FMDB *Transactions on Sustainable Computer Letters*

Predicting Brain Diseases from FMRI-Functional Magnetic Resonance Imaging with Machine Learning Techniques for Early Diagnosis and Treatment

Divya Saxena[1,*], Sukaant Chaudhary[2]

[1]Knight Foundation School of Computing and Information Sciences, Florida International University, Miami, Florida, USA.
[2]eCloud Labs Inc, Iseline, New Jersey, United States of America.
dsaxe001@fiu.edu[1], sukaant@gmail.com[2]

Abstract: FMRI classification with neural networks entails training a machine learning model to categorize brain pictures collected from MRI scans. This method uses neural networks to learn complicated patterns and features from MRI data, allowing for accurate and automated classification of brain disorders or structures. Various neural network architectures have been explored and evaluated to determine the most effective model for distinguishing between fMRI images of individuals who are healthy and those who are patients. A promising new method for identifying brain illnesses and examining brain structure is fMRI classification with neural networks. Although this method is still being refined, it has the potential to completely alter how we now identify and treat brain illnesses. The job determines which neural network design should be used for fMRI categorization. For instance, a CNN may be a viable option if the objective is to categorize fMRI data from a particular time point. However, an RNN or LSTM network may be preferable if the objective is to categorize fMRI data from a series of time points. MRI images are a valuable source of information for diagnosing brain illnesses. However, manually classifying MRI images is a time-consuming and challenging task. This is where neural networks come in. Neural networks can be used to automatically classify MRI images with high accuracy. Various combinations of neural networks have been explored to classify MRI images to identify brain illness. In addition to individual neural networks, there have also been some studies that have explored the use of hybrid neural networks. Neural network categorization for MRI is a rapidly developing field. As neural networks get stronger, they can classify MRI pictures more accurately and with more accuracy.

Keywords: Magnetic Resonance Imaging (MRI); Convolutional Neural Network (CNN); Graph Convolution Networks (GCN); Recurrent Neural Network (RNN); Long Short-Term Memory (LSTM); Neuro-Imaging; functional MRI (fMRI); Machine Learning.

Received on: 17/10/2022, **Revised on:** 29/12/2022, **Accepted on:** 27/01/2023, **Published on:** 19/02/2023

Cited by: D. Saxena and S. Chaudhary, "Predicting Brain Diseases from FMRI-Functional Magnetic Resonance Imaging with Machine Learning Techniques for Early Diagnosis and Treatment," *FMDB Transactions on Sustainable Computer Letters*., vol. 1, no. 1, pp. 33–48, 2023.

1. Introduction

The study of the anatomy, connection, and function of the brain and nervous system is the focus of the medical imaging field known as neuroimaging. It is essential to comprehend various neurological and psychological problems and how the brain normally functions. Neuro-imaging is a branch of medical imaging that diagnoses disease and assesses brain health [2] and tells:

[*]Corresponding author.

- How the brain works
- How various activities impact the brain

Various neural network topologies have been investigated and tested to discover the most effective model for discriminating between MRI photographs of healthy and sick people. Several neural networks like convolutional neural networks (CNN, or ConvNet), Graph Convolutional Networks (GCNs) and hybrid models have been evaluated to get medical information from images [1]. A hybrid neural network is a form of artificial neural network that incorporates parts from different neural network architectures or methodologies to capitalize on their unique strengths. This combination enables more sophisticated and powerful models to handle more complex jobs. Hybrid neural networks are created to handle specific difficulties or benefit from diverse data or features. Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Fully Connected Networks (FCNs) are examples of neural network topologies that can be merged to build a single model. This is frequently accomplished by stacking or merging the layers of several topologies, resulting in a network that can handle both spatial and sequential data. Hierarchical hybrid networks have been used. Hierarchical hybrid networks consist of multiple networks, where each level handles different levels of abstraction or aspects of the data. This can be particularly useful for tasks that involve multi-scale analysis. When a single form of neural network architecture is insufficient to address the intricacies of a specific task or dataset, hybrid neural networks are used. They have applications in various fields, including computer vision, natural language processing, healthcare, and others. The problem in developing hybrid networks is balancing computational complexity, training time, and model performance. Four models have been utilized:

1.1. Convolutional Neural Network

A convolutional neural network (CNN, or ConvNet) is a deep neural network often used to analyse visual imagery in deep learning [3]. The term "convolutional neural network" refers to the network's use of the mathematical operation convolution. Convolution is a subset of linear operations. Convolutional networks are simple neural networks with at least one layer that uses convolution instead of ordinary matrix multiplication. MRI images were fed into CNN to classify pictures as Healthy or Patient. CNNs excel in processing grid-like data, such as photographs, where spatial correlations between neighbouring pixels are critical. CNNs use convolutional layers to capture local patterns and hierarchical characteristics in data. They are commonly used for image classification, object identification, and picture segmentation. An independent test dataset was used to evaluate the model's performance.

1.2. Graph Convolutional Networks (GCNs)

Currently, most graph neural network models have a somewhat universal architecture in common. I will refer to these models as Graph Convolutional Networks (GCNs) [4], convolutional because filter parameters are typically shared over all locations in the graph. For these models, the goal is then to learn a function of signals/features on a graph G = (V, E), which takes as input:

- A feature description xi for every node i; summarized in an N×D feature matrix X (N: number of nodes, D
- number of input features)
- A representative description of the graph structure in matrix form, typically in the form of an adjacency matrix A
- (or some function thereof) and produces a node-level output Z (an N×F feature matrix, where F is the number of output features per node). Graph-level outputs can be modelled by introducing some form of pooling operation.

2D Images have been given as Input to GCN for binary classification of images as Healthy and Patient.
GCNs are intended to deal with graph-structured data, in which nodes represent entities and edges represent relationships or connections between them. They aggregate information from surrounding nodes to conduct convolution-like actions on the graph. GCNs are frequently used for node classification, connection prediction, and graph-level classification.

1.3. GCN + CNN Hybrid Model

A hybrid neural network architecture combining Graph Convolutional Networks (GCNs) and Convolutional Neural Networks (CNNs) to process data with spatial and graph-like features is an example of a hybrid neural network architecture. This hybrid technique is especially beneficial for applications that require data with both image-like patterns and relational connections. Combining GCNs and CNNs can be especially useful when the data has spatial and relational features. For example, data in activities such as assessing social networks or examining brain connectivity may include both graph-like associations and image-like properties.

