide range of customizable widgets, including buttons, text fields, dropdown menus, and more. These components can be easily styled and customized to align with the system's design requirements and provide a consistent and intuitive user experience. One of the key advantages of ttkbootstrap is its seamless integration with the Windows system. The framework is designed to work smoothly and efficiently within the Windows environment, ensuring compatibility and optimal performance.

7.2. Environment Configuration of System Construction

The computer parameters of the AAD system are as follows: 11th Gen Intel(R) Core(TM) i7-1165G7 @2.80GHz with a maximum frequency of 2.80 GHz, accompanied by 8.00 GB of

RAM. AAD system is deployed directly on the local machine, utilizing the aforementioned specifications.

The AAD system does not have high CPU requirements. It can be used effectively even on devices with single-core processors. The memory requirement varies depending on the size of the dataset. For smaller datasets, the memory demand is not significant. However, for larger datasets, a relatively higher amount of memory is needed to ensure smooth processing. Additionally, the use of a GPU can significantly accelerate data processing rates, providing a faster and more efficient experience.

Each module is installed in a different environment. As follows:

AAD Frontend: Window 10 + ttkbootstrap 1.10.1 + Python 3.8

AAD Core: Window 10 + Python 3.8 + PyMongo 4.3.3

AAD Trainer: Window 10 + Python3.8 + Pytorch Geometric + Pytorch Geometric Temporal

7.3. Implementation Detail

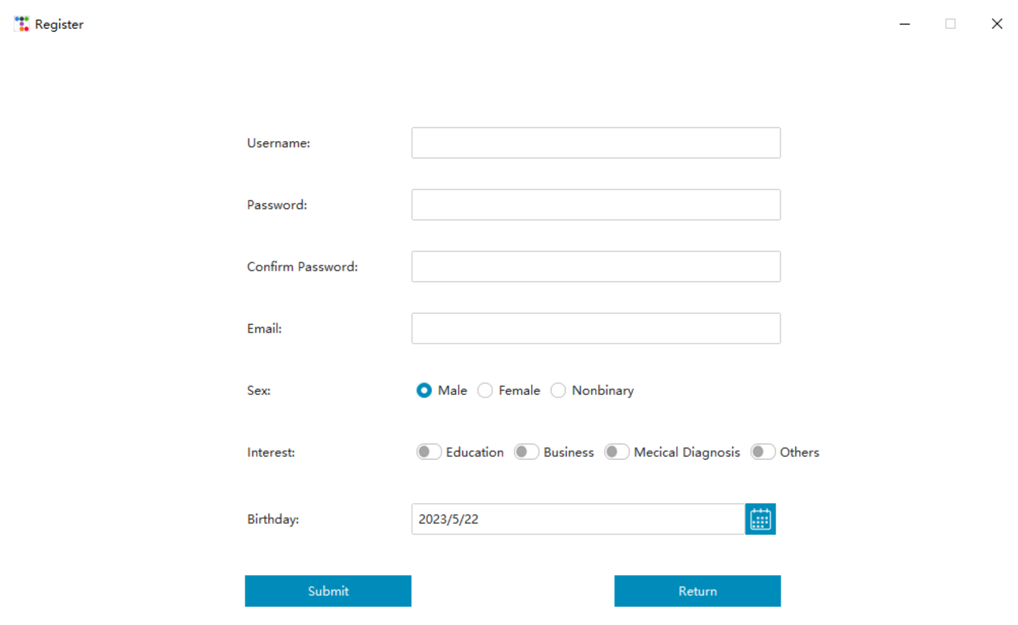
This section describes the implementation details of the key functions of AAD system. The system mainly consists of three primary functional modules: the user management module, the model training module, the aided diagnosis module, and an independent module for leaving messages. The following sections will provide individual introductions to these modules.

7.3.1. User Management Module

(1) Registration

The user registration functionality is a prerequisite operation for the login system, specifically targeting visitors who have not yet registered. The overall interface of the system is depicted in Figure 7.2. The page incorporates various widgets such as Entry, Radiobutton, Checkbutton, and DateEntry. These widgets serve as mediums for information collection, gathering user data.

The registration process involves multiple steps to ensure the legitimacy of user information. The overall workflow is illustrated in Figure 7.3. As a visitor, the first step is to log in and access the registration interface. Then, the visitor is required to fill in all the personal information fields on the form. On the frontend, the system checks if the passwords entered by the visitor match and verifies if the username and password fields are not empty. Additionally, it verifies if the password meets the standards, such as containing both uppercase and lowercase letters and being at least 8 characters long. These measures are in place to ensure the security of user accounts.


Register

Username:

Password:

Confirm Password:

Email:

Sex: [✓ Male] Female Nonbinary

Interest: Education Business Mecical Diagnosis Others

Birthday: 2023/5/22

Figure 7.2 Registration Interface

Once all the information meets the requirements, the registration form is submitted to the backend. The backend verifies if the username provided is already taken and returns an error message to the frontend if a duplicate is found. The frontend then displays a prompt using a message window to notify the user of the error. If all conditions are met, the system displays a success message using a message window to inform the user of successful registration. The system also retains the username filled in during registration and automatically populates it in the username Entry field on the Login page.

(2) Login

The Login interface primarily utilizes buttons, Entry fields, and Labels, among other widgets, to interact with users and collect their information as Figure 7.4. During the design process, careful consideration was given to the selection and arrangement of these components to ensure seamless user interaction and data input.

The system presets an administrator account with the username "admin" which has access rights to the entire system. Additionally, the administrator has the capability to modify user permissions, elevating regular users to administrator status. The main interface consists of Entry fields for the username and password. Once the user enters both pieces of information, they are sent to the backend. The backend then queries the corresponding record in the database and performs interface redirection based on the retrieved user's permission information. Depending on the user's identity, they will be directed to different interfaces. The detailed flow chart of login operation is presented through Figure 7.5.

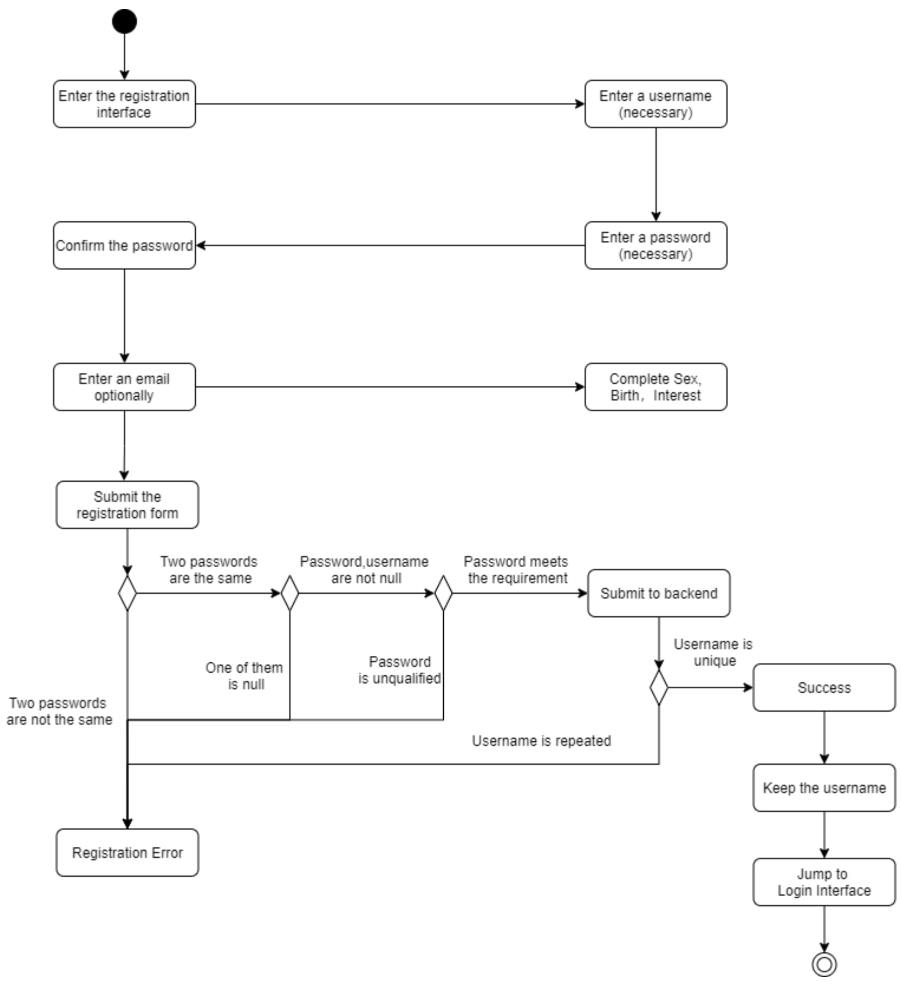

Enter the registration interface
Enter a username (necessary)
Enter a password (necessary)
Confirm the password
Enter an email optionally
Complete Sex, Birth, Interest
Submit the registration form
Two passwords are the same
Password,username are not null
Password meets the requirement
Submit to backend
Username is unique
Success
One of them is null
Password is unqualified
Two passwords are not the same
Username is repeated
Keep the username
Registration Error
Jump to Login Interface

