Automatic Identification of Neuropsychiatric Disorders from Brain Networks Using Deep Learning

Software Project Final Report

Team: MA_B_2

Jun Yi Choo
30219396
jcho0057@student.monash.edu

Teng Ian Khoo
30242452
tkho0004@student.monash.edu

Zhong Kern Fong
30284104

zfon0001@student.monash.edu

Supervisor : A/Prof Ting Chee Ming

Word count: 10,619

28 May 2021

Contents

1 Introduction

The most fascinating, complicated and crucial human organ in the human body is the brain. It acts as a control centre where it controls the homeostatic balance of every human system, coordinates movements and actions, enables us to feel emotions and mach more. With many medical studies and research being performed over many centuries, there is still much more that is yet to be discovered. Every organ of the human body is susceptible to physical injuries, diseases and disorders that is why finding out and treating the causes of these ailments is such an important part of human society. This also includes the brain, where it is vulnerable to a class of ailments called neuropsychiatric disorders such as Alzheimer's disease (AD), major depressive disorder (MDD), Huntington's disease (HD), Parkinson's disease, Amyotrophic Lateral Sclerosis (ALS) and Schizophrenia. With the recent outbreak of Covid-19 pandemic, a study published by The Lancet Psychiatry revealed that 1 in 3 Covid-19 patients are diagnosed with a neuropsychiatric condition within 6 months with the most common illness being anxiety and depression (Taquet et al., 2021).

Major depressive disorder is the most prevalent neuropsychiatric disorder and is the highest contributor to the overall global burden of diseases (Depression 2020). MDD is accountable for influencing emotions, general behaviours and ability to generate negative thoughts. Some core symptoms of MDD includes anxiety, fatigue, sexual dysfunction and sleep disturbance which validates that MDD heavily alters the average lifestyle of a human being (Kennedy, 2008). The growth and improvements in the field of technology with particular regard to deep learning has given potential to perform early detection of neuropsychiatric disorders before it advances into a chronic state.

This project report is committed to exploring different methodology and research on various deep learning architectures, specifically state-of-the-art Convolutional Neural Network (CNN), Multi Layered Perceptron (MLP) and a hybrid CNN and Recurrent Neural Network (RNN) called CNN-LSTM models and put these architectures into use by predicting potential MDD patients from a control. The training of the Deep Learning models will be based on the dataset provided by Vanderbilt University after preprocessing it. As the initial dataset provided is a 3D functional Magnetic Resonance Imaging (fMRI) time series data, connectivity matrices are extracted from it. After that, the connectivity matrices will be used to classify between control patients and patients that are potentially suffering from MDD or is suffering from MDD.

2 Background

In the following section below, included are literature, journals and research papers that are related to the proposed project topic, that is 'Automatic Identification of MDD patients using Deep Learning models'. It will be divided into two sections, literature review and summary. In the review section, it discusses on the achievements of CNN architecture, differences between fMRI and rs-fMRI and general architectures of deep learning model. In the literature summary, included are several research paper and journals that are related to the project that aligns with the deep learning model that was developed for the project.

2.1 Literature Review

ImageNet Large-Scale Visual Recognition Challenge (ILSVRC2012) was held on 2012 and AlexNet took the world by surprise when it won over the first runner-up with 11% difference in error (Russakovsky, O. et al, 2015). Since then, there are many variations of CNNs are designed around the base model of AlexNet. The architecture of CNNs has evolved from 22-layer deep GoogLeNet and VGG in 2014, 34-layer deep ResNet in 2015 to MobileNetV3 in 2019 which goes to show how much CNN architectures has evolved to attain better accuracy in rapidly changing challenging tasks. With new innovations as a product of research advancements overtime in the field of deep learning, there will be more CNNs that would progressively improve. The aim of this literature review is to critically evaluate and analyse the imperfections of similar and previous works, to be able to rationalize and refine our proposed CNN methodology. It would then be deployed as a web application as a concept of computer aided design (CAD) to be used by health professionals to assist in classifying potential patients from MDD.

EEG and fMRI data are image data that may be required to distinguish patients that are suffering from MDD but such data are only understandable by health professionals with extensive mastery in the medical field. Non-experts within the health and medical field may have trouble grasping useful data from the image data. To transform such image data into charts or graphs, it requires many years of substantial training and exposure to the data. Although with utilization of modern machine learning algorithms, experts are still required to oversee crafting of important features from the data to ensure that the machine learning algorithm can optimally predict or classify the extracted features.

The advancement and evolution to deep neural network architectures such as AutoEncoders (AE) and Generative Adversarial Network(GAN), automatic feature extractions are done automatically without requiring supervision, specialist knowledge and involvements of human. In context of CNN, the convolutional layer has the ability to cleverly determine the spatial features of an image automatically and learn its features. Hence, the barrier into the domain of neuropsychiatric is diminished with experts in the deep learning field requiring minimum knowledge on neuropsychiatric to collaborate with health professionals and contribute to researches in the related field.

Finally, literature contents within this section will discuss on the depth of fMRI as it is the main type of data that would be utilized within our project. Furthermore, some research papers included discusses on the studies done in classifying psychological orders through both deep learning techniques and machine learning. Specifically within the deep learning segment, we would talk over researches that were previously conducted on different types of neuropsychiatric disorders such as AD and SZ which makes use of fMRI data for classification of patients. We will then conclude on the research done with related works to our final project and provide a summary of the main items discussed within our literature review.

2.1.1 fMRI Data

An American physician known as Raymond Vahan Damadian who is a medical practitioner is known for his invention of an imaging method known as field-focused nuclear magnetic resonance (NMR) where he collaborated with Andrew Maudsley in 1977 to further refine the imaging method to produce the very first image of a human body part, a finger (Elster, 2019). In today's world, NMR has been refined and evolved to what is now known as magnetic resonance imaging (MRI). With the established MRI technology, fMRI is a class of imaging method that is built upon it where powerful magnetic fields are induced onto the human body to produce regional time-varying differences within the brain's metabolism levels. With constant simulation of activity to the brain, the rate of firing of neural synapses and other similar signalling processes increases which give rise to pile up of waste materials as oxygen is depleted and the process of glycolysis carried out. Sequentially, this creates a region for increased blood flow to the affected area to regulate the internal state of human body through homeostasis. MRI is able to identify the differences in levels of oxygenation due to the activity of red blood cells in blood flow, hence Blood Oxygen Level Dependent or also known as BOLD was considered a better measuring method in contrast to to Cerebral Blood Flow (CBF) because CBF has disadvantages such as increased rate of acquisition in terms of results, lower sensitivity and it is delicate to human activities such as movement when compared between the aforementioned methods (Glover G. H., 2011). In recent times, Resting State fMRI (rs-fMRI) is another domain of interest that was researched which has overtaken the primary task of fMRI. This can be justified as it is much easier to extract rs-fMRIto task-fMRI since there are no requirements to develop particular tasks when a subject is undergoing an MRI scan. In relation to experiment trials, there is also a lower variance which permits an increase in cross-site collaboration in studies between institutions (Khosla m M., et al, 2019).

Prior to utilizing the raw fMRI data for further examination, there are some required steps such as pre-processing to be completed to make certain that the selected brain regions between each subjects are normalized and reduce artifacts extracted from the data, each pre-processing step may be merged or used as its own to adjust the fMRI data. A research study published in 2006, further explains the pipeline paths to assist an individual in pre-processing the fMRI data, the study did not include the most suitable method to refine the quality of the fMRI data as it is dependent on the conditions of the fMRI data (Strother, S. C. 2006). In a separated study published in 2008, it has stated that any general pre-processing steps produces a substantial impact on the connectivity estimates that is extracted from the functional MRI (Gavrilescu, M. et al., 2008). A further factor as to why fMRI shows a huge domain of interest within researchers because fMRI data are less complicated to be obtained when compared to

different methods of extracting neuroimages such as PET, it is not an invasive method, produces fairly large spatial definition, is lower in costs and is readily available in most academy (Glover G. H., 2011).

2.2 Literature summary

Deep Learning approach with relation to project

To counter the ever traditional machine learning approach, certain deep learning architecture especially CNN features unsupervised self learning through the automated feature extraction technology on hand crafted features that can be drawn out from multivariate fMRI data without the need of knowledge from a specialist. This inference is crucial in the medical related field as it allows extraction of neuroimages from fMRI or EEG data that can be remodelled as raw images to be trained on a CNN model as input where unique properties and features are used to produce precise predictions to any average clinicians. A paper that was published in 2016 during the International Conference on Digital Image Computing: Techniques and Applications (DICTA) reveals that the automated feature extraction and classification from a CNN towards a standard MNIST dataset is able to surpass an MLP which was pre-trained with hand crafted features with a difference of 99% by the CNN and 72% by MLP. Furthermore, in the other two tasks, the CNN model is also able to outperform the MLP model or produces very similar predictions.