1.4. GCN + RNN Hybrid Model

RNNs excel at analysing sequential or temporal data. They keep hidden states where they capture information from prior time steps and use it to process the present input. RNNs are commonly employed in applications such as natural language processing, time series prediction, and sequence-to-sequence matching [5]. A hybrid neural architecture capable of processing data with both graph structures and sequential/temporal dependencies is created by combining Graph Convolutional Networks (GCNs) with Recurrent Neural Networks (RNNs). This method is excellent for jobs requiring relational and temporal information, such as evaluating data from social networks, molecular chemistry, and spatiotemporal data from many fields. The main advantage of this hybrid design is its ability to manage data with graph-based relationships and temporal dependencies. It is critical to properly design the combination of GCNs and RNNs [6] and consider elements like data pretreatment and model regularization to get the best performance on your specific assignment.

2. Literature Review

There are several techniques used in neuroimaging to capture different aspects of brain activity and structure:

2.1. Structural Imaging

- Computed Tomography (CT): This method uses X-rays taken at various angles around the head to produce cross-sectional images of the brain's structure. It's very helpful for spotting structural anomalies like tumours or bleeding. With the help of strong magnetic fields and radio waves,
- Magnetic resonance imaging (MRI) may produce precise brain tissue pictures. It is adaptable and can produce ionizing radiation-free, high-resolution images of the brain's internal architecture. Different MRI sequences can highlight various tissue types, including cerebrospinal fluid, white matter, and gray matter.

2.2. Functional Imaging

- Functional Magnetic Resonance Imaging (fMRI): The fMRI method (functional magnetic resonance imaging) uses blood flow and oxygenation alterations to estimate brain activity. It is frequently employed to investigate brain areas that light up when engaged in particular activities or reactions to stimuli, assisting researchers in understanding cognitive processes and functional connectivity.
- Positron Emission Tomography (PET): Positron Emission Tomography (PET) scans involve injecting a small quantity of radioactive material into the body, which causes it to release positrons that the scanner can detect. One can view brain activity by monitoring the radioactive substance's distribution, which is frequently connected to glucose metabolism or particular neurotransmitter receptors.

2.3. Connectivity Imaging

- Diffusion Tensor Imaging (DTI): Diffusion Tensor Imaging (DTI) is a specific type of MRI used to map the routes of the brain's white matter tracts. It investigates the anatomical connection and communication among various brain regions.
- Resting-State fMRI: This method looks at brain activity while a person is at rest and identifies patterns of intrinsic brain connection. This technique is performed to explore functional connectivity and find networks of regions that frequently activate together.

Neuroimaging [8] has numerous applications in scientific and therapeutic settings. It's critical for researching brain development, detecting neurological disorders (including Alzheimer's, stroke, epilepsy, and multiple sclerosis), designing neurosurgical procedures, and furthering our understanding of brain function in health and disease. However, interpreting neuroimaging data takes expertise because of the complexity of the human brain's structure and function.

Medical imaging uses machine learning [8] to enhance disease diagnosis, treatment, and monitoring. Here are some instances of applications of machine learning in medical imaging. The science of machine learning is expanding quickly, and new applications are being developed in medical imaging. More and more jobs in the medical imaging workflow will be able to be automated as machine learning algorithms advance in sophistication. Because of this, Doctors and other healthcare workers will be able to concentrate on more difficult duties, such as patient care and treatment planning.

Despite many difficulties, machine learning is an innovative technique that has the potential to completely change the medical imaging industry. Machine learning [7] will play a bigger role in disease diagnosis, treatment, and monitoring as machine learning algorithms get more advanced and more medical picture data are available. Some examples of how machine learning is used in neuroimaging research are to identify patterns in brain activity, classify brain disorders, predict the progression of brain disorders, develop new treatments for neurological disorders and many more.

3. Dataset Preparation

Pre-processing MRI (Magnetic Resonance Imaging) data is essential for ensuring its quality, consistency, and accuracy before analysing it or feeding it into machine learning models. Sample images of the dataset used:

3.1. Healthy Person

A healthy person's brain will normally seem normal on an MRI scan, showing no symptoms of disease or injury. The cerebrospinal fluid (CSF) will be clear, and the brain tissue will be a consistent shade of greyish-white. The brain's blood vessels will appear as bright white lines. Figure 1 shows the MRI scan of a healthy person.

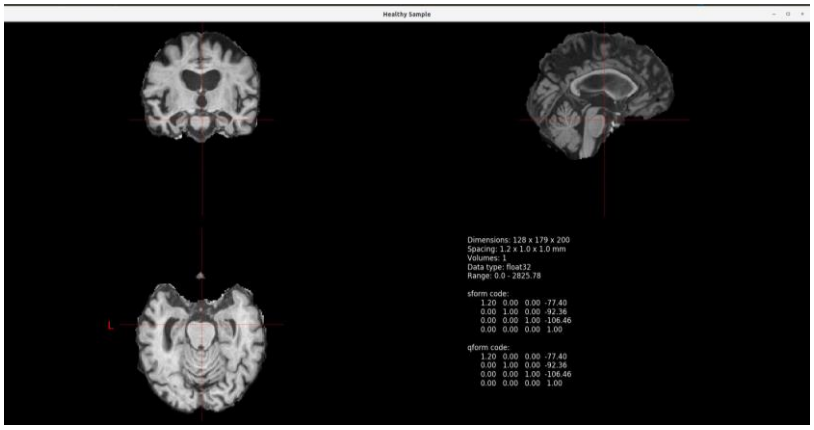


Figure 1: MRI scan of a healthy person

3.2. Patient

The MRI scan may reveal various abnormalities depending on the patient's state. On a patient's MRI scan, frequent anomalies that could be discovered include Tumors, Strokes, Infections, Degenerative diseases etc. Figure 2 shows the MRI scan of a patient having Alzheimer's disease.