Figure 7.3 Registration Flow Chart

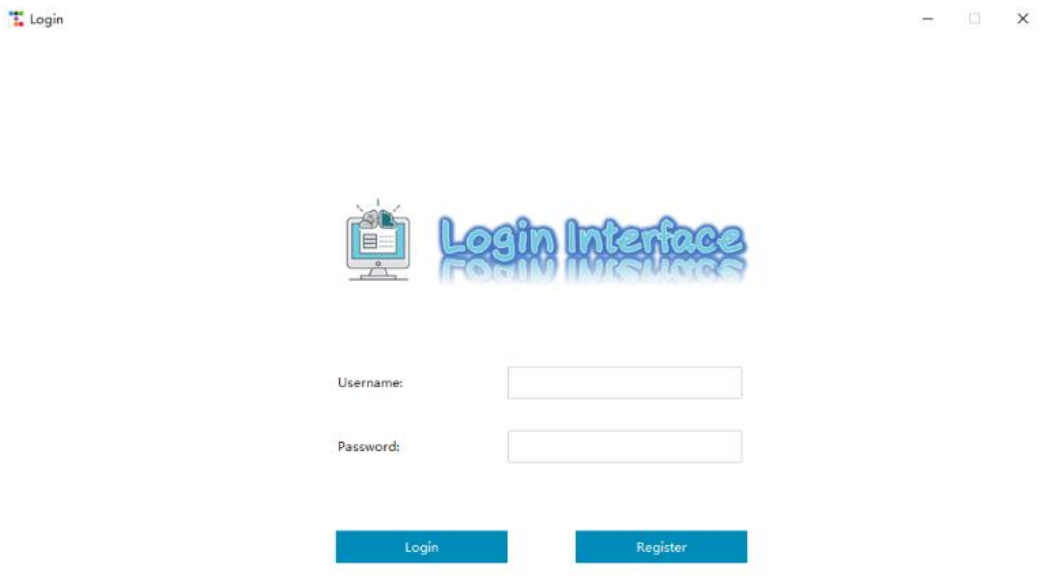


Figure 7.4 Login Interface

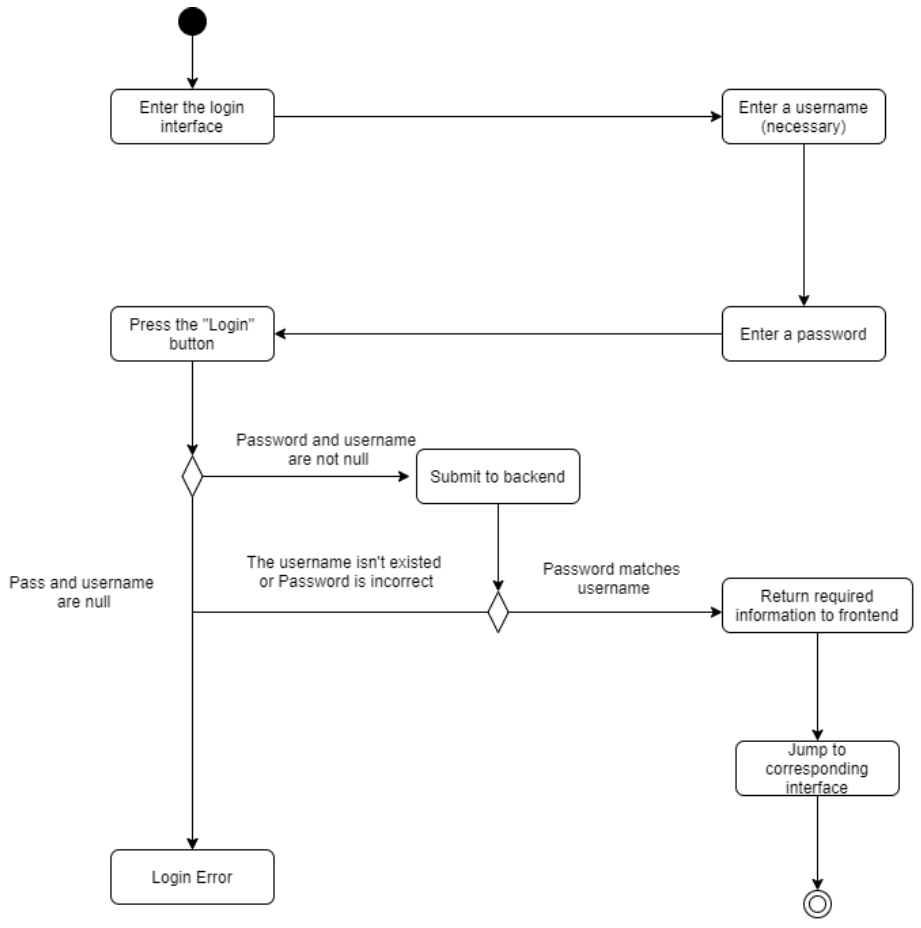

Enter the login interface
Enter a username (necessary)
Press the "Login" button
Enter a password
Password and username are not null
Submit to backend
Pass and username are null
The username isn't existed or Password is incorrect
Password matches username
Return required information to frontend
Jump to corresponding interface
Login Error

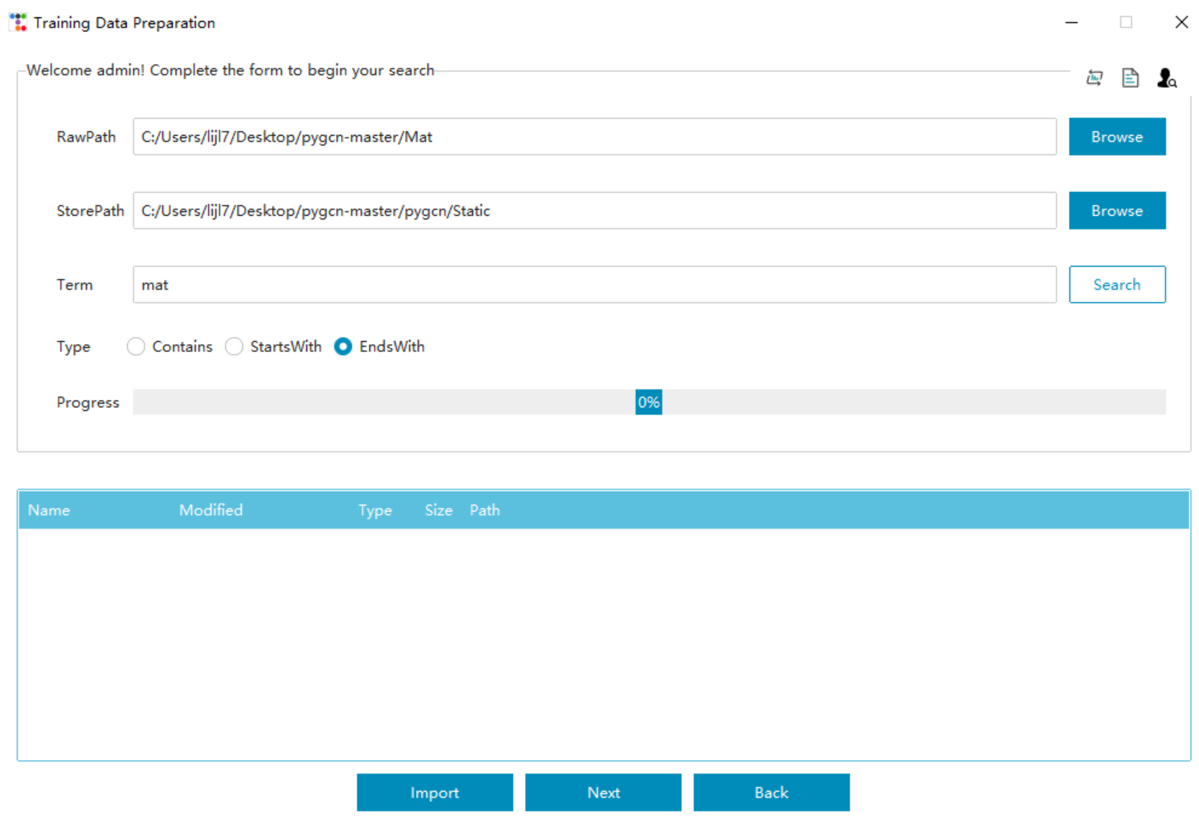
Figure 7.5 Login Flow Chart

7.3.2. Model Training Module

(1) Training Dataset Import

This interface utilizes several different widgets compared to the previously mentioned interfaces. The main components include Entry, RadioButton, Progressbar, and Treeview. The Progressbar displays the progress of data import, while the Treeview displays all the files under the selected file path, allowing for accurate filtering of specific file formats when combined with the RadioButton. At the bottom of the interface, there are several buttons that allow users to navigate to other interfaces, as shown in Figure 7.6.

This section involves interaction with both the local system data and the MongoDB database. The frontend system can query and manage all files on the local machine, enabling efficient data searching. To speed up the search process for raw data, a data filtering mechanism has been implemented to accurately identify files with specific file extensions. Once the desired files are found and confirmed to be in a format compatible with the Pytorch Geometric library, the data import process can commence. Upon successful completion of data import, the progress bar reaches 100% to indicate to the user that they can proceed to the next step. The system also includes a validation check, which displays a warning message using a message window if the "Next" button is clicked before the data import process is complete. This ensures that users can utilize the system effectively. The entire workflow of training data import procedure is depicted in Figure 7.7.