In a research article published in 2020, a deep learning approach on differential diagnosis of patients between Alzheimer's disease (AD) and vascular dementia (VD) was carried out using MLP (Castellazzi et al., 2020). With similar neurological symptoms between AD and VD, it is common to cause misdiagnosis through conventional clinical checkups, hence a solution should be proposed to solve this problem. When machine learning methods are combined with MRI it has been proven that it is able to enhance the classification accuracy of such neurodegenerative diseases including AD and VD. The researched was conducted on three different machine learning algorithm, artificial neural network (ANN), support vector machine (SVM) and adaptive neuro-fuzzy inference system (ANFIS). However, the focus is on the performance of ANN in this research as the model used is MLP. With a test of 33 AD and 27 VD patients, ANFIS was able to achieve the highest prediction rate of 82.75% while MLP achieved 58.25% from data set of rs-fMRI derived using graph theory metrics. Even with optimization methods used within the research, this literature proves that MLP may not be the most optimum model in classifying neuroimaging data extracted from fMRI.

A BioMedicine journal that was published by The Lancer in 2019 features discriminating patients suffering from Schizophrenia by using a multi-scale recurrent neural network (RNN) applied on a dynamic time sequence of multi-site fMRI data (Yan et al., 2019). The research discusses that the traditional fMRI based classification mainly focuses on using spatial maps and functional connectivity as input. Instead of the traditional approach, this study is pivoted around investigating the dynamic time courses directly but temporal information is ignored. With a large data set of 558 Schizophrenia patients and 542 healthy controls, time courses of fMRI independent components (ICs) are extracted directly. Two methodologies were carried out during the study such as the leave-one-feature-out and multi-site pooling was tackled to identify the biomarker feature that largely contributes to classification of Schizophrenia. The classification accuracy for the simple CNN-GRU model on Schizophrenia patients was 80.8% while the multi-scale CNN-LSTM had 81.6%. The researchers were able to conclude that the leave-one-feature-out method was the key to the identification of Schizophrenia biomarker and the time courses from two brain components, cerebellum and dorsal striatum plays a large role in causing Schizophrenia.

There is another similar research to the journal published by The Lancer in 2019 on brain connectivity networks that focuses on analyzing on fMRI images that was using CNN-RNN model, however this research paper discusses on using CNN model to identify biomarkers on neuropsychiatric disorders such as Schizophrenia (Shahriman et al., 2020). The two methods that are focused within this paper are using 16 frequency band channels of electroencephalography (EEG) to record impairments between low and high frequencies within the cortical network of schizophrenic patients and fMRI that has the ability to identify schizophrenic patients with functional impairments in the frontal and sensory brain regions. The total data set of size 84 consists of 45 schizophrenic patients and 39 health controls. The performance of CNN was able to record a high prediction of 86.9% by using a cross validation method

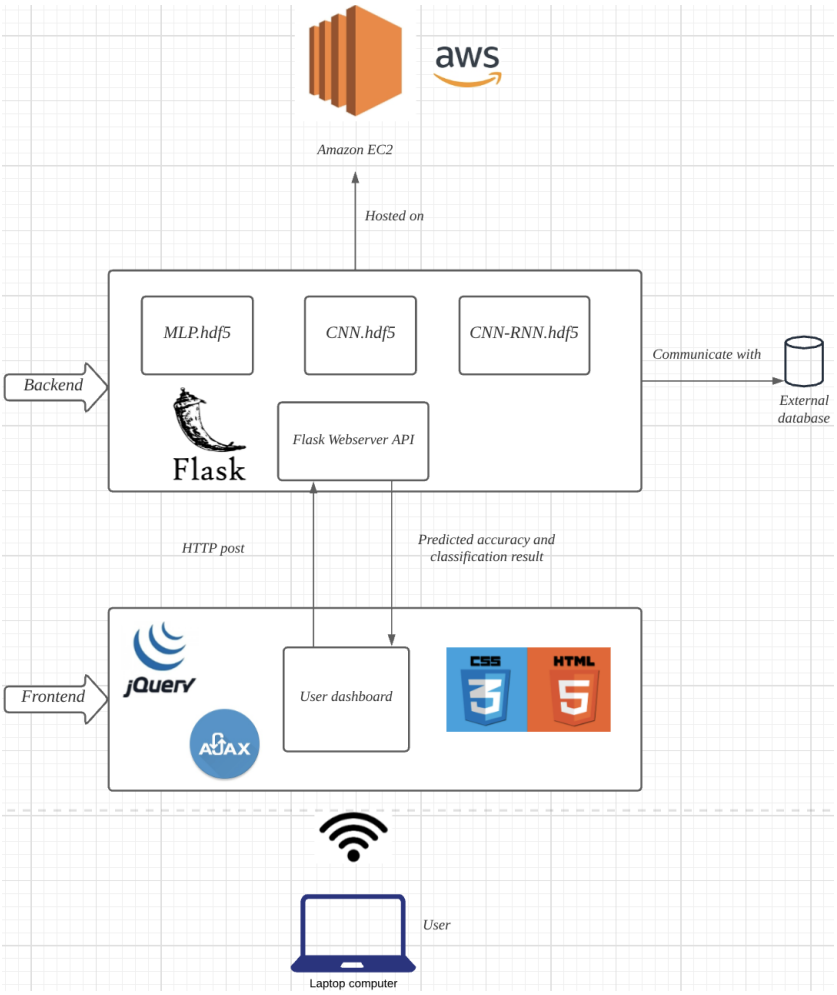

Figure 1: Overall system architecture

known as k-fold that is set to five-folds. Hence, it is justified that the CNN architecture has the ability to classify patients suffering from schizophrenia disorder.

With the few literature reviews that is related to our project from the section above, we are able to conclude the direction of our system architecture and that the researched deep learning models are able to assist in automatic identification of neuropsychiatric disorders from using data of brain networks.

3 Methodology

The main aim for this project was to build a robust web based classification model to aid medical professionals in their diagnosis of patients suffering from MDD. To achieve this we will be testing and implementing several different Deep Learning architectures in order to identify solid models that will be used for the classification task.

3.1 System Architecture

The overall system architecture and the workflow of our system is shown in Figure 1. In particular there are 3 major aspects of our overall system architecture. These are:

- **Front End**: A web application will be built using HTML, CSS,, jQuery and Ajax. The design of the user interface is dictated by HTML and CSS while the interactivity of the user interface is dependent on the Javascript library called jQuery. Since our web application will be talking to the

backend server, Javascript Ajax will be used to send and retrieve data in the background without reloading the page.

- **Back end**: Once the back end server receives an Ajax request from the front end, it will identify the type of request being sent in and perform classification based on the respective machine learning model. For example, if the type of request sent by front end is "mlp" and the patient id 0, the server will then communicate with the external database and retrieve the data of patient 0 and perform prediction on it. Then the server will return the outcome to the front end part so that it can be displayed to the user.
- **Machine Learning Models**: Many months of research has been conducted by us in order to build different machine leaning models, we then performed hyper-parameter tuning on all models in order to optimise their performance, which are then evaluated based on their testing and validation accuracy. We will then deploy these completed models for use in the final classification task. Each model and the process used will be described below in section 3.3.

3.2 Data set and Preprocessing

For our project we were given a dataset by Professor Hakmook Kang from Vanderbilt University School of Medicine Department of Biostatics, which contains 25 controls and 20 positive samples.

The original paper of the dataset states that participants were scanned on a Siemens 3.0T Trio Tim scanner, with an 8 channel head coil, with some specific MR sequences. The details of the sequences can be found in the original paper. After the scanning is done, images from the scan was pre-processed using the Conn toolbox (version 15.g) in SPM 12. The paper also mentions that there is no significant difference in age, sex or education between the MDD group of participants and healthy controls, however, there is a huge difference in Montgomery–Asberg Depression Rating Scale (MADRS) and Beck Depression Inventory (BDI) scores between those two groups (Albert, 2019).

Therefore in this paper, when classifying the covariance matrix of MDD patients and the health controls, no additional assumptions are made on the samples apart from those that are mentioned above which are in the original paper. The dataset is then split into two separate distinctive parts, both which we used different Deep Learning models and techniques for classification.