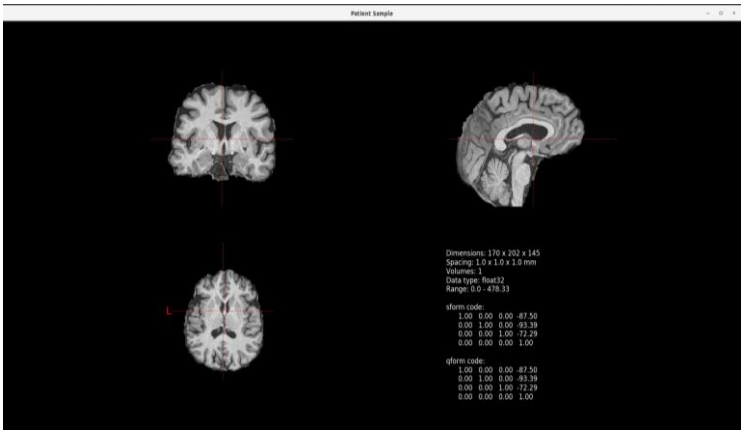


Figure 2: MRI scan of the patient

Two datasets have been used for classification:

3.3. average_fmri_feature_matrix

The word "average_fmri_feature_matrix" relates to a feature matrix obtained from fMRI data. fMRI is a neuroimaging technology that uses blood oxygenation and flow variations to infer neural activity in the brain. According to the name, this feature matrix is created by averaging specific features taken from fMRI data. Researchers frequently pre-process raw fMRI data in fMRI analysis to eliminate noise, adjust for artefacts, and extract useful features. These characteristics could include

brain activity measurements in specific regions or networks, connections between distinct brain regions, or statistical measures relating to activation patterns. An "average_fmri_feature_matrix" might be generated by averaging these retrieved features over numerous fMRI scans, as shown in Figure 3. This could be done to create a typical brain activity or connectivity profile that typifies a specific situation or task. Remember that the exact meaning of "average_fmri_feature_matrix" may change depending on the context and the methods used to analyze the fMRI data. It is best to consult the source or documentation where you found the problem. Matrices with the size of 150x150

- Resized to 112x112: In machine learning, resizing images is a typical procedure. This is so because many machine learning models, such as convolutional neural networks, anticipate a certain size for their input data.
- Flatten into a 1-dimensional array: Flattening a multidimensional array into a 1-dimensional array is a common operation in machine learning. This is because many machine learning models, such as neural networks, expect their input data to be in a 1-dimensional array.

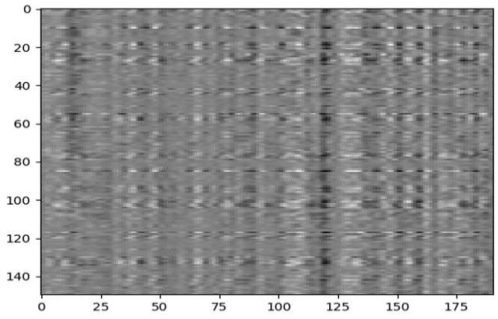


Figure 3: Sample of average_fmri_feature_matrix

3.4. common_fiber_matrix

In the context of MRI, the phrase "common_fiber_matrix" most likely refers to a matrix that shows the connectivity or interaction between distinct brain regions or locations, with a particular emphasis on the concept of white matter tracts or fibres, as shown in Figure 4. Researchers frequently investigate connection patterns inside the brain in functional magnetic resonance imaging (fMRI) and diffusion tensor imaging (DTI) to understand how various regions communicate and operate together. White matter tracts are bundles of nerve fibres that connect different brain areas and allow information to be transmitted. The creation and interpretation of a "common_fiber_matrix" would be determined by the research question and methodology employed. Such matrices are frequently used in studies examining brain connections, functional networks, and how brain regions collaborate to conduct various cognitive and sensory processes. Matrices with the size of 150x150

- Resized to 112x112: Resizing images is a common operation in machine learning. This is because many machine learning models, such as convolutional neural networks, expect their input data to be of a specific size.
- Flatten into the 1-dimensional array: A frequent operation in machine learning is to flatten a multidimensional array into a 1-dimensional array. Many machine learning models, including neural networks, anticipate having a 1-dimensional input data array.

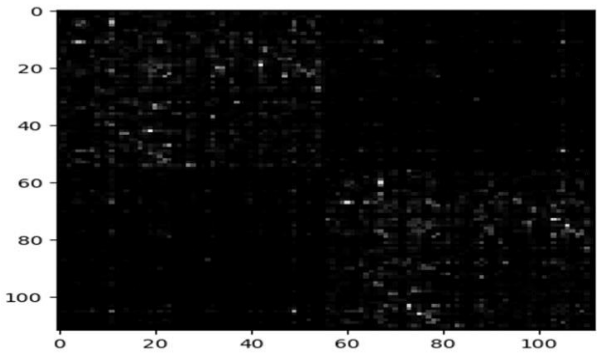


Figure 4: Sample of common_fiber_matrix

The following are common MRI image preparation steps:

Two Python files are created to process the datasets as below:

- /Cnn_Mri_Classification/Data_preprocessing/preprocess_data.py
- /Cnn_Mri_Classification/Data_preprocessing/preprocessing_testing.py

The below steps have been followed for both Training and Testing datasets, but testing data is kept separately to avoid over-fitting the model.

The code has been implemented as below:

Unzip .nii in patient_raw, health_raw:
nii raw MRI images will be saved in a folder separately for training and testing i.e.
(/Cnn_Mri_Classification/6389_project1/6389_project1/Training/health_raw,/Cnn_Mri_Classification/6389_project1/6389_project1/Training/patient_raw) and
(/Cnn_Mri_Classification/6389_project1/6389_project1/Testing/health_raw,/Cnn_Mri_Classification/6389_project1/6389_project1/Testing/patient_raw)