Training Data Preparation

Welcome admin! Complete the form to begin your search

RawPath: C:/Users/lijl7/Desktop/pygcn-master/Mat

StorePath: C:/Users/lijl7/Desktop/pygcn-master/pygcn/Static

Term: mat

Type: Contains StartsWith [✓ EndsWith]

Progress: 0%

Name	Modified	Type	Size	Path

Figure 7.6 Training Dataset Import Interface

(2) Model Parameter Adjustment

This section represents the most complex and comprehensive interface in terms of frontend design. It incorporates a wide range of widgets, including ProgressBar, CheckButton, Scale, Combobox, and Button. To ensure a visually appealing and consistent theme throughout the interface, various small icons and logo images have been added. Some of these images also have interactive effects. For example, clicking on certain icons, such as the "back" icon, allows the Administrator user to reset the corresponding Scale widget to its initial value. Additionally, when a specific model's CheckButton is selected, it locks the other model options' CheckButtons, and the name of the selected model is displayed in the model label on the panel. The ProgressBar serves as a reminder for the Administrator user, indicating the progress of parameter selection and model training processes. These thoughtful design choices contribute to user-friendliness and enhance the overall user experience, as shown in Figure 7.8.

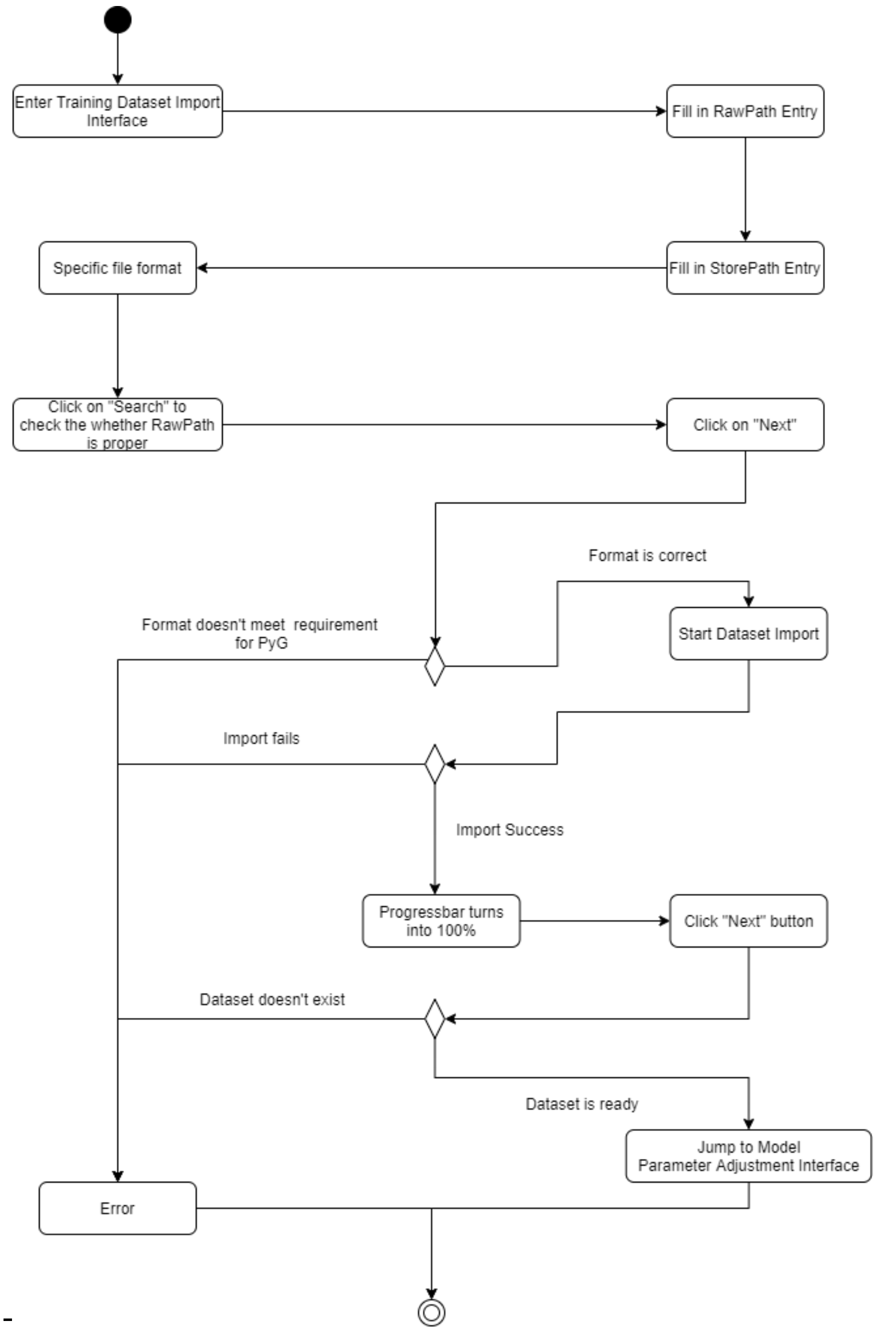
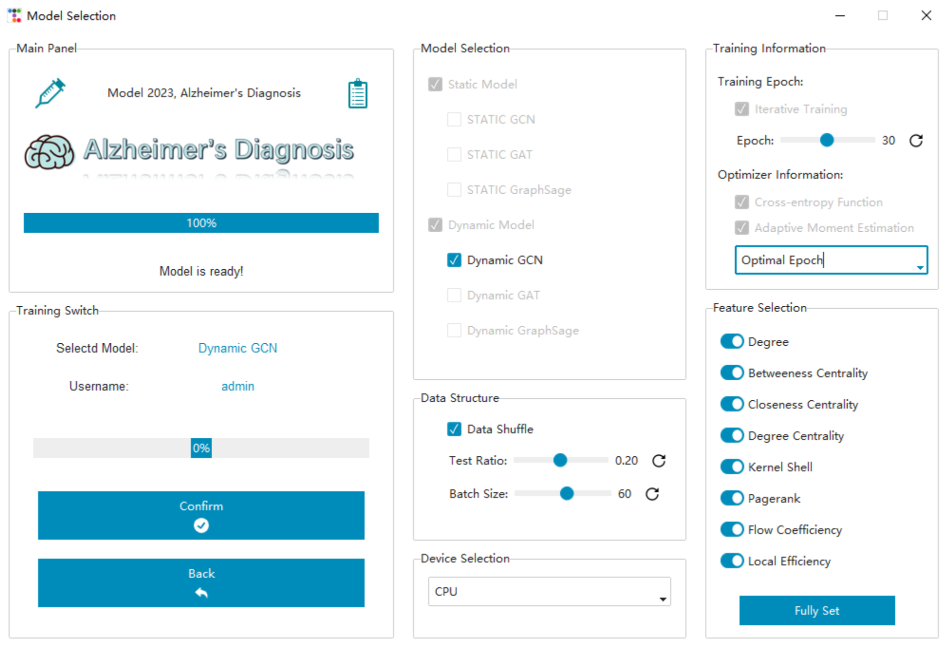

Enter Training Dataset Import Interface
Fill in RawPath Entry
Fill in StorePath Entry
Specific file format
Click on "Search" to check the whether RawPath is proper
Click on "Next"
Format is correct
Format doesn't meet requirement for PyG
Start Dataset Import
Import fails
Import Success
Progressbar turns into 100%
Click "Next" button
Dataset doesn't exist
Dataset is ready
Jump to Model Parameter Adjustment Interface
Error

Figure 7.7 Training Dataset Import Flow Chart

Once the training data import process is completed, the system automatically navigates to the Model Parameter Adjustment Interface. In the initial interface, most parameters are pre-set with default values. The Administrator user needs to select the desired model and choose the brain network node features to incorporate. After adding both selections, the existing parameters can be further adjusted based on specific requirements. Once all parameter settings are finalized, the progress bar below the Main Panel area reaches 100%, signaling to the user that all parameter information has been completed. At this point, the user can click the

"Confirm" button to initiate the parameter training process.



100%
Model is ready!

Training Switch
Selectd Model: Dynamic GCN
Username: admin
0%

Model Selection
☑ Static Model
☐ STATIC GCN
☐ STATIC GAT
☐ STATIC GraphSage
☑ Dynamic Model
☑ Dynamic GCN
☐ Dynamic GAT
☐ Dynamic GraphSage

Data Structure
☑ Data Shuffle
Test Ratio: 0.20
Batch Size: 60

Device Selection
CPU

Training Information
Training Epoch:
☑ Iterative Training
Epoch: 30
Optimizer Information:
☑ Cross-entropy Function
☑ Adaptive Moment Estimation
Optimal Epoch

Feature Selection
Degree
Betweeness Centrality
Closeness Centrality
Degree Centrality
Kernel Shell
Pagerank
Flow Coefficiency
Local Efficiency

Figure 7.8 Model Parameter Adjustment Interface

After clicking "Confirm," the system backend invokes the models implemented using the Pytorch Geometric and Pytorch Geometric Temporal libraries. The frontend system locks the "Confirm" button and begins the model training process. Once the model training is completed, the progress bar in the "Training Switch" area reaches 100%, and a message box pops up indicating that the model results are ready. Upon the Administrator user's response to the message box, the system automatically navigates to the Training Dataset Visualization interface, where the most recent training result is listed above. The entire workflow of model parameter selection procedure is depicted in Figure 7.9.