3.2.1 Static Covariance Matrix

The first dataset is the static functional connectivity data, which is a Pearson correlation matrix of the original fMRI time series (for single subject the time series dimension is ROIxT = 116x150). The connectivity matrix "data " dimension is NxROIxROI = 45x116x116, where N is number of subjects mixing healthy and major depressive disorder patients. ROI is a region of interest or brain cluster which is represented by a single time series in fMRI data. Since this data already comes preprocessed we do not need to any additional signal preprocessing on the data, only slightly modify the shape of the matrices for input into our models.

3.2.2 Dynamic Covariance Matrix

The second dataset was the raw time-series fmri data, which was given in a MATLAB file containing two variables:

1. fmridata" of shape (116,150,45) →(nodes, time points, subjects).
2. "labels" of shape (45,1) (subjects , 1), the data labels 1: MDD, 0: Healthy control.

We could then process this time series data for our purposes, which was to generate a dynamic functional connectivity data, which also a Pearson correlation matrix. In order to calculate our dynamic matrix, we first created a symmetrical 116 x 116 matrix (i.e matrix[i][j] == matrix[j][i]) to hold all values of a single patient's nodes.

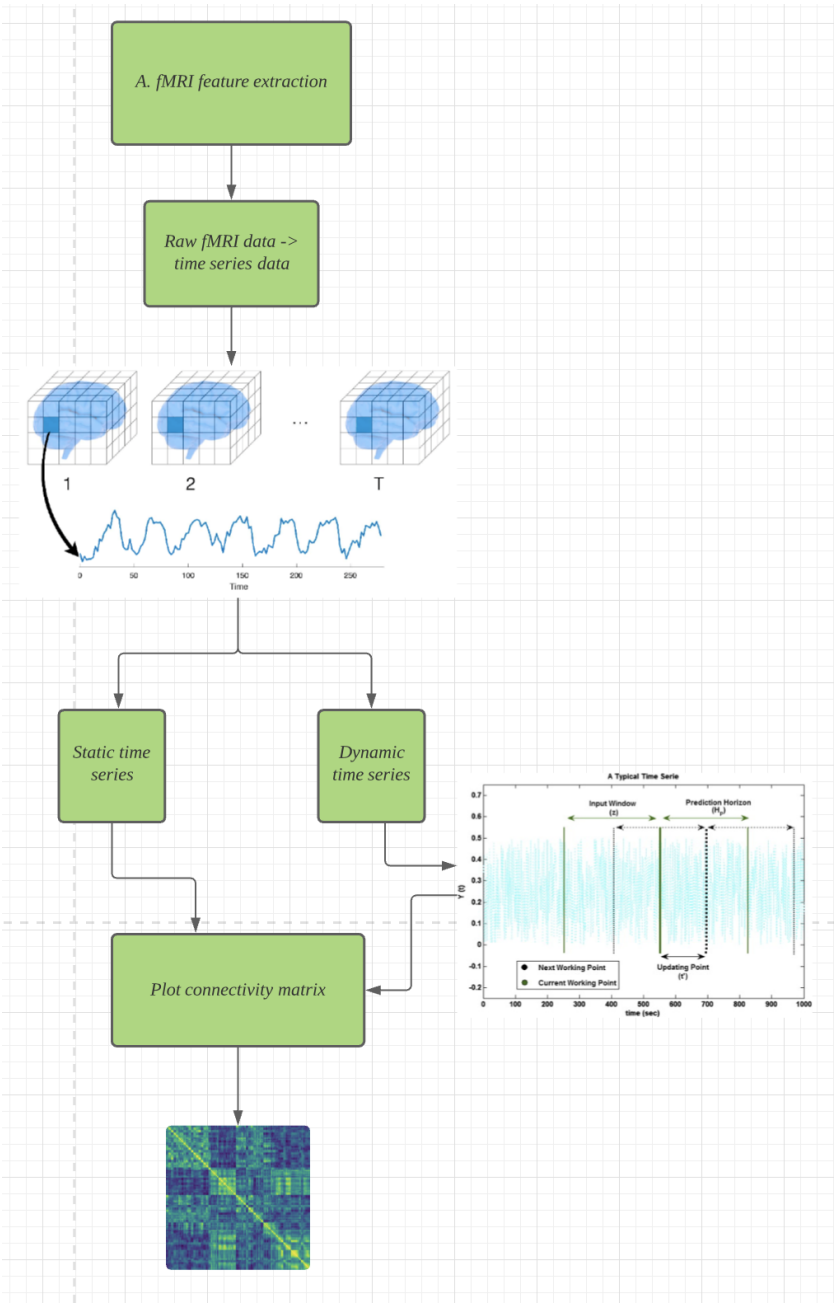

Figure 2: Data preprocessing steps. Sidén, Per. (2020) Mozaffari et al (2014)

We then took a a number of observations from our data, in our case we took 50 time points of data in order to calculate our co variance and stored it at each element of the matrix, so that each element in our 116x116 matrix now holds an array of 50 time point data. We then iterate through each element in the matrix and use the pearsonR function from scipy.stats to calculate each element's co variance with all other elements and will write the results to a csv file. The stride of the window is increased by 5 and we take another 50 observations and calculate the co variance, this creates overlapping in the signals being looked at and artificially creates more data points for us to look at. We then repeat this process until we reach the end of the data points available for patient 1 then we continue on to patient 2 and so on.

The pearsonR function used calculates the Pearson correlation coeffient between the different data sets, and it is calculated using the formula:

$$r = \frac{\sum(x_i - \hat{x})(y_i - \hat{y})}{\sqrt{\sum(x_i - \hat{x})^2(y_i - \hat{y})^2}}$$

Where x_i is the array containing the 50 data points of one ROI and y_i is the other 50 data points of the other ROI being compared.

3.2.3 Loading the Matrices

Before we can fit the data into our models we need to read and preprocess some of the values. For both the dynamic and static Covariance matrices, we load the given MATLAB file as well as the labels using scipy.io functions. We then have to shrink the size of our labels to on dimensional by flattening it.

For our MLP model, since a co variance matrix it a reflection of itself at the y = -x line, we can just only look at half of our data without losing any information and making training much faster and with less noise.

For our CNN model, since the initial data is of dimensions 45 x 116 x 116, we need to expand the dimensions of the input data by 1 in order to have a 'channel' to input into our CNN, we expanded the dims by 1 in order to have a Black and white image represented (45 x 116 x 116 x 1), although you could expand the last channel by 3 for a RGB image (45 x 116 x 116 x 3).

Finally for our CNN-LSTM we first have to read our generated dynamic connectivity matrix where our data ends up as a matrix of dimensions 45 x 20 x 116 x 116, corresponding to the Patient ID, Observation Number, ROI, ROI.

3.3 Deep Learning Models Used

After thorough literature research, referencing work done by Gao et al, (2018), we found that mostly all work done on MDD classification using deep learning or machine learning architectures relied on a SVM structure, or when a CNN was used (Ke H. et al, 2020) it was used on EEG signals rather than fmri. Hence we decided to implement 3 different models to both create the most robust model that we can, as well as explore how accurate other architectures are. All of our models were built using the Keras library and its API and all the source code for our models can be found attached in the appendix.

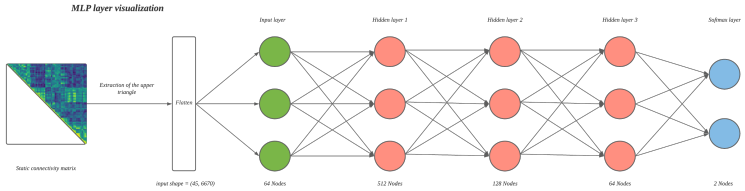

Figure 3: MLP Architecture

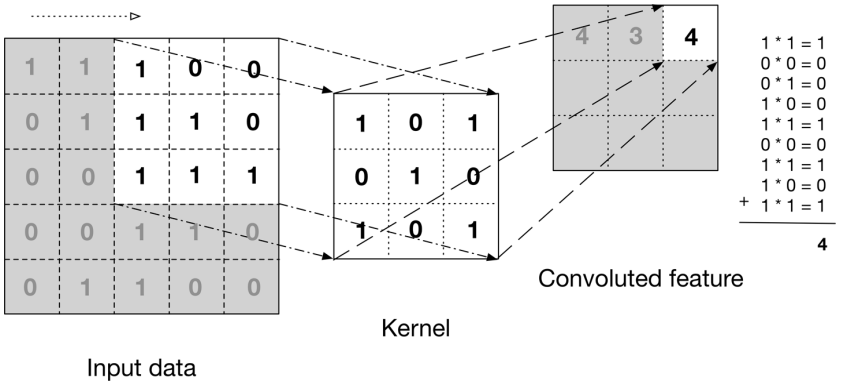

Figure 4: Convolution operation (Manav M., 2021)

3.3.1 Multi-Layer Perceptron

An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Each node is a neuron that uses a nonlinear activation function, such as tanh or relu, these are mathematical functions that transforms the input data in different ways and are used for different purposes. An MLP is a supervised learning feed-forward neural network, where during the forward pass, the inputs are fed to each neuron in order to classify the inputs. During backpropagation the error between the classified and actual labels are minimised and the weights of the previous layers are updated in order to get a better classification in future passes.