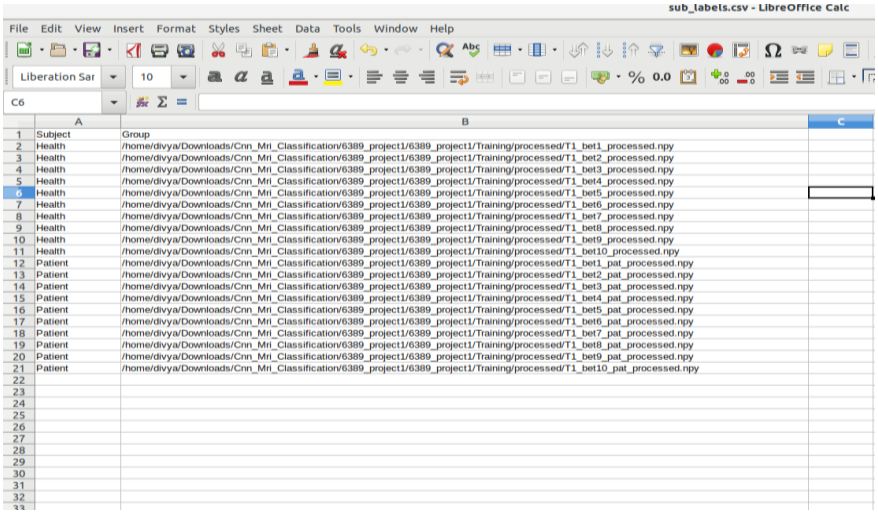
Filtered .nii to .npy in folder(patient_filtered, health_ filtered):
Images pre-processed differently also have a different head orientation, so they were removed. This is done by filtering the dimensions. By checking, it is found that if the last dimension s[2] is larger than the first dimension s[0], then the head orientation in this image is different to the majority. Also, the image was pre-processed differently can be seen from the name of the file.
(/Cnn_Mri_Classification/6389_project1/6389_project1/Training/health_filtered,/Cnn_Mri_Classification/6389_project1/6389_project1/Training/patient_filtered)and
(/Cnn_Mri_Classification/6389_project1/6389_project1/Testing/health_filtered,/Cnn_Mri_Classification/6389_project1/6389_project1/Testing/patient_filtered)

Processed all healthy/patients to the processed folder:
Pre-processing of original images. Pre-processing includes skull-stripping, brain cropping, resizing and intensity normalisation. MRI Image has been resized to 64*64*64 due to limited GPU and CPU.
(/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed)And
(/Cnn_Mri_Classification/6389_project1/6389_project1/Testing/processed)

Final step is after pre-processing, data has been split into two parts :
Training and Validation. All the data has been converted into single files .npy. And Testing datasets as well. All files have been created in the Input folder as data eir corresponding labels.

- test_data.npy - Array of Input images for Testing dataset
- test_label.npy - Array of Labels of Input images for Testing dataset
- train_data.npy - Array of Input images for Training dataset
- train_label.npy - Array of Labels of Input images for Training dataset
- val_data.npy - Array of Input images for Validation dataset
- val_label.npy - Array of Labels of Input images for Validation dataset

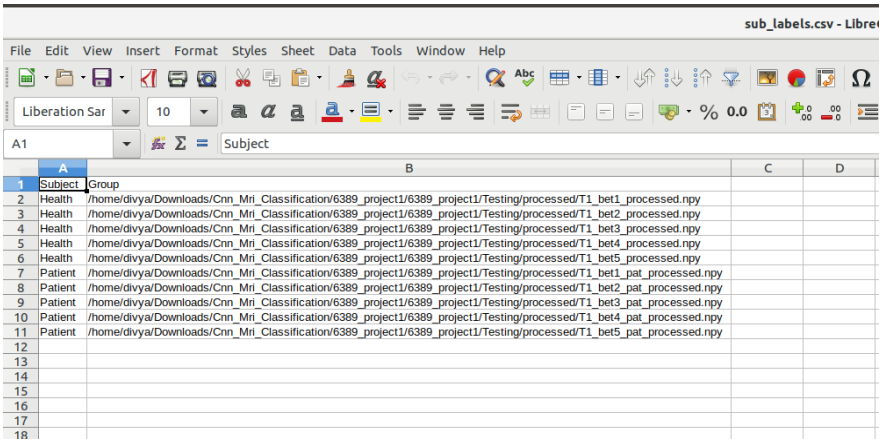
Some manual steps required for Training the model: Created labels for Training/Validation in sub_label.csv (refer to Figure 5) as Subject and Group as the heading at location /Cnn_Mri_Classification/6389_project1/6389_project1/Training



	A	B
1	Subject	Group
2	Health	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet1_processed.npy
3	Health	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet2_processed.npy
4	Health	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet3_processed.npy
5	Health	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet4_processed.npy
6	Health	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet5_processed.npy
7	Health	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet6_processed.npy
8	Health	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet7_processed.npy
9	Health	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet8_processed.npy
10	Health	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet9_processed.npy
11	Health	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet10_processed.npy
12	Patient	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet1_pat_processed.npy
13	Patient	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet2_pat_processed.npy
14	Patient	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet3_pat_processed.npy
15	Patient	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet4_pat_processed.npy
16	Patient	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet5_pat_processed.npy
17	Patient	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet6_pat_processed.npy
18	Patient	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet7_pat_processed.npy
19	Patient	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet8_pat_processed.npy
20	Patient	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet9_pat_processed.npy
21	Patient	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Training/processed/T1_bet10_pat_processed.npy

Figure 5: Labels file created for Training

Created labels as shown in Figure 6 for Testing in sub_label.csv as Subject and Group as the heading at location /Cnn_Mri_Classification/6389_project1/6389_project1/Testing



	A	B
1	Subject	Group
2	Health	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Testing/processed/T1_bet1_processed.npy
3	Health	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Testing/processed/T1_bet2_processed.npy
4	Health	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Testing/processed/T1_bet3_processed.npy
5	Health	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Testing/processed/T1_bet4_processed.npy
6	Health	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Testing/processed/T1_bet5_processed.npy
7	Patient	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Testing/processed/T1_bet1_pat_processed.npy
8	Patient	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Testing/processed/T1_bet2_pat_processed.npy
9	Patient	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Testing/processed/T1_bet3_pat_processed.npy
10	Patient	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Testing/processed/T1_bet4_pat_processed.npy
11	Patient	/home/divya/Downloads/Cnn_Mri_Classification/6389_project1/6389_project1/Testing/processed/T1_bet5_pat_processed.npy

Figure 6: Labels file created for testing

4. Models Used

There are various neural networks which have been tried and tested:

4.1. CNN Model Architecture

Convolutional neural networks (CNNs) are specifically designed to process image data. CNN Model architecture is as follows (Figure 7):

- Conv1D(64, kernel_size=(2), activation='elu')
- Conv1D(32, kernel_size=(2), activation='elu')
- Dense(512, activation='relu')
- Dense(total_classes, activation='softmax')

```
Model: "cnn_net"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            multiple                  192
_________________________________________________________________
conv1d_2 (Conv1D)            multiple                  4128
_________________________________________________________________
flatten_2 (Flatten)          multiple                  0
_________________________________________________________________
dense_4 (Dense)              multiple                  205488640
_________________________________________________________________
dense_5 (Dense)              multiple                  1026
=================================================================
Total params: 205,493,986
Trainable params: 205,493,986
Non-trainable params: 0
```

Figure 7: CNN Model architecture

4.2. GCN Model Architecture

Graph convolutional networks (GCNs) are a type of neural network specifically designed for processing graph-structured data. GCN Model architecture is as follows (Figure 8):

- GraphConv(64, activation='elu')
- GraphConv(32, activation='elu')
- Dense(512, activation='relu')
- Dense(total_classes, activation='softmax')

```
Model: "graph_net"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
graph_conv (GraphConv)       multiple                  128
_________________________________________________________________
graph_conv_1 (GraphConv)     multiple                  2080
_________________________________________________________________
flatten (Flatten)            multiple                  0
_________________________________________________________________
dense (Dense)                multiple                  205521408
_________________________________________________________________
dense_1 (Dense)              multiple                  1026
=================================================================
Total params: 205,524,642
Trainable params: 205,524,642
Non-trainable params: 0
```

Figure 8: GCN Model architecture

4.3. GCN + CNN hybrid model Architecture

Creation of adjacency matrix:

- based on the following work
- “Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering”
- The K-NN graph is statically determined from a regular grid of pixels using the 2d coordinates.
- Every pixel is a node.
- Two nodes are connected if they are neighbours.

According to the K-NN graph (Figure 9). The node features of each graph are the images vectorized and rescaled to [0,1].

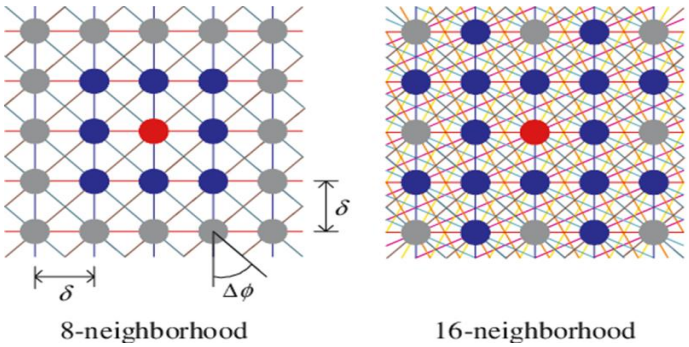


Figure 9: K-NN graph for 8 and 16 neighbourhood

GCN and CNN are both neural network architectures used for machine learning tasks. However, they have different strengths and weaknesses. Model architecture is as follows (Figure 10):

- GraphConv(128, activation='elu')
- GraphConv(64, activation='elu')
- Conv1D(32, kernel_size=(2), kernel_initializer = 'he_uniform')
- Dense(512, activation='relu')
- Dense(total_classes, activation='softmax')

```
Model: "graph_cnn_net"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
graph_conv_2 (GraphConv)     multiple                  256
_________________________________________________________________
graph_conv_3 (GraphConv)     multiple                  8256
_________________________________________________________________
conv1d (Conv1D)              multiple                  4128
_________________________________________________________________
flatten_1 (Flatten)          multiple                  0
_________________________________________________________________
dense_2 (Dense)              multiple                  205505024
_________________________________________________________________
dense_3 (Dense)              multiple                  1026
=================================================================
Total params: 205,518,690
Trainable params: 205,518,690
Non-trainable params: 0
```

Figure 10: GCN+CNN hybrid Model architecture

4.4. GCN + RNN hybrid model Architecture

A GCN + RNN hybrid architecture can combine the strengths of both GCNs and RNNs. The GCN can be used to learn representations of the nodes in the graph, and the RNN can be used to process the sequential data associated with each node. Model architecture is as follows (Figure 11):

- GraphConv(128, activation='elu')
- GraphConv(64, activation='elu')
- LSTM(32, kernel_initializer = 'he_uniform')
- Dense(512, activation='relu')
- Dense(total_classes, activation='softmax')

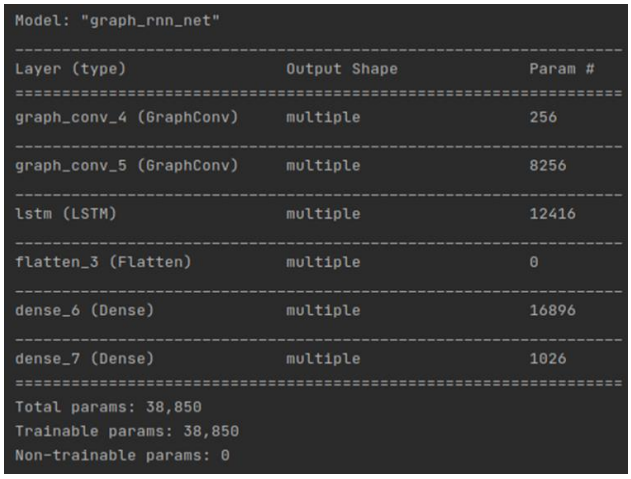


Figure 11: GCN+RNN hybrid Model architecture

5. Methods and Parameters Used

Some of the parameters which are used in training different models are as follows:

- Batch size: Batch size is a hyperparameter in machine learning that controls the number of training examples processed at a time. Larger batch sizes can improve the performance of machine learning models but also slow the training process. Here, batch size = 32.
- Number of epochs: In machine learning, an epoch is a single pass through the entire training dataset. The number of epochs is a hyperparameter that controls how often the model will see the entire training dataset during training. Here, the number of epochs = 50.
- Learning Rate: The learning rate is a hyperparameter in machine learning that controls how much the model's parameters are updated during training. A larger learning rate can lead to faster training, but it can also make the model more likely to overfit the training data. A lower learning rate can lead to slower training, making the model more likely to converge to a good solution. Here, Learning Rate = le-3.