(3) Training Dataset Visualization

This interface primarily utilizes two encapsulated controls: TableView and Button. The search box above enables both fuzzy and precise queries, displaying only the relevant information of interest in the TableView, as shown in Figure 7.10. Right-clicking on a specific data field provides several additional functionalities, including filtering, sorting, alignment, moving, and deletion. Filtering hides all records that differ from the selected field. Sorting offers two options: ascending and descending order, allowing the user to retrieve the training record with the highest accuracy. Alignment and moving functions facilitate page adjustments based on user requirements. The deletion operation does not affect the database but rather hides the respective record in the view. At the bottom of this interface, there are two buttons. One of them is the "Export" button, which allows the Administrator user to export all information within the page frame in CSV format. This facilitates easy access to database information, enabling integration and analysis.

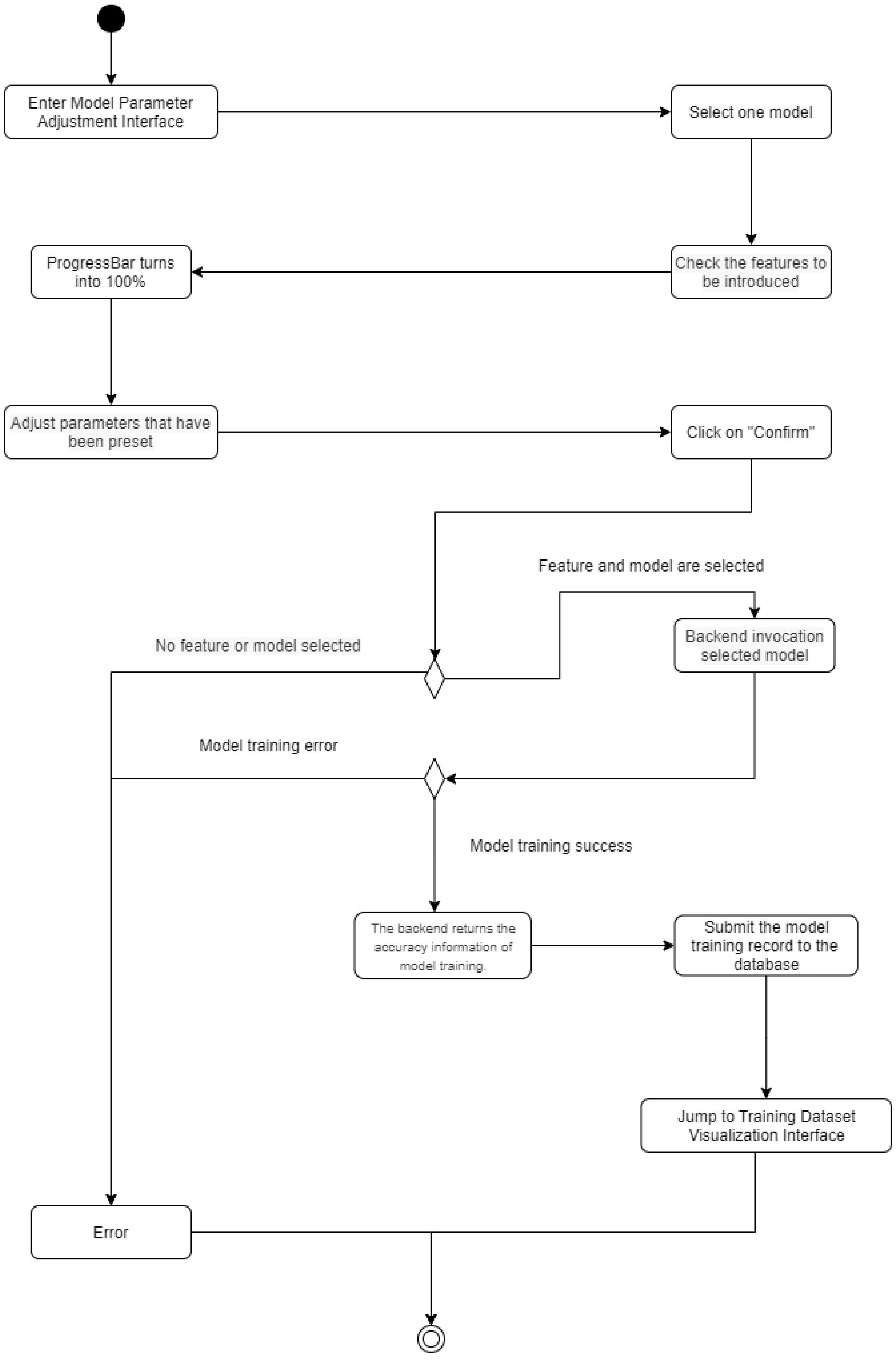
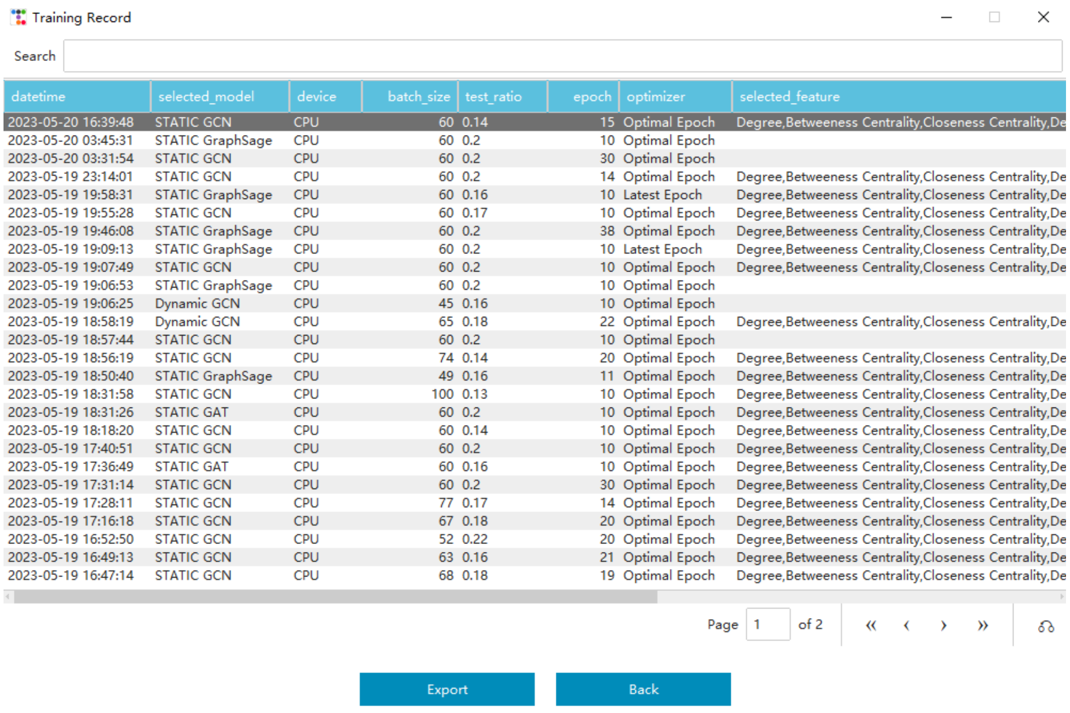

Enter Model Parameter Adjustment Interface
Select one model
Check the features to be introduced
ProgressBar turns into 100%
Adjust parameters that have been preset
Click on "Confirm"
Feature and model are selected
Backend invocation selected model
No feature or model selected
Model training error
Model training success
The backend returns the accuracy information of model training.
Submit the model training record to the database
Jump to Training Dataset Visualization Interface
Error

Figure 7.9 Model Parameter Adjustment Flow Chart



datetime	selected_model	device	batch_size	test_ratio	epoch	optimizer	selected_feature
2023-05-20 16:39:48	STATIC GCN	CPU	60	0.14	15	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-20 03:45:31	STATIC GraphSage	CPU	60	0.2	10	Optimal Epoch	
2023-05-20 03:31:54	STATIC GCN	CPU	60	0.2	30	Optimal Epoch	
2023-05-19 23:14:01	STATIC GCN	CPU	60	0.2	14	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 19:58:31	STATIC GraphSage	CPU	60	0.16	10	Latest Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 19:55:28	STATIC GCN	CPU	60	0.17	10	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 19:46:08	STATIC GraphSage	CPU	60	0.2	38	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 19:09:13	STATIC GraphSage	CPU	60	0.2	10	Latest Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 19:07:49	STATIC GCN	CPU	60	0.2	10	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 19:06:53	STATIC GraphSage	CPU	60	0.2	10	Optimal Epoch	
2023-05-19 19:06:25	Dynamic GCN	CPU	45	0.16	10	Optimal Epoch	
2023-05-19 18:58:19	Dynamic GCN	CPU	65	0.18	22	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 18:57:44	STATIC GCN	CPU	60	0.2	10	Optimal Epoch	
2023-05-19 18:56:19	STATIC GCN	CPU	74	0.14	20	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 18:50:40	STATIC GraphSage	CPU	49	0.16	11	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 18:31:58	STATIC GCN	CPU	100	0.13	10	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 18:31:26	STATIC GAT	CPU	60	0.2	10	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 18:18:20	STATIC GCN	CPU	60	0.14	10	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 17:40:51	STATIC GCN	CPU	60	0.2	10	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 17:36:49	STATIC GAT	CPU	60	0.16	10	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 17:31:14	STATIC GCN	CPU	60	0.2	30	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 17:28:11	STATIC GCN	CPU	77	0.17	14	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 17:16:18	STATIC GCN	CPU	67	0.18	20	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 16:52:50	STATIC GCN	CPU	52	0.22	20	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 16:49:13	STATIC GCN	CPU	63	0.16	21	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De
2023-05-19 16:47:14	STATIC GCN	CPU	68	0.18	19	Optimal Epoch	Degree,Betweeness Centrality,Closeness Centrality,De