Our model is a 3 hidden layer MLP, with a BatchNormalization and Dropout layer between each hidden layer, we have built the model according the the highest validation accuracy done during parameter tuning, with our model having the shape of the one in Figure 3.

3.3.2 Convolutional Neural Network

A CNN is a class of deep Neural Networks that are most commonly used for visual analysis of images, this is due to the convolution function performed by the convolutional layer, hence the name CNN. This function is a mathematical function that passes a kernel/filter over the input data and we extract features by doing transformations of the input data with respect to the Kernel similarly shown in Figure 4, this is particularly useful in extracting Spacial features of the image.

The earlier layers of the model tend to extract more basic features such as horizontal or vertical lines, but the deeper layers start extracting more distinct features such as corners or edges and the final layers may extract features such as faces.

For our model we are using 3 CNN blocks chained one after another, with each block containing a BatchNormalization and pooling layer as well. The details of our model can be viewed at Figure 5.

The major advantage of a CNN is that the feature maps are automatically learned and extracted, hence a even non medically personnel can develop a CNN that can classify MDD images without specialist knowledge on what hand crafted features would need to be extracted using other traditional Deep Learning

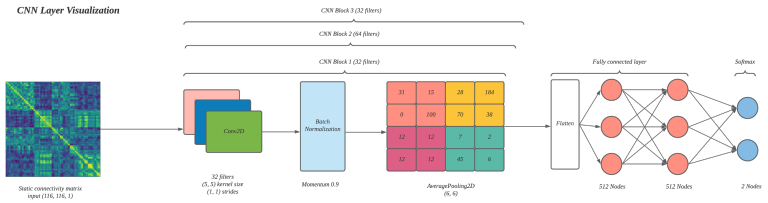

Figure 5: CNN Architecture

methods.

3.3.3 CNN-LSTM

A CNN-LSTM is a hybrid model that aims to combine both the advantages of a CNN and LSTM into one mode. The CNN layer is adept at mapping out spatial features and the LSTM is specialised for sequence inputs. Hence we thought a CNN-LSTM would be ideal to detect the changes in the Co variance Matrix of an MDD patient throughout a sequence of time, i.e the Dynamic Connectivity Matrix.

As we already discussed the CNN layer before, we shall briefly explain the LSTM layer. An LSTM is an extension of a RNN, which is a Recurrent Neural Network, where the output of one cell is passed along to itself as well for the next computation where the previous and current input is compared, this makes the RNN and by extension the LSTM useful for sequenced inputs. However the problems with an RNN is that if the input is long enough, the inputs no longer change the weights of the model in any meaningful way, essentially causing the model not to learn anything at all, this is the gradient vanishing problem.

The LSTM model has many different activation functions and 'gates' introduced in order to combat this problem. These gates decide what relevant information gets passed along the network, and these gates act as 'memory' in order to hold more longer term memory.

For our model we are using 3 CNN-LSTM blocks chained one after another, with each block containing a BatchNormalization and pooling layer as well. The details of our model can be viewed at Figure 7.

Hence the CNN-LSTM is a hybrid model in of itself and aims to take advantage of both CNN and LSTM architectures for classification. We chose to use it because the time-series fmri data measures values at the ROI in the brain, and each ROI is spatially connected to one another we take advantage of this with the CNN Layer,the final model architecture once again includes both Average-Pooling and Batch Normalization layers, the resons why will be explained later on.

3.4 Model Training

For all model training we implemented K-Fold cross validation, this is because it is useful for evaluating machines with limited data samples, in the case of our CNN and MLP, just 45 data samples. The data is split into the train, test and validation data sets and the model is trained on the training set and then checked and tuned to the validation set, then another group is changed to be the test set and another the validation, etc. until all groups have been trained on the model k - 1 times.

The general procedure is as follows:

1. Shuffle the dataset
2. Split the dataset into K groups
3. Split the groups into training, testing and/or validation sets
4. Fit the model on the training set, and evaluate on the test/validation set

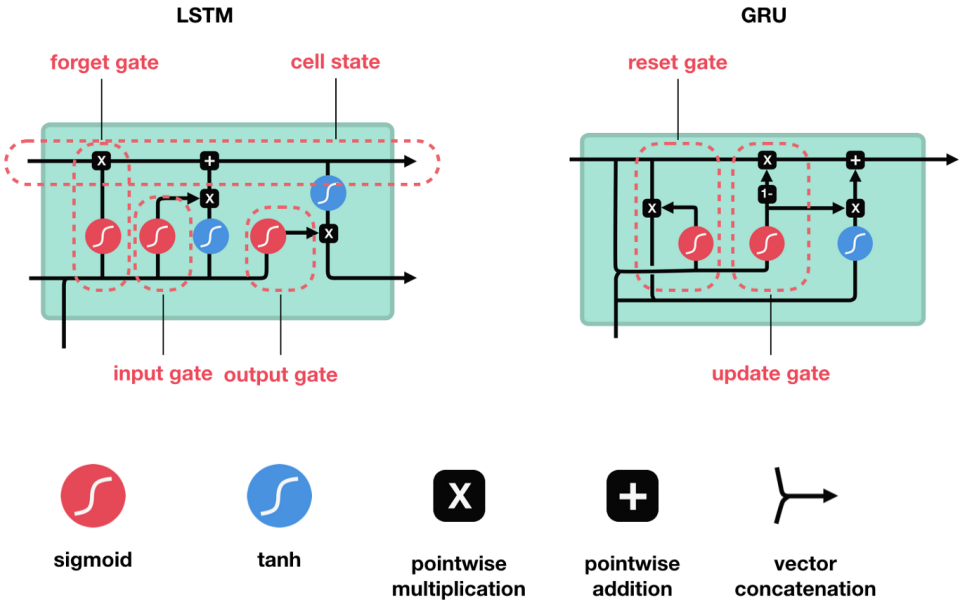

Figure 6: LSTM model (Phi M., 2018)

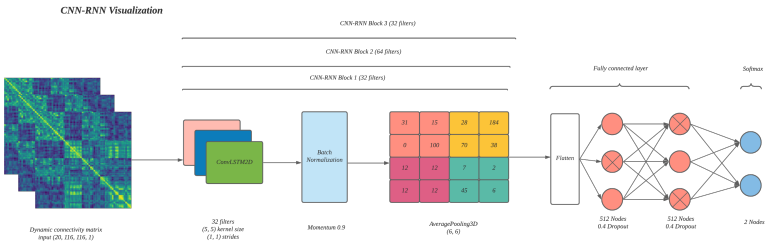

Figure 7: CNN-LSTM Architecture

5. Keep track of the evaluation score, if it is the best score depending on your metrics (val_accuracy, testing_accuracy) then save the model

6. Once the best model has been found, we will load the model and get our final accuracy.

The reason why we use K fold is so that we can look through our limited data samples multiple times as well as this method results in generally a less biased estimate of the model compared to other training methods.

3.5 Model Evaluation

Model evaluation is an important process to ensure that our models meet our predefined accuracy. Usually, this process is included in the model training by splitting part of the training dataset into validation dataset and we will take note of the validation accuracy. However most of the times model evaluation is done by testing our model on testing dataset . However, if the testing accuracy is low, we could add Batch Normalization, Dropout and Pooling Layers on our model architectures to achieve higher accuracy. After adding these layers, hyper parameter tuning is indispensable so that we would know how much of these layers should be added to the model.