5.1. Optimizer

Adam Optimizer has been used. The Adam optimizer is a popular optimization algorithm in machine learning, and it is used for problems such as MRI image categorization. Because of its variable learning rate and momentum-based strategy, it is especially useful for training neural network models. Here is how the Adam optimizer can be used for MRI categorization. Implemented the Adam optimizer in chosen deep learning framework (such as TensorFlow or PyTorch). The optimizer requires setting parameters like learning rate, beta1, beta2, and epsilon.

5.2. Loss

SparseCategoricalCrossentropy() has been used. SparseCategoricalCrossentropy() is a loss function commonly used in classification tasks where the target labels are integers representing class categories, and the model's output produces a probability distribution over these categories. It's often used for multi-class classification problems where each data point belongs to a single class out of multiple possible classes. The term "sparse" categorical cross-entropy refers to the fact that it does not require one-time encoding of the target labels, which can be memory-intensive for large classification tasks. Instead, it computes the cross-entropy loss using the integer labels directly. The models are compiled with the Adam optimizer and the SparseCategoricalCrossentropy() loss function. During training, the model's outputs will be directly compared with the integer labels to compute the loss. The accuracy metric is also added to monitor the model's classification performance.

5.3. Accuracy

SparseCategoricalAccuracy() is a metric often used to assess the predictive accuracy of a model in multi-class classification problems when the target labels are integers representing class categories. This metric is frequently utilised when working with integer-encoded target labels rather than one-hot encoded labels. The models are compiled with the Adam optimizer, the

sparse_categorical_crossentropy loss function, and the SparseCategoricalAccuracy() metric. During training and evaluation, the metric will calculate the accuracy by comparing the highest predicted class with the true integer label for each data point.

5.4. Overfitting

Overfitting is a typical problem in machine learning and deep learning in which a model performs well on training data but struggles to generalize to new, previously unseen data. In other words, the model learns to memorise the training examples instead of capturing the fundamental patterns that would allow it to make correct predictions on new data. We also faced overfitting. So, to prevent overfitting, we followed some steps:

- Gradually decreasing the learning rate: The convergence and stability of the training process are improved by gradually lowering the learning rate in neural networks. A low learning rate can lead the model to converge too slowly or become stuck in a local minimum, while a high learning rate can cause the model to quickly overshoot the ideal solution. The model can change its weights more precisely and steer clear of these issues by gradually lowering the learning rate.
- Using L2 regularization: L2 regularization, commonly called weight decay, avoids overfitting in neural networks. It operates by adding a penalty to the loss function whose value is inversely proportionate to the network's squared weights.

This penalty makes the network's weights smaller, preventing the network from overly fitting the training data and producing inaccurate predictions for brand-new data. The Regularization factor = 4e-4 has been used. The formula is: loss = (1 - lambda) * original_loss + lambda * sum(w^2)

Where:

- loss is the total loss of the network
- original_loss is the loss without L2 regularization
- lambda is a hyperparameter that controls the strength of the regularization
- w is a weight in the network
- w^2 is the square of the weight
- Using dropout: A regularization method for neural networks is called dropout. During training, a certain proportion of the neurons in a layer are randomly eliminated (set to zero). This prevents overfitting by making the network learn to rely on more neurons than just a few.

In Figure 12, we see the learning rate drops with increasing the number of epochs to avoid overfitting.

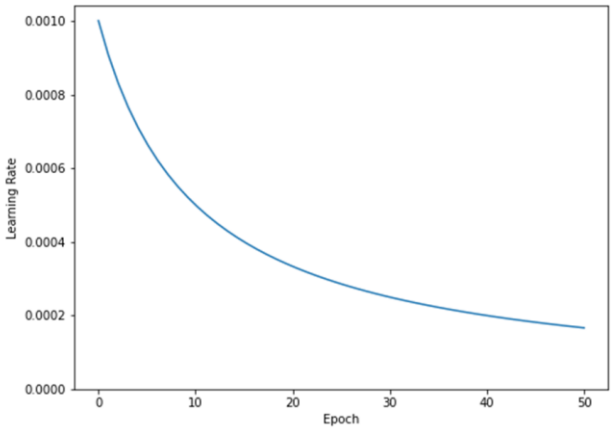


Figure 12: To handle overfitting.

6. Results for the "common_fiber_matrix" dataset

6.1. Loss During Training

When assessing the effectiveness of a machine learning model, loss during training is another crucial measure to take into account. Loss is a metric to gauge how effectively a model can foretell the training data labels. A lower loss shows that the model is more effectively predicting the training data labels. Initially, during training, the Hybrid model (GCN and CNN) losses

more than other models. After 50 epochs, the Hybrid model (GCN and RNN) still has a higher loss than others. While other models trained well after 50 epochs. Refer to Figure 13.

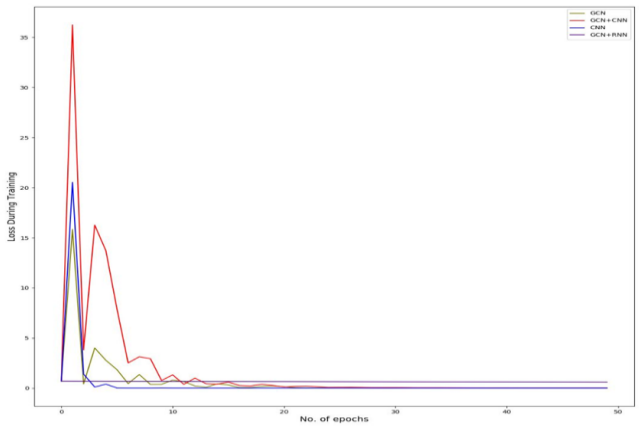


Figure 13: Graph to show Loss During Training for the "common_fiber_matrix" dataset

6.2. Accuracy During Training

Accuracy during training is also an important metric to consider when evaluating the performance of a machine-learning model. Accuracy is the percentage of training images correctly classified by the model. A higher accuracy indicates that the model is learning the training data well. CNN has the highest accuracy. Note: Training this model for another 50 epochs results in substantial overfitting. After CNN, GCN performed well based on the graph. After GCN, the hybrid model (GCN and CNN) performed well. Lastly, it's a hybrid model (GCN and CNN). Refer Figure 14.