Figure 7.10 Training Dataset Visualization Interface

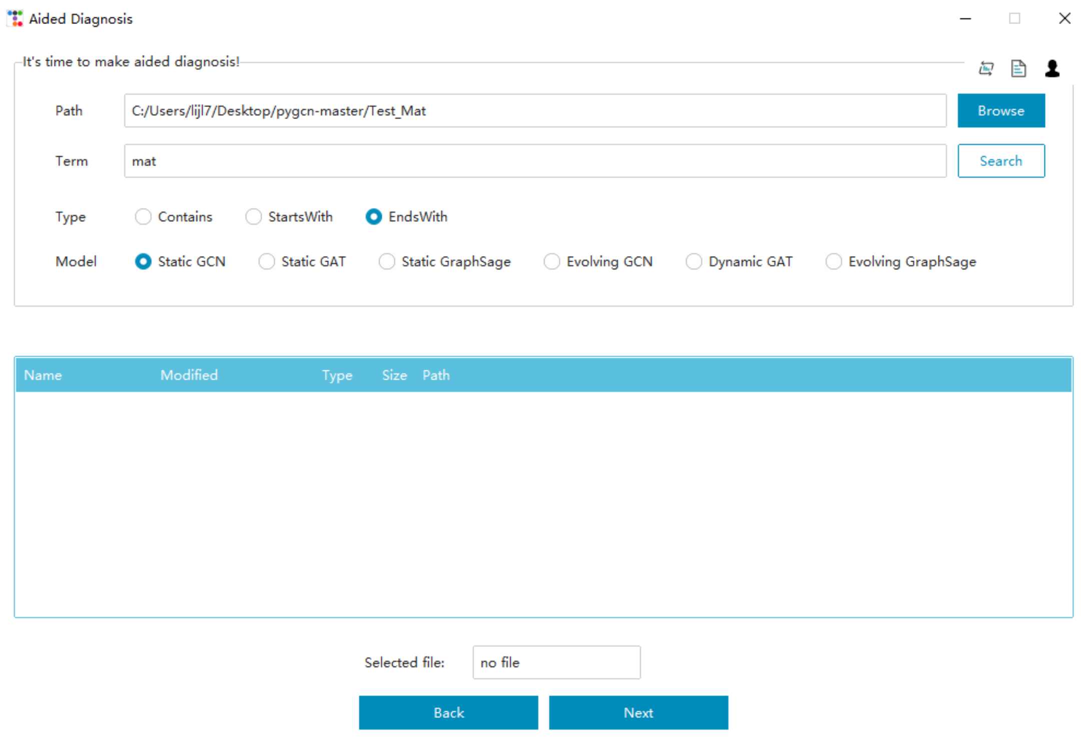
7.3.3. Aided Diagnosis Module

(1) Aided Diagnosis

The functionality of the diagnostic assistance requires the implementation of two interfaces. The first interface, as shown in Figure 7.11, primarily utilizes controls such as Entry, RadioButton, Progressbar, and Treeview. It shares similarities with the Administrator user's Training Dataset Import interface, with similar functionalities. The common parts are not reiterated. However, in this interface, upon clicking the "Search" button, all files in the selected folder that meet the specified criteria will be displayed. We need to choose one file as a test case for the diagnostic operation. After selecting a test case, the filename will appear in the Entry at the bottom of the interface, serving as the data for the subsequent steps.

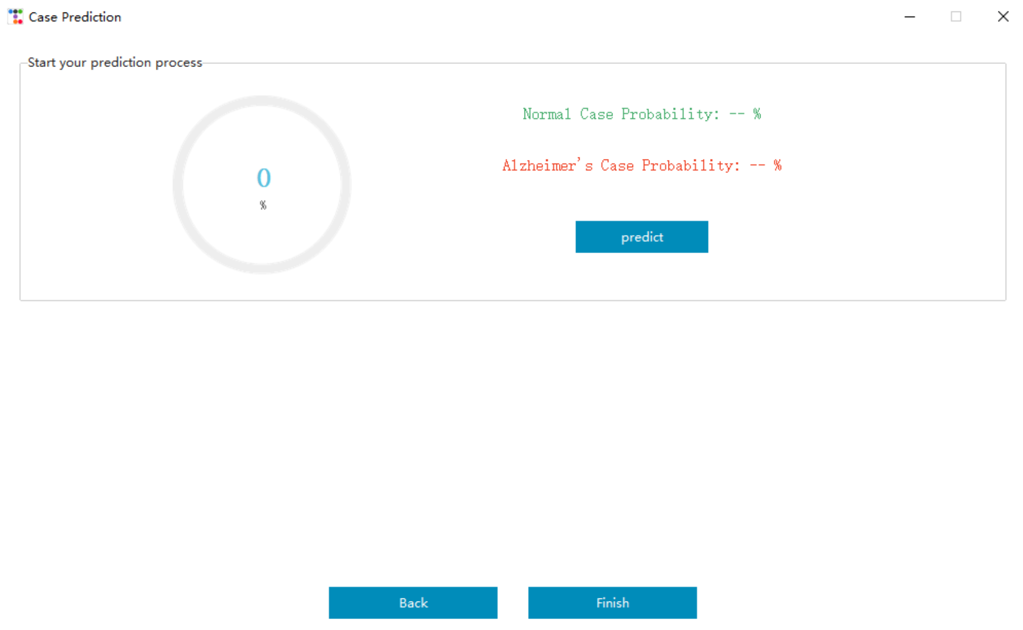
The other interface is used to present the prediction results. It employs a Meter and Label control to display the probability values of the predicted normal and diseased cases, as shown in Figure 7.12.

The process of importing test cases is similar to the previous chapter's procedure for importing training data and will not be reiterated here. Once the data import is completed, the interface will transition to the prediction result display. The backend passes the imported test case data to the selected model. When the model training is completed, the results are returned to the frontend, where the probabilities of the two categories in the binary classification can be displayed. After obtaining the prediction results, the user's prediction log will be stored in the database. Users can access the Diagnosis Record Visualization Interface to retrieve relevant test case prediction results. Advanced users and administrators can view the historical test results of all users, while ordinary users can only view their own data. The overall process is illustrated in Figure 7.13.



Path: C:/Users/lijl7/Desktop/pygcn-master/Test_Mat
Term: mat
Type: ○ Contains ○ StartsWith [✓ EndsWith]
Model: [✓ Static GCN] ○ Static GAT ○ Static GraphSage ○ Evolving GCN ○ Dynamic GAT ○ Evolving GraphSage
Selected file: no file