3.5.1 Batch Normalization

We often utilised a BatchNormalization layer in all our models, this is as Ioffe, S., Szegedy, C. (2015, June) showed, adding one helped to standardise the inputs and help reducing internal covariate shift, allowing for larger learning rates to be used and hence our model is trained faster as a result, which is especially useful in our more resource expensive CNN-LSTM model.

3.5.2 Dropout

We also utilised many Dropout layers in our models, As Srivastava, N. et al (2014) researched this helps reduce the overfitting of our model, where the model picks up on the noise in the data and is trained so that it is a good classifier on the testing set but bad at new data. This works by causing some of the nodes to be ignored or 'left out' so that the weights are not updated and other nodes will have to generalize in order to compensate for the dropped node. All of this is done to make a more robust, general model that works well even on new and unseen data.

3.5.3 Pooling Layers

For CNN sometimes the feature maps can be really large especially when stacking Convolutional Layers, there are a lot of trainable parameters and training can take very long or even over fit the model. The way to reduce the dimensionality of these feature maps from a CNN layer is through a Pooling Layer, as an added benefit the down sampled feature maps are now more robust to changes in the input due to only the most prominent features being passed through the Pooling layer.

For our models we are doing Average Pooling, where we summarise the most average features from the image to be passed on to the next layer. Like how a CNN Layer passes a kernel through the image, the Pooling layer does as well, it passes a kernel through the feature maps extracted by the CNN layer and and down samples the feature maps. Pooling layers follows after a CNN Layer and an activation function has been passed through the outputs and the pool_size kernel can be chosen by the user.

3.5.4 Hyper Parameter Tuning

One of the most crucial element of a model is what hyper parameters to choose in order to train the model, this is in order get a more desirable model in terms of accuracy. For all our model tuning we used code provided our supervisor Dr Fuad, which we tuned to suit each model being evaluated, in order to automatically build, test and save how each parameter effected our validation accuracy.

For the optimization we used a library, scikit-optimize in order to use the function gp_minimize. This function performs Bayesian optimization using Gaussian Processes. Because Neural networks are often large and may take an extremely long time to train, optimising these hyper parameters would take a particularly long time. Hence, the function approximates the optimal parameters by following a Gaussian probabilistic model.

The function balances exploitation where the predicted value is close to the expected optimum and exploration, where the function is further away from the optimum. Due to being able to predict how optimum the parameters are, Bayesian optimization has been shown, in practice to obtain better results in fewer evaluations compared to other optimisation algorithms.

3.6 Implementation of Front End and Back End Architecture

Upon having the best model from our training, the best model would be deployed so that it can take in new images and infer on it. There will be a total of 3 best models, namely best_mlp_model, best_cnn_model and best_cnnlstm_model and these models would be deployed to a server of our web application so that it is ready to infer on new images.

The front end of our web application is built using HTML, CSS and jQuery, mainly by modifying the open source dashboard web application called DeskApp on Github (Ankit H., 2018). The user interface was modified professionally so that it looks like a proper medical portal used by hospitals. Some additional jQuery functions were also added to suit our requirements of showing the prediction results and also plotting the images of connectivity matrices. The reason why open source dashboard was adapted was because designing a dashboard from scratch would take plenty of our time as we are not very proficient in HTML and CSS designs. Besides, the open source dashboard is well maintained by the community and hence there will be minimum code logic errors to happen.

However, we implemented our back end from scratch because the functionalities that we needed are very specific and coding it from scratch would save us more time. The back end is implemented using a light weight Python framework called Flask and we are making assumption that our database is an external database. The reason being medical data of a patient is a very sensitive information and such web application is not supposed to have its own database to store the information.

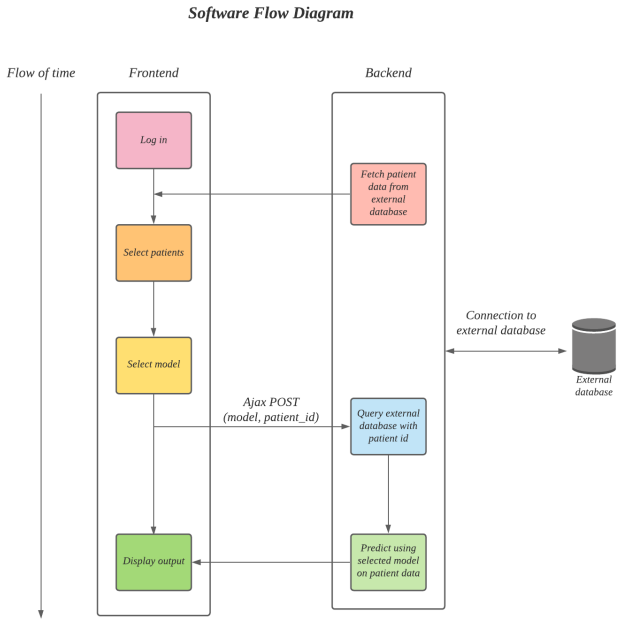

Figure 8: Software flow diagram

The flow of our software is as shown in the figure above. When the web application starts, the back end will fetch all patient information from the external database that belongs to a hospital and then send it to the front end to be displayed. The user will then be able to see a list of patients that are under their care. Once the doctor selects a patient and the type of deep learning model that they want to use to in order to perform prediction, an Ajax POST request is sent to the server together with the patient id. The server will then query the external database again for relevant data and then perform prediction on the data. The outcome of the prediction in terms of percentage of being a MDD patient is then sent back to the front end and be displayed to the doctor. The doctor will then make relevant judgement based on the predicted percentage.

Note that since this project has not collaborated with any hospitals, the relevant patient information will be stored in the Flask server instead of an entirely new database. No communication can be done between our server with the hospital's database thus far.

4 Project Management

4.1 Introduction to Conceivement of Project

In our previous semester, we were offered several project topics such as analysis of Deepfake, authentication with image processing, text processing with machine learning and deep learning classification. After researching the other available topics, we have decided to select this topic for our project as we felt that the topic was researching into a field of study that is important to society. We then reached out to the supervisor, A/Prof Ting and organised a meeting with to gain more understanding of the scope, objectives, requirements and deliverables of the project.

With the team reaching an agreement on the aforementioned topic, our supervisor introduced himself and shared some background information of himself as well as some of his previous researches that were completed throughout the years along with his partner Dr. Fuad. He told us that he is working on the automatic classification of MDD patients and controls using Deep Learning models and asked if we were interested to proceed our project with this project. After thorough discussion among ourselves

we decided to proceed with this topic as it would be extremely useful to the research in the medical field if we could produce some decent classifiers for MDD patients.

In the following meetings, he shared with us on how time series data can be obtained from fMRI images through brain voxels and how we could transform the time series data into a connectivity matrix. Furthermore, he provided us with some resources on the CNN deep learning model architecture entitled computer vision by Andrew Ng. We were also provided with several research papers such as how the brain network performs in neural networks and mental diseases. Finally, Dr. Fuad has provided us with functional connectivity matrices dataset of patients suffering from MDD and controls that is used throughout our project as we would need to develop a prototype for our project proposal.

With all the resources and information provided from our supervisor, we have decided to conduct an internal group meeting to discuss on our approach to tackle the project and development of a suitable prototype. We read up on the resources provided and watched the computer vision video which explains CNN architecture thoroughly. The team has agreed to produce a CNN model as the research paper was able to show fascinating results and we were provided with some base knowledge of the architecture.

4.2 Approach of Project Management

From our project proposal, we have decided to utilize the waterfall model approach throughout the development phase of the project because the phases of waterfall model such as requirements, design and implementation of the project were clear to us. The requirements of the project was to develop a deep learning model which is able to classify between patients with neuropsychiatric disorders from the control patients within an acceptable accuracy. In the design phase, we would develop several deep learning models from as simple as a multi layered perceptron (MLP) model to a complex hybrid convolutional neural network-recurrent neural network (CNN-RNN) model and decide on the best model along with justifications.

However during the implementation phase, we had encountered an issue that was highly expected from the proposed risk register that conflicted with the waterfall model. The issue was that the performance of the deep learning models were not acceptable. We required more weekly meetings with our supervisor to discuss on how changes could be made as an attempt to improve its performance. Hence, we had to switch to an Agile development methodology and prioritize on conducting weekly sprint to discuss on issues and solutions with relation to project development. During those meetings, we would update our supervisor regarding our deep learning model improvement based on our previous week's meeting. In retrospect, we adapted to Agile methodology fairly quickly and benefited immensely from it.