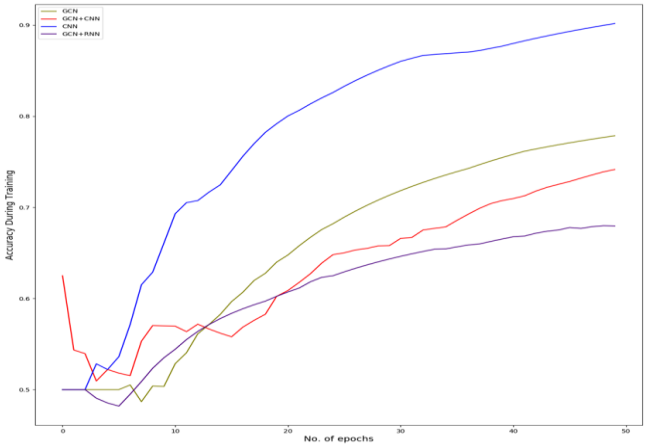


Figure 14: Graph to show Accuracy During Training for the "common_fiber_matrix" dataset

6.3. Accuracy During Testing

When assessing the effectiveness of a machine learning model, accuracy during testing is a crucial measure to take into account. The proportion of test photos the model correctly categorizes is known as accuracy. A higher accuracy shows that the model is operating more effectively. CNN has the highest accuracy. After CNN, GCN performed well based on the graph. After GCN, the hybrid model (GCN and CNN) performed well. Lastly, it's a hybrid model (GCN and CNN). Refer to Figure 15.

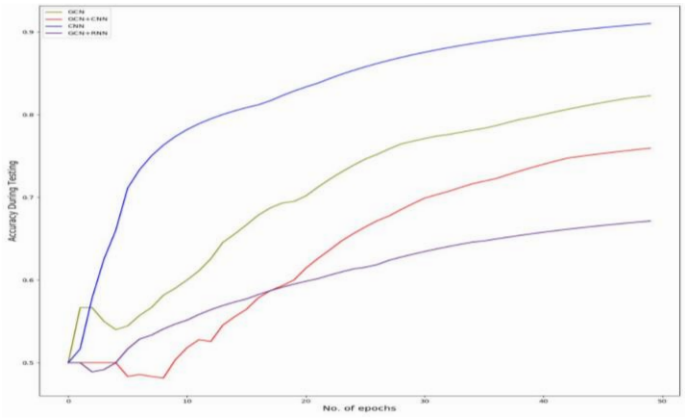


Figure 15: Graph to show Accuracy during Testing for the "common_fiber_matrix" dataset

6.4. Final Accuracies on the test dataset

The final accuracies on the test dataset for different machine learning models can vary depending on the specific dataset and the model architecture. The final accuracy (refer to Figure 15) on the test dataset on a scale of 0 to 1 are:

- GCN + CNN hybrid model is 0.741
- GCN + RNN hybrid model is 0.6787
- CNN is 0.90
- GCN is 0.777

7. Results for the "average_fmri_feature_matrix" dataset

7.1. Loss During Training

Loss during training is also an important metric to consider when evaluating the performance of a machine-learning model. Loss is a measure of how well the model predicts the training data labels. A lower loss indicates that the model predicts the training data labels more accurately. Initially, during training, the Hybrid model (GCN and RNN) losses more than other models. After 50 epochs, the Hybrid model (GCN and RNN) still has a higher loss than others. While other models trained well after 50 epochs. Refer to Figure 16.

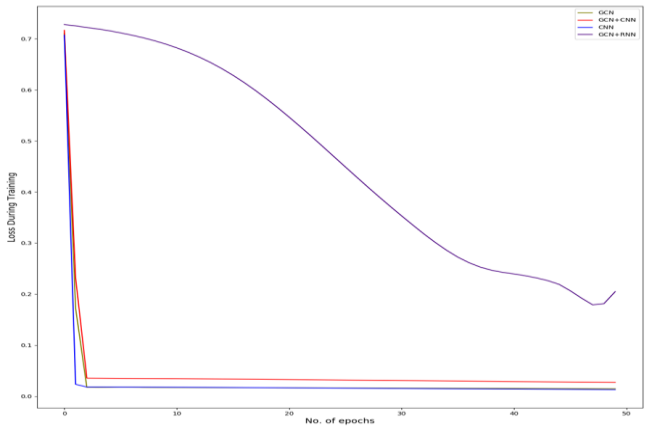


Figure 16: Graph to show Loss During Training for the "average_fmri_feature_matrix" dataset.

7. 2. Accuracy During Training

Another crucial measure to take into account when assessing the performance of a machine learning model is accuracy during training. The percentage of training photos the model correctly categorizes is known as accuracy. A higher accuracy shows that

the model is successfully absorbing the training set of data. GCN has the highest accuracy. After GCN, CNN performed well based on the graph. After CNN, the hybrid model (GCN and CNN) performed well. Lastly, it's a hybrid model (GCN and RNN). Refer to Figure 17.

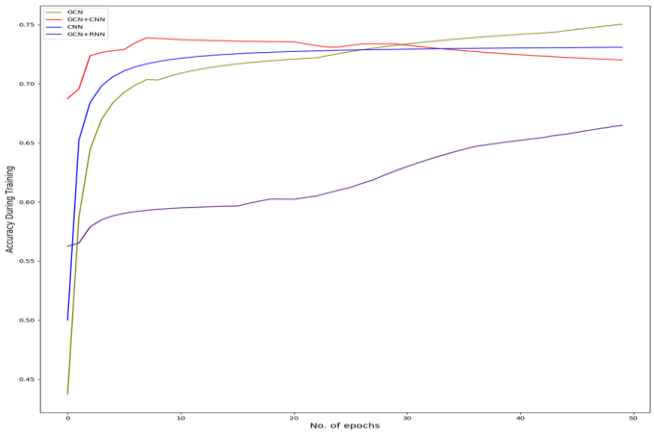


Figure 17: Graph to show Accuracy During Training for the "average_fmri_feature_matrix" dataset.