Figure 7.11 Test Case Import Interface



Normal Case Probability: -- %
Alzheimer's Case Probability: -- %

Figure 7.12 Prediction Visualization Interface

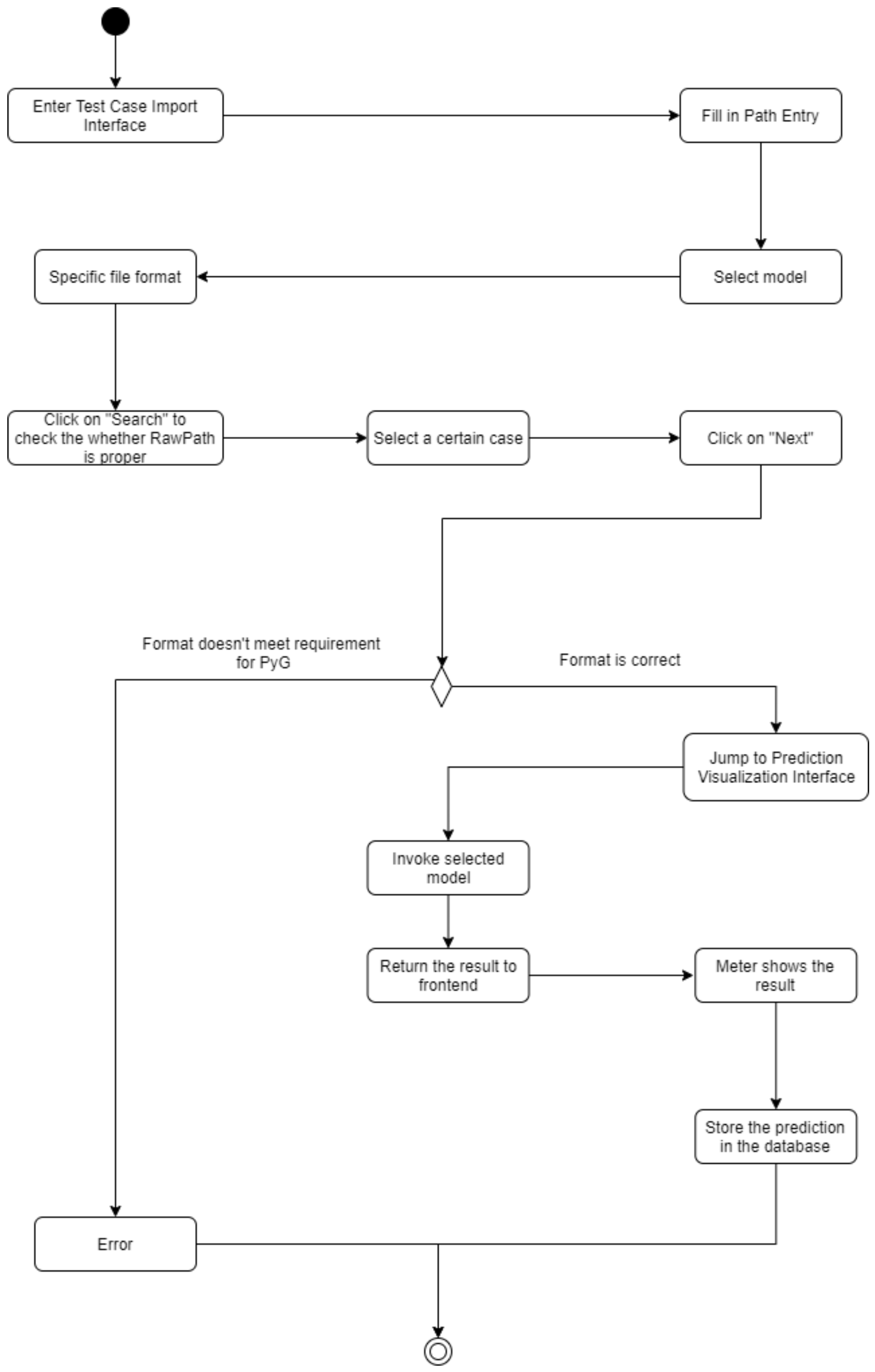
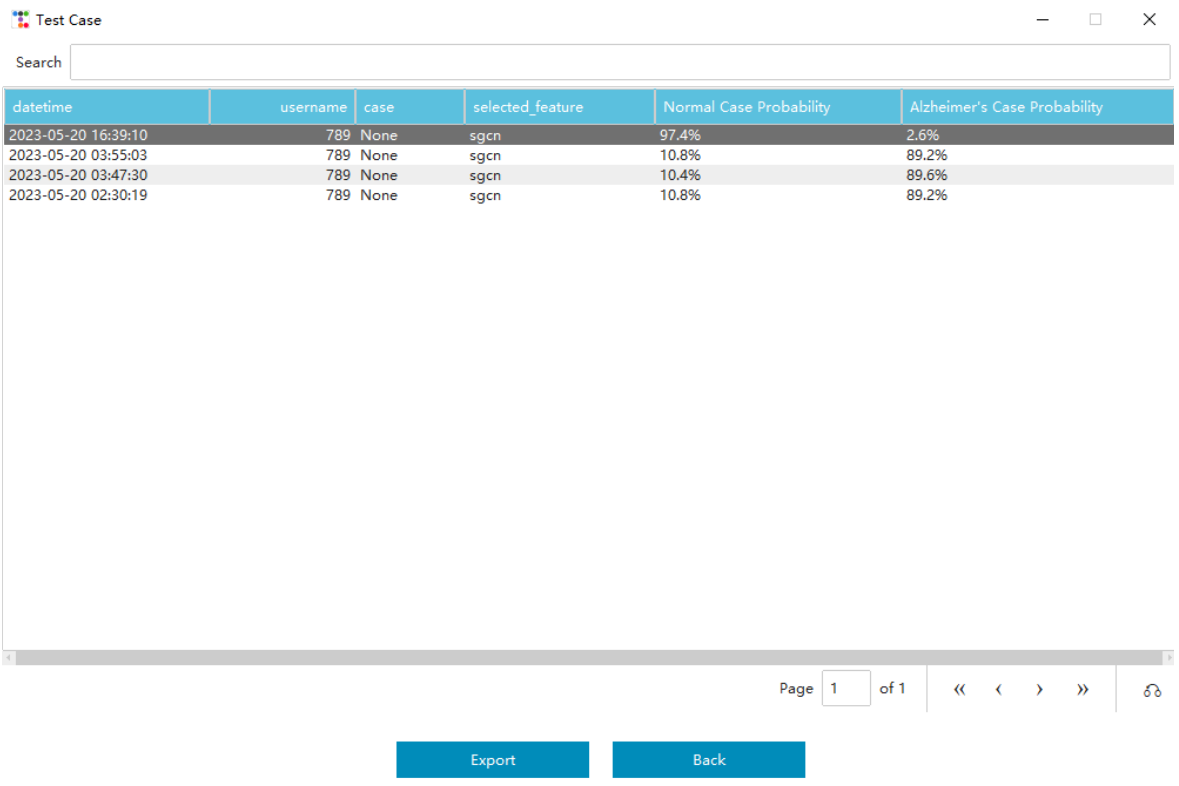

Enter Test Case Import Interface
Fill in Path Entry
Select model
Specific file format
Click on "Search" to check the whether RawPath is proper
Select a certain case
Click on "Next"
Format doesn't meet requirement for PyG
Format is correct
Jump to Prediction Visualization Interface
Invoke selected model
Return the result to frontend
Meter shows the result
Store the prediction in the database
Error

Figure 7.13 Aided Diagnosis Flow Chart

(2) Diagnosis Record Visualization

This interface relies heavily on two encapsulated controls: TableView and Button. Positioned above, the search box facilitates both fuzzy and precise queries, exhibiting exclusively the pertinent information of interest within the TableView, as shown in Figure 7.14. Right-clicking on a specific data field unlocks a host of supplementary functionalities, encompassing filtering, sorting, alignment, moving, and deletion. Filtering discreetly conceals records deviating from the chosen field. Sorting furnishes the user with two alternatives:

ascending and descending order, enabling them to retrieve the patient with the highest probability of illness, thereby raising sufficient awareness and facilitating appropriate treatment. Aligning and moving functions gracefully accommodate page adjustments tailored to user preferences. The deletion process safeguards the database's integrity while discreetly withdrawing the corresponding record from view. Positioned at the interface's bottom, a duo of buttons awaits. Among them, the "Export" button empowers the Administrator user to extract all information encapsulated within the page frame, promptly encoding it in CSV format. This seamless exportation process expedites convenient access to database insights, fostering seamless integration and comprehensive analysis.



datetime	username	case	selected_feature	Normal Case Probability	Alzheimer's Case Probability
2023-05-20 16:39:10	789	None	sgcn	97.4%	2.6%
2023-05-20 03:55:03	789	None	sgcn	10.8%	89.2%
2023-05-20 03:47:30	789	None	sgcn	10.4%	89.6%
2023-05-20 02:30:19	789	None	sgcn	10.8%	89.2%

Figure 7.14 Diagnosis Record Visualization Interface

7.3.4. Message Module

Advanced users can provide feedback and suggestions to the system for the purpose of system improvement, facilitating collaboration between developers and administrator users. Upon accessing this interface, the Administrator user is required to enter information in the Text input box and click the "Submit" button to complete the operation. The interface layout is depicted in Figure 7.15. The database will store all the messages submitted by advanced users.

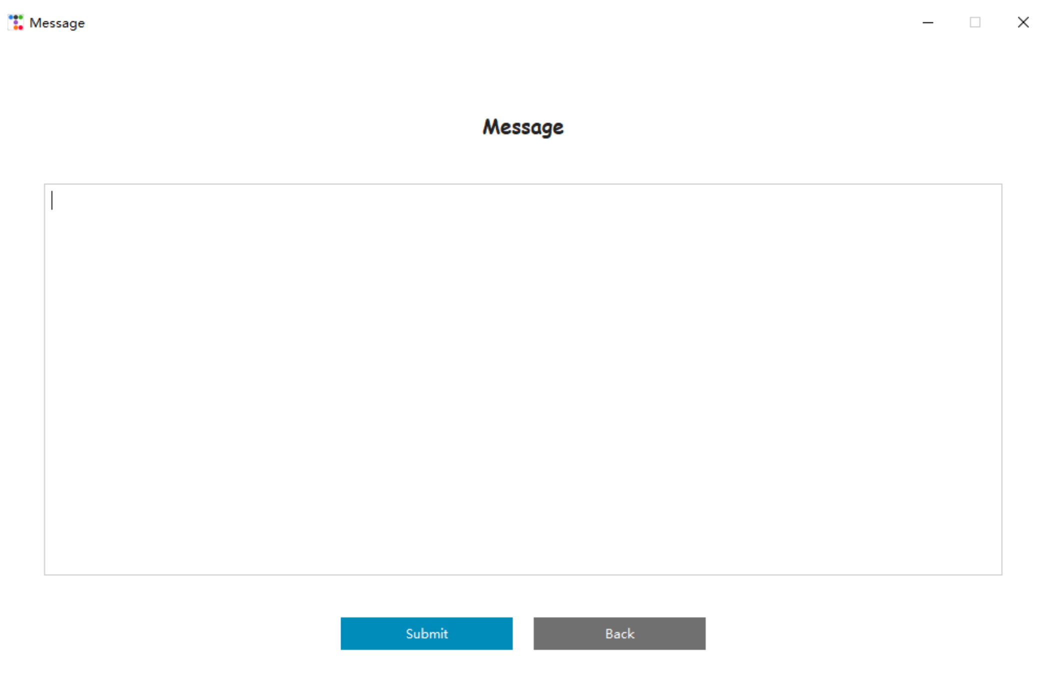


Figure 7.15 Leaving Message Interface

7.4. Summary

This chapter provides an overview of the implementation of the AAD system, covering the technologies employed in each module, system environment configuration, frontend interface development, and key functionalities of the AAD system. The logical flow of the main functionality implementation process is presented through flowcharts. Furthermore, it explains the styling of the frontend interface and the interconnected reactions among different controls. This chapter offers a comprehensive insight into the design philosophy and implementation approach of the entire system.