Name	**Role**	**Description**
Dr. Ting Chee Ming	Product owner/Supervisor	Provide advice and suggests on improvements to be made.
Fong Zhong Kern	Scrum master	Ensure that the project timeline and progress are going well. Verifying with team to ensure that milestones are achieved.
Choo Jun Yi, Khoo Teng Ian	Team developers	Assure consistent quality development of the products.

Table 1: Roles in Agile Methodology

4.3 Project Planning and Execution

With our roles assigned in the first week, we proceeded to construct our work breakdown structure (WBS) which will be included into the appendix. The WBS will be constantly updated should there be any drastic changes during the development phase. Reconstruction of our current WBS with reference to the previous WBS and project timeline that was proposed from the previous semester was required. This was mainly due to drastic changes in the proposed product of the project and project methodology used.

After WBS is created and validated by our team, our project manager modified the Gantt chart from our project proposal to fit the updated project specification. A copy of the Gantt chart is added into the appendix section. Our Gantt chart depicts the updated tasks to be completed throughout the span of the project that will be handled by all members of the team. The Gantt chart is essential to the team as it creates a rough draft of planning and scheduling of all the tasks throughout the project. It was able to assist us in identifying and focusing on the core aspects of our project. To ensure our project satisfies the requirements and coverage of all our users that are mainly doctors and patients, we have included a requirement traceability matrix (RTM) that can be found in the appendix. The RTM gives a rough idea of our user requirements so that our tech developers would create tests that satisfies the user's needs.

With the ongoing Covid-19 pandemic our communication medium is heavily impacted as we are no longer able to meet physically. This has made it more difficult for the team to provide immediate assistance as our relay of information will now be done virtually. Besides that, we are also unable to provide real time code reviews and give feedback on improvements and optimizations. Code reviews are important as we are traversing into the deep learning field which we know less about. As such, the team has decided to take this chance to work independently, professionally and proficiently. Multiple pre-emptive measures had been carried out to reduce the issues that were previously raised.

To setup our foundation for our communication mediums, we relied on familiar and dependable tools such as Whatsapp, Zoom and Discord. Our medium of file sharing would be through Google Drive, as such a shared drive had been created. This allows quick and easy access to research papers, pre-processed data and our code implementations. Through Whatsapp group chat and Zoom, we are able to organize our formal weekly meeting with our supervisor as well as keep ourselves updated with pending and completed works by our team members. Discord was chosen as our daily informal discussion as it allows us to immediately host a voice chat quickly within our team while Zoom is mainly used for formal meetings with our supervisor where we are able to schedule a meeting ahead of time.

To understand our strengths and drawbacks with respect to technical abilities and project management skills, we performed a SWOT analysis among ourselves to minimize obstructions throughout the project. The summarize result are as such:

1. Strengths:
    - There are no issues for the team members to work collectively and cohesively because the team members have worked in a team in other projects
    - The team members are experienced in programming in the required programming language (Python3) and external projects regarding Deep Learning have been done by the team before.
    - Team members are familiar with Waterfall model and Agile environment. They are also able to work in both environments.

2. Weakness:
    - Restricted and limited project duration.
    - Insufficient computational resources for development of complex deep learning model.

3. Opportunities:
    - Supervisor and his partner are well versed in the field of deep learning which may assist the team.

- The area of research is still very new in terms of integrating it with Deep Learning models
- Ample research materials of the topic are available on the Internet

4. Threats:
    - Unable to complete the project in time.
    - Final software product may be of lower quality than proposed.

After we had identified and understood our SWOT analysis, we planned our breakdown of tasks with respect to our team members strengths and determine the major milestones of our project. The milestones are as follows:

1. Production of static and dynamic fMRI time series data: The static and dynamic time series data is required to produce connectivity matrix that would be used as an input for our deep learning models.
2. Deep learning model: Develop and train multiple deep learning models such as MLP, CNN, CNN-RNN that can classify the input data with acceptable accuracy.
3. Front-end user interface: A dashboard interface system that is able to receive input data to be passed to the back-end framework for classification.
4. Back-end framework: A back-end framework to host all our deep learning models.
5. Product testing: A well and comprehensive overall system test on all developed deep learning models hosted on the back-end framework and front-end user interface dashboard.

The breakdown tasks with respect to our team members strengths are as follows:

Tasks	**Type**	**Completed by**
Production of static and dynamic time series data from fMRI	Data pre-processing	Khoo Teng Ian
Development of deep learning model	Programming	Choo Jun Yi, Khoo Teng Ian, Fong Zhong Kern
Front-end user interface	Development	Choo Jun Yi
Back-end framework	Development	Choo Jun Yi
Project reports	Project management	Fong Zhong Kern
Final product testing	Testing	Fong Zhong Kern, Khoo Teng Ian, Choo Jun Yi

Table 2: Tasks Breakdown

4.4 Project Resources

To execute the project, there are some basic computation resources that are required. The software and hardware requirements can be referred to Table 3.

Hardware Requirements	**Decision**	**Function**
Operating system (OS)	Microsoft Windows 10 and MacOS	1 of our team members used Windows 10 while the other members used MacOS. All of our software are supported by the OS.
Processor (CPU)	Intel Core I5	All of our members uses Intel Core i5 and has averaged similar performance to quad core Intel Core i5 6600k.
Memory (RAM)	8GB and 16GB	A minimum of 4GB was required to train our CNN-RNN model as it contained a high number of trainable parameters.
Storage (HDD/SSD)	128GB	A base minimum of storage is required to store our codes of our deep learning models.

Table 3: Hardware Requirements

The following table discusses on the software libraries that are used throughout the project. Refer to Table 4.

Software Libraries	**Function**
Keras library	The open source Keras library that is developed on Tensorflow backend. It allows the team to build all the deep learning models throughout the project.
Numpy	Numpy library is used to visualize the shapes of our input data to ensure that the input shape fits all our deep learning models appropriately. It also allows transformation of multi-dimensional arrays (ndarray) for the vectors of connectivity matrix from the fMRI data.
Scikit-learn	Scikit learn allows us to use K-fold cross validation technique on all our models split our data instead of assigning a random splitting.
Scipy.io	This library allows us to read data from matlab files which mainly stores fMRI connectivity matrices.
os	os provides the functionality to create folders and store our best model used during training combined with early stopping technique.

Table 4: Software Libraries used

Finally, Table 5 discusses the software used and its justification.

Software Requirements	Justification
Spyder	We initially used Spyder to train our MLP and CNN models but was quickly dropped when developing CNN-RNN model as it requires heavy computational resources.
Microsoft office	Words was mainly used to document all our weekly meeting minutes with our supervisor and Dr. Fuad.
Visual Studio Code (VSS)	Teng Ian used VSS when he was required to pre-process fMRI datato produce static and dynamic connectivity matrix.
Google Chrome or Mozilla Firefox	A suitable web browser that supports Google Colab where we performed hyperparameter tuning and trained our CNN-RNN model as it requires heavy computational resources.
Ubuntu LTS	It acts as a machine image of our EC2 instance to host our models.
Zoom, Discord and Whatsapp	Our virtual communication mediums.

Table 5: Software requirements

4.5 Risk Management

Included in the appendix is the updated status of risk register from our project proposal. Refer to table 6 for our risk assessment.

4.6 Limitations Encountered during Project Management

In the following segment, we will discuss a compilation of limitations that we experienced throughout the course of the project from a project management standpoint.

1. Opinion conflicts: As we were developing several deep learning models for the project, we had some opinion conflicts. Some of which includes are the deciding on the best type of deep learning model to be used for our submission. There were also conflict on methods to improve the models to achieve the highest accuracy as we had limited time for development.

2. Time constraints: Since the team members are final year students, most of the units enrolled are challenging and have heavy workloads. Managing our time was challenging because of the constant workload from different units with similar deadlines. There were some weeks where we had to halt our progression which affected the scheduled proposed timeline. To ensure that we do not delay the project even further, we sacrificed some personal time to hasten project progress immediately after the other assignments were completed to meet the planned deadlines of tasks.

3. Communication: This posed a significant challenge to us as the entire project period is done virtually due to the Covid-19 pandemic. From team members enrolling in different units with time clashes to team members having personal issues. It was certainly difficult to arrange a suitable time for weekly sprints through Zoom and Discord. Besides the weekly sprints and the closure of campus, this meant that we are unable to have physical communications which leaves us with Whatsapp as our only medium of communication when we had issues or discussions with our allocated tasks.