7. 3. Accuracy During Testing

Accuracy during testing is an important metric to consider when evaluating the performance of a machine-learning model. Accuracy is the percentage of test images correctly classified by the model. A higher accuracy indicates that the model is performing better. GCN has the highest accuracy. After GCN, CNN performed well based on the graph. After CNN, the hybrid model (GCN and CNN) performed well. Lastly, it's a hybrid model (GCN and CNN). Refer to Figures 18.

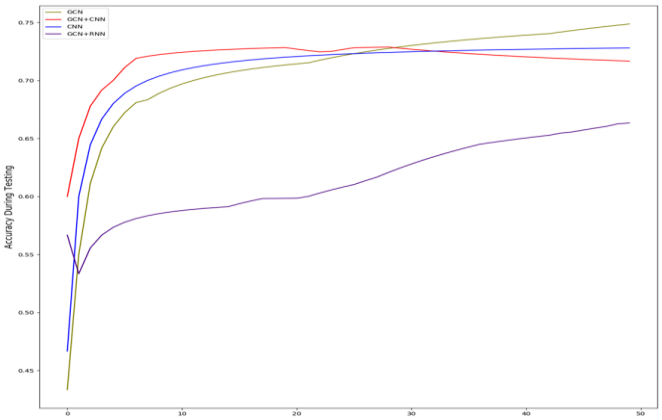


Figure 18: Graph to show Accuracy During Testing for the "average_fmri_feature_matrix" dataset

7.4. Final Accuracies on the test dataset

Depending on the particular dataset and the model architecture, the ultimate accuracies on the test dataset for various machine learning models can change. The final accuracy (refer to Figure 18) on the test dataset on a scale of 0 to 1 are:

- GCN + CNN is 0.7161
- GCN + RNN is 0.663
- CNN is 0.728
- GCN is 0.748

Table 1: Final accuracies of different models used for two datasets

Datasets / Models	average_fmri_feature_matrix	common_fiber_matrix
GCN + CNN	0.7161	0.741
GCN + RNN	0.663	0.6787
CNN	0.728	0.90
GCN	0.748	0.777

8. Conclusion

Several models have been tested for two datasets to determine which model can be used to test MRI scans to classify brain images between a healthy person and a patient. The best model for classifying brain images between healthy people and patients will vary depending on the dataset and tand. It is also important to consider the other metrics, such as precision, recall, and F1 score, when evaluating the performance of a machine learning model. Once the models have been trained and tested, the results can be analyzed to determine which model performs the best. The "Accuracy" metrics have been used to evaluate the performance of the models. Accuracy is the percentage of images that are correctly classified. The model with the highest accuracy is the best for testing MRI scans. Below, Table 1 shows the accuracies of different neural network architectures on both datasets, i.e. "average_fmri_feature_matrix" and "common_fiber_matrix". GCN + CNN hybrid model performed almost the same for both datasets with +-.05. GCN + RNN hybrid model is not a good model, which has performed worst in all the models tested. CNN model performed well for the common_fiber_matrix dataset (approximately 90% accuracy). GCN performed almost the same for both datasets with +-.04

Acknowledgement: We thank our family whose prayers have got us here. Their belief in our vision helped us achieve our goal. We are also extremely grateful to our friends who stood by us patiently during the research.

Data Availability Statement: The data, graphs, and code that support the findings of this study are available from the corresponding author upon reasonable request.

Funding Statement: No funding was received to conduct the research.

Conflicts of Interest Statement: Authors collectively produce this work where they all agree with the work's points, issues and findings.

Ethics and Consent Statement: Authors of the work unanimously consent to make this publication available to all interested people for reading, teaching and learning.

References

1. D. Saxena, "A Non-Contact Based System to Measure SPO2 and Systolic/Diastolic Blood Pressure using Rgb-Nir Camera."; Order No. 29331388, The University of Texas at Arlington, United States -- Texas, 2022.
2. S. A. H. Batouli et al., "Iranian Brain Imaging Database: A neuropsychiatric database of healthy brain," Basic Clin. Neurosci., vol. 12, no. 1, pp. 115–132, 2021.
3. V. Bhanumathi and R. Sangeetha, "CNN Based Training and Classification of MRI Brain Images," 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), Coimbatore, India, pp. 129-133, 2019, doi: 10.1109/ICACCS.2019.8728447.
4. Z. Qin, Z. Liu and P. Zhu, "Aiding Alzheimer's Disease Diagnosis Using Graph Convolutional Networks Based on rs-fMRI Data," 2022 15th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Beijing, China, pp. 1-7, 2022, doi: 10.1109/CISP-BMEI56279.2022.9980159.
5. E. Z. Chen, P. Wang, X. Chen, T. Chen and S. Sun, "Pyramid Convolutional RNN for MRI Image Reconstruction," in IEEE Transactions on Medical Imaging, vol. 41, no. 8, pp. 2033-2047, 2022, doi: 10.1109/TMI.2022.3153849.

6. Y. Gao, J. M. Phillips, Y. Zheng, R. Min, P. T. Fletcher and G. Gerig, "Fully convolutional structured LSTM networks for joint 4D medical image segmentation," 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), Washington, DC, USA, pp. 1104-1108, 2018, doi: 10.1109/ISBI.2018.8363764.
7. S. Dahiya, S. Vijayalakshmi and M. Sabharwal, "Alzheimer's Disease Detection using Machine Learning: A Review," 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, pp. 288-293, 2021, doi: 10.1109/ICAC3N53548.2021.9725703.
8. S. R. Sandeep, S. Ahamad, D. Saxena, K. Srivastava, S. Jaiswal, and A. Bora, "To understand the relationship between Machine learning and Artificial intelligence in large and diversified business organizations," Materials Today: Proceedings, vol. 56, pp. 2082–2086, 2022.
9. V. Patil and S. L. Nisha, "Detection of Alzheimer's Disease Using Machine Learning and Image Processing," 2021 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON), Pune, India, pp. 1-5, 2021, doi: 10.1109/SMARTGENCON51891.2021.9645743.