Chapter 8 System Test

System testing is typically employed to validate the robustness of software functionalities, ensuring smooth operations, seamless data accessibility, and stable performance. By designing test cases and operational procedures, it becomes possible to verify the program code parameters and data. In this chapter, we employ functional testing, integrity testing, and performance testing methodologies to thoroughly assess the reliability and efficiency of the AAD (Aided Alzheimer's Diagnosis) system. Furthermore, these tests facilitate the identification of any potential vulnerabilities, optimization opportunities, and overall system enhancements.

8.1. Functional Test

8.1.1. Purpose of Test

The primary objective of functional testing for the AAD system is to ensure the system's functionality aligns with user requirements. Testers execute the test by interacting with component modules or inputting data. Testers may include developers or potential users, who assess the system from their unique perspectives as different user types may have varying access to functions.

8.1.2. Test Method

Taking into consideration the specific attributes of the AAD system, testers create multiple test cases and subsequently execute corresponding operations based on the prescribed test steps. These operations may include interacting with buttons, inputting data, and examining images, among others. The attained outcomes are then compared against the anticipated results to analyze the actual impact. The manual testing approach is employed to systematically evaluate each designed test case.

8.1.3. Test Case

The test scenarios encompass the essential functionalities of the entire system. Due to spatial constraints, this document provides illustrations of the testing procedure through two exemplars: the rating changing test and the running status result viewing test. These specific instances are elaborated upon in Table 8.1.

Table 8.1 Functional test cases

ID	test case name	Specific operation steps	Expected target	Result
1	Register test 1	1. Login to system as visitor. 2. Fill in the user name completely, enter the password "Gem010131 @ neu", and confirm the password with "gem010131 @ neu". 3. Improve other optional information on the form. 4. Click on “submit” button.	The prompt box shows "Password doesn't match!".	Consistent with expected results.
2	Register test 2	1. Login to system as visitor. 2. Fill in the username completely, enter the password "gem010131 @ neu", and confirm the password with "gem010131 @ neu". 3. Improve other optional information on the form. 4. Click on "submit" button.	The prompt box shows "Password must contain at least one uppercase letter.".	Consistent with expected results.
3	Register test 3	1. Login to system as visitor. 2. Leave the username blank, enter the password "Gem010131 @ neu", and confirm the password with "Gem010131 @ neu". 3. Improve other optional information on the form. 4. Click on "submit" button.	The prompt box shows "Please complete the information.".	Consistent with expected results.
4	Login test 1	1. Enter the login interface of the system. 2. Enter the username "admin" and its corresponding password. 3. Click on "Login" button.	The system automatically jumps to the Training Dataset Import Interface.	Consistent with expected results.
5	Login test 2	1. Enter the login interface of the system. 2. Enter the username "doctor1" and its corresponding password. 3. Click on “Login” button.	The system automatically jumps to the Test Case Import Interface.	Consistent with expected results.

Table 8.1 (continued) Functional test cases

ID	test case name	Specific operation steps	Expected target	Result
6	Model Training test 1	1. Enter Training Dataset Import Interface. 2. There are files that do not end in ". mat" format in the selected native file path. 3. Click the “Import” button to import the dataset.	The prompt box shows "The selected path has a file format that cannot be processed.".	Consistent with expected results.
7	Model Training test 2	1. Enter Training Dataset Import Interface. 2. Click the "Next" button before importing the data.	The prompt box shows "You forget to import the dataset.".	Consistent with expected results.
8	Model Training test 3	1. Enter Model Parameters Adjustment Interface. 2. Without selecting the model and feature information, click the "Confirm" button directly.	The prompt box shows "You haven’t chosen a model.".	Consistent with expected results.
9	Model Training test 4	1. Enter Model Parameters Adjustment Interface. 2. Select the model and feature information and click the "Confirm" button.	The system starts to train the model, and after the model training is completed, the training information is stored in the database.	Consistent with expected results.
10	Aided Diagnosis test 1	1. Log on to the system Test Case Import Interface as ordinal user or advanced user. 2. The path has not been designated, and the corresponding test case has not been chosen. 3. Click on “Next” button.	The prompt box shows "You forget to choose a test case.".	Consistent with expected results.

Table 8.1 (continued) Functional test cases

ID	test case name	Specific operation steps	Expected target	Result
11	Aided Diagnosis test 2	1. Log on to the system Test Case Import Interface as ordinal user or advanced user. 2. Select test cases and models. 3. Click on “Next” button.	The system navigates to the Case Prediction Interface.	Consistent with expected results.
12	Aided Diagnosis test 3	1. The system enters the Case Prediction Interface. 2. Click on “Predict” button.	The Meter control displays the predicted probability results and stores the prediction outcome into the database.	Consistent with expected results.
13	Leave Message test 1	1. After logging in as an advanced user, access the Message Interface. 2. Leave the text area blank.	The prompt box shows "Please enter your message before submitting.".	Consistent with expected results.

8.2. Data Integrity Test

8.2.1. Purpose of Test

The purpose of the data integrity test is to validate the completeness of system data storage, ensure consistency between frontend and backend displays, and verify the persistence of data consistency after modifications.

8.2.2. Test Method

Conduct tests on system components that involve data addition, modification, deletion, and other operations to validate data transformations and document changes in both data quantity and content.

8.2.3. Test Case

Data integrity testing primarily revolves around scrutinizing data. Due to limited space, this section elucidates the testing process by emphasizing scenarios associated with database operations. These exemplary instances are comprehensively outlined in Table 8.2.

Table 8.2 Data integrity test cases

ID	test case name	Specific operation steps	Expected target	Result
1	Personal Information Update test 1	1. Login to system as an ordinary user. 2. Access the personal information modification interface. 3. Modify personal information and leave the fields that do not require changes with their default settings. 4. Click on “Submit” button.	New personal information is submitted to the database successfully.	Consiste nt with expected results.
2	Grant Authorization test 1	1. Login to system as an administrator user. 2. Access the user record statistics interface. 3. Select the information column of a specific user and enter the information modification interface. 4. In addition to being able to modify any information for the user, it is also possible to enhance their permission control. Upgrade an ordinary user to an administrator user. 5. Navigate back to the login interface. 6. Use this identity to proceed with the login process again.	After completing the login, the user can directly navigate to the Training Dataset Import Interface.	Consiste nt with expected results.

8.3. Summary

This chapter provides an overview of the system testing conducted on the AAD system, encompassing functional testing and data integrity testing. Functional testing ensures that the system's functions align with user requirements, while data integrity testing verifies the completeness of data storage, consistency between frontend and backend displays, and the ability to maintain consistency after modifications. Through comprehensive testing processes, the system effectively responds to various erroneous operations, thereby preventing data loss and system failures resulting from user errors.

Chapter 9 Summary and Prospect

9.1. Summary

This study focuses on fMRI images and constructs them into dynamic brain functional networks. The method used is an evolving graph neural network, which fuses information from brain networks at different time points to perform aided diagnosis. The research conclusions are as follows:

(1) By studying the network of brain nodes, we design eight different node features to reconstruct the topological information of the brain network. To achieve better diagnostic performance, we evaluate the accuracy of aided diagnosis and analyze the impact of various features on diagnostic accuracy. The research results show that node feature information needs to be enriched and can achieve better diagnostic performance under sufficient information influence.

(2) By employing evolving neural network based on GraphSage and evolving GCN, we effectively fuse node information from different time points in the brain network. Through extensive experiments, it can be verified that constructing dynamic brain networks yields better aided diagnostic results compared to static brain networks with the same test data.

The main innovative points:

(1) Adoption of Evolving Graph Neural Network: The article proposes the use of evolving graph neural networks for assisting in the diagnosis of Alzheimer's disease. Specifically, the GraphSage sampling algorithm is designed to fuse information from different temporal dimensions of the brain network nodes. This approach enables the model to capture temporal dynamics and integrate them into the diagnostic process.

(2) Development of the AAD System: The article presents the AAD (Alzheimer's Aided Diagnosis) system, which provides healthcare professionals with advanced aided diagnostic capabilities. The system includes modules for model training and aided diagnosis, catering to different user roles. Additionally, the system incorporates database visualization functionality, facilitating the management of medical data, including diagnostic data and model training parameters.

9.2. Prospect

The research work of this paper has room for improvement in many aspects.