Risk	Description	Solution
Lack of computational power due to hardware limitations.	As we are required to explore the different architectures of models, some models have a high number of parameters which may require heavy resources to train. The majority of our team members are using Macbook which are not equipped with an external graphical processing unit (GPU) and the availability of high random access memory (RAM).	To solve the problem, we have limited the maximum number of parameters that we can train on some models and we used cloud computing via Google Colab with GPU hardware acceleration to train complex models with hyperparameter tuning technique.
Delays in starting the project.	We were unable to work on the project during our summer break one of our member had enrolled in industry based learning (IBL). During the beginning of the semester, we had also delayed the commencement of the project as we had trouble prioritizing our tasks.	We consulted our supervisor to assist us with the ordering of importance of tasks which led us to prioritize and commence our project development.
Physical meetings and discussion.	With the fluctuations in Covid-19 cases throughout the year 2021 in Malaysia, we were unable to conduct any physical meetings and discussions between our team members and supervisor.	We utilized Zoom for formal meeting and discussions with our supervisor while Discord is used for informal internal discussions between ourselves. We would also use Whatsapp to organize our weekly Zoom meeting links.

Table 6: Risk assessment

5 Outcomes

5.1 Results Achieved

It is important to understand the research questions and hypothesis of our project as it acts as a foundation for our research and the direction in which we will progress to meet our outcomes. At the commencement of the project, we discussed what were the objectives of the project and this is what has influenced and justified our system architecture.

- **Research Question 1: Is it feasible to train a Deep Learning model by providing input data of connectivity matrices between MDD and control patients that can classify them?**

  Outcome: Achieved

  Justification:
  Our initial proposal was for us to design a CNN model and to train it with the provided matrix data. However, as we continued working and tweaking the model layers, our average accuracy stayed at around 33%.
  That was before we had done any parameter tuning as described in the methodology and after some more research done on the CNN architecture, as well as testing and deploying a MLP and CNN-LSTM architecture, we have finally achieved results of around 53 to 60 % accuracy.

- **Research Question 2: Is it possible to extract dynamic time series data from fMRI of MDD patients that would then be converted into connectivity matrix. If so, would you be able to classify all these new data points?**

  Outcome: Achieved

  Justification:
  At the commencement of our project we were given the static connectivity matrix which was already

preprocessed. As there was only 45 labeled data points, we considered that that this was moderately scarce and hence we were given a data set containing the raw time-series fmri signals. Hence we had to extract and process this data to form our dynamic connectivity matrices as described in the methodology. We were able to build and deploy a hybrid CNN-LSTM architecture that was able to work on this newly generated data set.

- **Research Question 3: If it is possible to extract dynamic time series data, what type of deep learning architecture would we be suitable for this operation?**

  Outcome: Achieved

  Justification:
  Our supervisor suggested to research on a deep learning architecture that is a combination of CNN and RNN. Thus, we stumbled upon a new deep learning architecture known as the convolutional LSTM 2D layer (convlstm2d). It is typically used for input data such as video calls, security camera footages, satellite imaging and films. As such, we were unsure if it has the ability to classify MDD data. After further research and building and tweaking the model, this model is indeed able to classify our input data to a certain extent.
  but there are drawbacks to the design. Since the input data is dynamic and the input of convlstm2d layer is a 5D float tensor, the number of parameters increases across each layers. Hence, it is tremendously taxing to train the model on high epochs with regard to computational resources on Google Colab and our local devices.

- **Research Question 4: With all the deep learning models developed throughout the project period, how are the performance of each model and which has the best capability of classifying the patients?**

  Outcome: Achieved

  Justification:
  We were able to develop three separate models, each using different Deep Learning techniques and architectures in order to classify our data sets, we also was able to compare the performance of each model using K-Fold cross validation and calculating the average accuracy performance of each model after they had been finished training.

5.2 How Requirements are Met?

Table 7 and 8 summarises the achievements and reasoning of all identified functional and non-functional requirements for our project .

5.2.1 Functional requirements

5.2.2 Non-Functional requirements

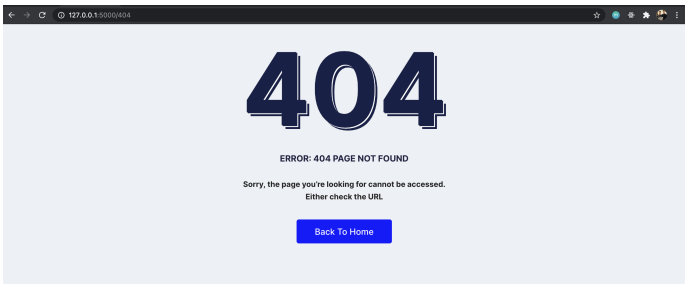

Figure 9: Barred action image

Requirement	**Achieved**	**Reasoning**
The deep learning model can automatically identify whether the data is a normal patient or patient with neuropsychiatric disorder with reasonable accuracy.	Somewhat Achieved	Compared to our initial estimations based on prior research, we did not achieve an adequate accuracy of our models, however research on fmri data using our selected models has not been deeply researched into before. Our supervisor, who is working on a similar project to ours also received a slightly improved accuracy when compared to our models.
The front-end of web application allows upload of an image to be classified at a time.	Not Achieved	We have changed this requirement from when we first conceived this requirement, that is because we realised that there are some considerable security concerns allowing a user to upload data to a secure and sensitive database. As well as the fact that our end user would not be in charge of handling the data input
The back-end of web application will preprocess input image to fit the input requirements of deep learning model.	Not Achieved	As stated in the above requirement we no longer allow an end user to upload data.
The web application must be able to serve multiple users at a time.	Achieved	As we are hosting our back end on an EC2 instance, this allows multiple users to connect to the web application if they have the link.
The software is able to store and retrieve information of a registered user through Firebase.	Not Achieved	We have changed this requirement as well, as Firebase will not be as secure as a hospital's dedicated database and due to the sensitivity of our data. Due to both security and ease of implementation for the hospital, we assume that the information is no longer stored by us.

Table 7: Functional Requirements

Requirement	**Achieved**	**Reasoning**
The web application is user friendly and is easy to operate and master.	Achieved	We believe that we have created a web portal that is easy to navigate, can be integrated with the hospital's data base to give users useful and relevant information, a snapshot of our web application can be seen in the Software Deliverables below.
The deep learning model should return results within a reasonable amount of time after it is sent to be processed.	Achieved	As the system is integrated with the Hospital's database, when the user wants to get the results of a classification, there is no sending over of large amounts of data from the front end to the back end, the model will be able to quickly preprocess and read the data from the database and return a quick result back.
The connection to the database and the back-end server must be secured.	Somewhat Achieved	As the software itself is to be implemented with the hospital's network and a secure login system is to be added, the connection to the database is controlled. The security for the back-end server is using a http connection and hence not as secure as it could be.
Error messages will be shown to the user when an action is prohibited.	Achieved	When an error occurs such as the user accessing barred function an error message like the one depicted in Figure 6 is shown.

Table 8: Non functional requirements

5.3 Design Justification

In this section we will be discussing our decisions made in our methodology and in the final design of the Deep Learning architecture.

5.3.1 Chosen Models

When the project was first initiated by us, we conceptualised a CNN model being a good fit for this classification problem. The rationale being that the fmri data is a signal depicting the brain activity in certain Regions of Interest (ROI), as these ROIs are located on different parts of the brain, how interconnected each ROI with each other is an important factor to consider. Hence a CNN which is competent in analysing spatial images would help in classifying Connectivity matrices which in of itself a spatial representation of the signal strengths in the ROI of a patient. In our initial findings, we achieved a sub par result. Hence we sought to use different model architectures to help supplement our accuracy. We chose a MLP because of the simplicity of implementation of the model, as well as a CNN-LSTM model to work on our time series data set.

5.3.2 Data Set Splitting

For both of our data sets we used K-Fold cross validation, we specifically chose to this implementation for the models using the static connectivity matrix, because as described in our methodology the lack of data points may cause our model to over fit and K-Fold is to help mitigate that. We also continued to use K-Fold for our CNN-LSTM despite having substantially more data points by reason of there being less bias compared to other training methods. During hyper parameter tuning and the training of our final model, we used a 5 K-Fold cross validation tactic. This is mainly due to the limitations of the size of our static correlation matrix, and any higher folds would result in our testing set only having a limited number of data. This means that around 9 of our 45 data points are kept aside for the testing of our model which is a reasonable split in our data in order to be able to effectively train the model without overfitting and still achieve competent accuracy.