9.2.1. Extended Dataset

Because of the time-consuming processing of fMRI data, the data sets used in this paper are limited. This limitation may have a certain impact on the research results. In order to overcome the limitation of data set and further verify the reliability and generalization ability of our method, future research can consider expanding the scale of data set to include more samples and multiple data sources. In addition, other relevant neuroimaging data and clinical information can be combined to obtain more comprehensive analysis results.

9.2.2. Improvement of Aided Diagnosis

While this study focuses on a binary classification problem, we acknowledge that Alzheimer's Disease encompasses multiple states, including Mild Cognitive Impairment (MCI) and Mild Alzheimer's Disease, among others. These states hold significant importance in the research and aided diagnosis of Alzheimer's disease. Therefore, future experiments can be extended to address the challenges of multi-classification problems. Furthermore, it is worth noting that this paper primarily employs brain network construction at the brain region level and does not explore voxel-level brain network construction. In future investigations, we can explore the voxel-level brain network construction approach and apply it in the aided Alzheimer’s diagnosis.

9.2.3. System Improvement

At present, the system only combines two neural network models of timing diagram, which can expand more technical means in the future. The system can be expanded in the process of beautifying the interface and enrich the users' senses.

References

[1] 李彩, 范炤. 基于机器学习的阿尔兹海默症分类预测[J]. 中国医学物理学杂志, 2020, 37(3):379-384.

[2] Jo T, Nho K, Saykin A. Deep Learning in Alzheimer’s disease: Diagnostic classification and prognostic prediction using neuroimaging data[J]. Frontiers in Aging Neuroscience, 2019, 11:1-14.

[3] 接标, 张道强. 面向脑网络的新型图核及其在 MCI 分类上的应用[J]. 计算机学报，2016, 39(8):1667-1680.

[4] Jie B, Zhang D, Wee C Y, *et al*. Topological graph kernel on multiple thresholded functional connectivity networks for mild cognitive impairment classification[J]. Human Brain Mapping, 2014, 35(7):2876-2897.

[5] 汪新蕾, 王之琼, 王中阳, 等. 面向阿尔茨海默病的脑网络多频融合图核[J]. 计算机学报, 2020, 43(1)：64-77.

[6] Cao B K, Zhan L, Kong X N, *et al*. Identification of discriminative subgraph patterns in fMRI brain networks in bipolar affective disorder[C]. International Conference on Brain Informatics and Health. London, 2015: 105-114.

[7] 王章辉, 赵宇海, 王国仁, 等. 多样性度量的 TOP-K 区分子图挖掘[J]. 计算机科学与探索, 2017, 11(09): 1379-1388.

[8] 朱镕, 邹兆年, 李建中. 不确定图上的 Top-k 稠密子图挖掘算法[J]. 计算机学报, 2016, 39(8):1570-1582.

[9] Yan X, Han J. gSpan: Graph-based substructure pattern mining[C].2002 IEEE International Conference on Data Mining. Maebashi: IEEE,2002: 721-724.

[10] Kong X, Yu PS, Wang X, *et al*. Discriminative feature selection for uncertain graph classification[C]. SIAM International Conference on Data Mining. Texas, 2013:82-93.

[11] Mrzic A, Meysman P, Bittremieux W, *et al*. Grasping frequent subgraph mining for bioinformatics applications[J]. BioData Mining, 2018,20(11):181-189.

[12] Suresh AT, Jereesh AS. Associated subgraph mining in biological network[C]. 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, 2017:80-85.

[13] Minh N, Tong H, Lijun A, *et al*. Predicting Alzheimer’s disease progression using deep recurrent neural networks[J]. NeuroImage, 2020,220(10),117203.

[14] Parisot S, Ktena S I, Ferrante E, *et al*. Disease prediction using graph convolutional networks: application to autism spectrum disorder and Alzheimer’s disease[J]. Medical Image Analysis, 2018, 48: 117-130.

[15] Mansu K, Jaesik K, Jeffrey Q, *et al*. Interpretable temporal graph neural network for prognostic prediction of Alzheimer's disease using longitudinal neuroimaging data[C]. BIBM 2021.Houston, TX, USA, 2021:1381-1384.

[16] Fischl B, Salat DH, Busa E, *et al*. Whole Brain Segmentation: Automated labeling of neuroanatomical structures in the human brain[J]. Neuron, 2002, 33(3): 341-355.

[17] Tzouriomazoyer N, Landeau B, Papathanassiou D, *et al*. Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain[J]. Neuroimage, 2002, 15(1): 273-289.

[18] Khazaee A, Ebrahimzadeh A. Application of advanced machine learning methods on resti ng-state fMRI network for identification of mild cognitive impairment and Alzheimer’s di sease [J]. Brain Imaging and Behavior, 2016, 10(3): 799-817.

[19] Benesty J, Chen J, Huang Y, *et al*. Pearson correlation coefficient[M]. Noise reduction in speech processing. Springer, Berlin, Heidelberg, 2009: 1-4.

[20] Wang Y, Ghumare E, Vandenberghe R, Dupont P. Comparison of different generalizations of clustering coefficient and local efficiency for weighted undirected graphs[J]. Neural Computation, 2017, 29, (2):313-331,.

[21] Ma N, Guan J, Zhao Y. Bringing PageRank to the citation analysis[J]. Information Processing & Management, 2008, 44(2): 800-810.

[22] Pareja A, Domeniconi G, Chen J, *et al*. Evolvegcn: Evolving graph convolutional networ ks for dynamic graphs[C]. Proceedings of the AAAI conference on Artificial Intelligence, 2020, 34(4): 5363-5370.

[23] Wang X, Xin J, Wang Z, *et al*. An evolving graph convolutional network for dynamic fun ctional brain network[J]. Applied Intelligence, 2022: 1-14.

Acknowledgement

First and foremost, I would like to express my heartfelt gratitude to my mentor, Professor Yang Xiaochun, for her meticulous guidance and unwavering support. Professor Yang's rigorous academic style, profound knowledge, and approachable demeanor have left an indelible impression on me. From the selection of the research topic to data collection, writing, revision, and finalization of the thesis, her insightful suggestions have enabled me to overcome obstacles and find effective solutions.

I would also like to extend my gratitude to Professor Xin and Ms. Wang from the School of Computer Science and Engineering. It is through their guidance that I delved into the field of computer-aided diagnosis for Alzheimer's disease and dedicated nearly three years of my undergraduate studies to researching diagnostic methods based on computer assistance.

I am deeply grateful to my parents for their upbringing and unwavering support in my endeavors. They have invested significant time and resources in nurturing me and providing me with better education.

In addition, I want to express my sincere appreciation to my numerous classmates who have accompanied me throughout the prolonged period of epidemic control, as well as the intense and stressful phases of exam preparation and competitions during my university years. I extend my thanks to Renbo Zhang for being my companion for almost three years, adding much joy to my otherwise mundane college life. I would like to express my gratitude to the best senior, Tianyi Liu, who has been my guiding light throughout my entire university journey. Without him, I would not have achieved the heights I reached during my college years. I am grateful to Jiaxing Wang, our class monitor, for teaming up with me on multiple course projects, where I learned various cutting-edge technologies through our collaboration. I would like to thank Jiazheng Zhang, my teammate in the university innovation and entrepreneurship competition, for achieving excellent results through our joint efforts. And to Dengpu, Wuyi, Hanshuo, Yangfan and many others, I cherish the fond memories we have shared.

Once again, I would like to express my sincere gratitude to all those who have contributed to the completion of my thesis and supported me throughout my journey. In the future, I will continue to work diligently and strive to live up to everyone's expectations.

Papers, Patents and Awards

(1) Published Papers

[1] 第二作者. 基于时序区分子图的阿尔茨海默症辅助诊断方法[J]. 东北大学学报(自然科学版), 2022 年 8 月 （EI 检索）
[2] 第三作者. 基于动态情绪驱动的人员应急疏散模型[J]. 东北大学学报(自然科学版), 2021 年 11 月（EI 检索）
[3] 第三作者. 基于多模式特征聚合的未来商业预测[J]. 计算机系统应用, 2023 年 2 月

(2) Authorized Software Copyrights

[4] 第一作者. 基于多频融合图核的阿尔茨海默病辅助诊断系统. 计算机软件著作权，授权编号：2022SR0252304
[5] 第一作者. 面向动态脑功能网络的时序区分子图序列搜索系统. 计算机软件著作权，授权编号：2022SR0078905
[6] 第二作者. 基于动态脑网络的阿尔茨海默病辅助诊断系统. 计算机软件著作权，授权编号：2022SR0190037
［7］第三作者. 一种带有高位补偿的坦克驾驶员升降椅[P]. 发明专利，授权编号：ZL202110187587.1

(3) Projects

[8] 2021 第十五批“国家大学生创新训练计划”项目.项目编号:210186, 结题获评:国家级优秀，排名第一.

(4) Scholarships and Honors

[9] 获得国家奖学金 2 次.
[10] 获宝钢优秀学生奖 1 次.
[11] 获辽宁省优秀毕业生称号.

(5) Competition Rewards

[12] 2021 全国大学生英语竞赛，国家级/ 一等奖
[13] 2021 中国高校计算机大赛-团体程序设计天梯赛，国家级/ 三等奖
[14] 2021“海创汇杯”智慧城市与智能建造大学生创新创业竞赛，国家级/ 一等奖
[15] 2021“华为杯”全国大学生物联网设计竞赛全国总决赛，国家级/ 三等奖

東北大學