5.3.3 Chosen Hyper parameters

We carried out hyper parameter tuning on Google Colab, as it was quite resource intensive for our local machine. The main metric that we used to optimise our models was the accuracy on the validation set. What resulted was a csv file that contained the results of the tuning run using sci-kit optimise. Figure 14 showed the top results of the tuning on our MLP model, we tested many aspects of the model to see what parameters had the best accuracy. We managed to get a 87.5% validation accuracy on one of the results and this was what we took as the best model and hence, built our MLP model on these metrics.

The top performing results in terms of validation accuracy can be seen at Figure 19 for the MLP model. Figure 20 for the CNN model and Figure 21 for the CNN-LSTM model.

For the MLP the most interesting results to note are that practically all the best models had a high number of fully connected layers as well as a high node count per layer. Surprisingly the most popular relu activation function does not perform well in most of the models, with a combination of tanh and sigmoid functions having higher accuracy comparatively.

For the CNN once again all the top results used a learning rate of 0.01, with all models using 3 Convolutional Layers and 2 Fully connected Layers, the kernel_size varied between all models, with the best kernel for each layer to be [4,3,3]. The other noteworthy results were that all models used a pool size of (6,6) for the pooling layer and the Dense nodes having 412 and 512 nodes.

For the CNN_LSTM model, we can see a similarity in the number of layers of our architecture, with having three Conv_Lstm layers and 2 fully connected layers being part of all optimised models.The best learning rate was also found to be around 0.01 and the models with larger pooling size seemed to perform better as well.
During the hyperparameter tuning, we found there were some limitations that we will be discussing in the later on section below.

5.4 Results

In this section, we will be plotting the final accuracies that we achieved from all our models.

5.4.1 Accuracy

Model	Training set	Testing set	Validation set
MLP	79.44%	60.00%	87.50%
CNN	100.00%	57.78%	80.00%
CNN-RNN	50.46%	42.33%	70.00%

Table 9: Accuracy of each model

For all our models, we achieved a higher validation and training accuracy compared to our test. The reason for this, especially in the MLP and CNN models is most likely due to the fact that we had a low number of samples in for our input, causing these models to over fit. However the high validation accuracy achieved during tuning makes us optimistic that given more data samples for our MLP and CNN architectures that we may achieve a much more accurate final accuracy.

5.5 Limitations of Project Outcomes

1. Limitation to user interface functionalities

Our current user dashboard design fulfils the basic functions and requirements of the project where users are allowed to fetch data inputs from an external database and display the predicted analysed results. However, further improvements to prediction of patients data can be done especially on CNN-RNN model where we could output the connectivity matrices of a particular patient in the form of a video that transitions instead of multiple static images. As such it would be a good contender for future works as we were constrained by design skills and had to work on a limited schedule on the project to produce our primary deliverable which is a software that is able to classify MDD patients.

2. Backend web service limitation

We utilized the free tier of Amazon EC2 for hosting of our deep learning models. It comes with hardware specification of 1 GB of memory and 32-bit or 64-bit platform support. We were also bounded by a maximum data transfer of 15GB from the users and a total hosting time of 750 hours. To prevent costs from incurring on the free tier, we had to limit the network bandwidth, monitor and control the number of traffic allowed onto the website. The user may also experience performance issues when the traffic is at its highest as bottlenecks may occur when the system is fetching the output data to the user.

3. Hyper parameter Tuning

Especially for our CNN and CNN-LSTM architectures, the methods used to tune our hyper parameters was extremely time and resource expensive even when run on Google Colab. As such, in order to reduce the amount of time needed to be taken, we had to fit these models on lower epochs and were unable to test a larger amount of parameters (e.g for the CNN, we kept the maximum number of filters to 128, whereas a larger number may perform better).

4. Performance of deep learning models

As discussed within our results of the performance of all our models, our MLP, CNN and CNN-RNN models have an average prediction accuracy of 60%, 57.78% and 53.33% respectively. This is because we are inexperienced within the field of deep learning and we were provided with a small dataset to train our models. Further improvements to the performance can be carried out on our deep learning models such as algorithm tuning and this would be our primary goal of all the listed future works.

5.6 Discussion of Possible Improvements and Future Works

As mentioned from our limitations of project outcomes, there are certainly room to improve within our product. Besides the limitations to the user interface functionalities and performance of deep learning models, we could improve on the security flaws of our web server. The public IP and port numbers are currently exposed, we could improve it by binding the public IP and port numbers to a public domain and return a generic top-level domain.

Besides that, from a feedback from our supervisor, we could return the output from our CNN-RNN model to be a video format of connectivity matrices of the selected patient. With this method, the user can clearly visualize the differences of the connectivity matrix through the naked eye throughout the video frames.

6 Software Deliverable

6.1 Summary of Software Deliverable

Our software deliverable for this project is a web application called MDD Medical Portal. It currently supports login for doctors, whereby doctors can login to the portal using their respective credentials. However, in the future we hope to expand the portal so that it is accessible by patients, as this would create a one stop medical portal to connect patients and doctors. An image of the front-end login interface is included below:

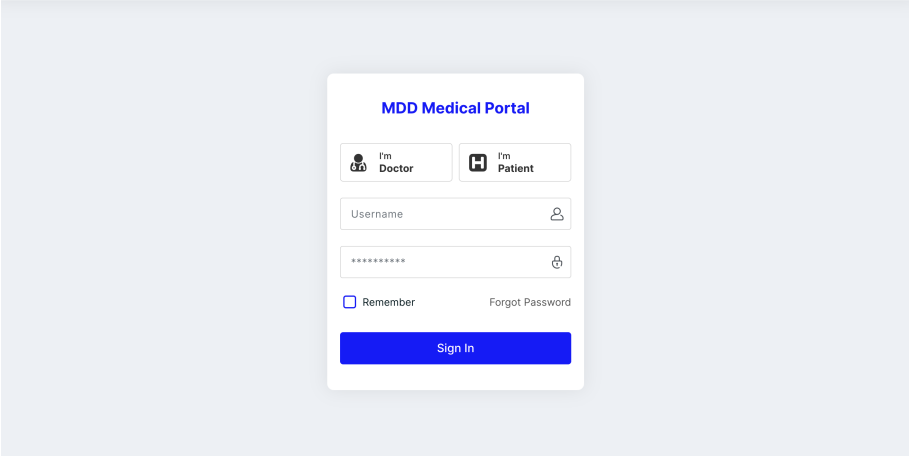

Figure 10: Image of the front-end login interface

Assuming that the login credentials of the doctor are correct, the doctor will then be redirected to the dashboard page where patients that are assigned to the user will be shown along with statistics of the medical portal. Included within the image are the average accuracy of all our current models.

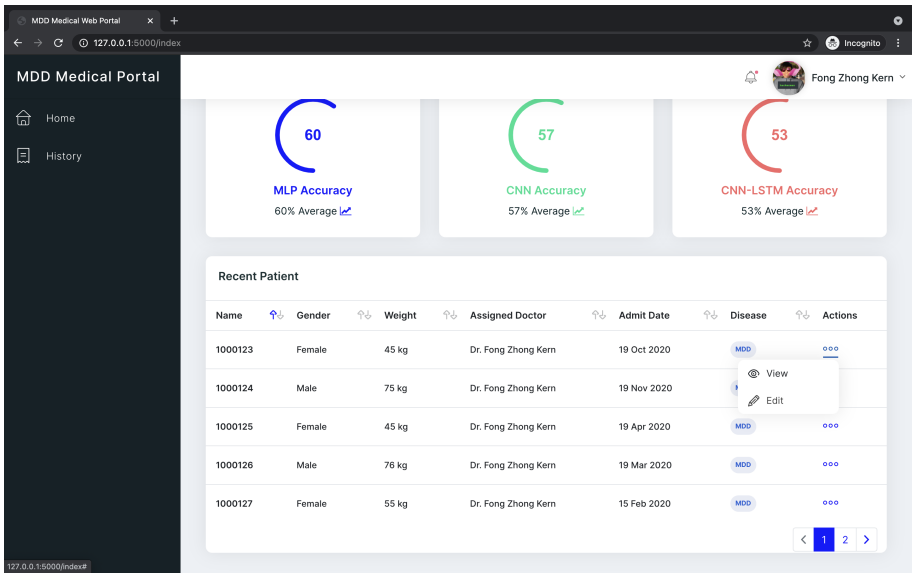

Figure 11: Image of user dashboard with patient details

Finally, doctors will be able to select a Deep Learning Model based on the selected patient data and predict it. A probability of how likely a patient could be a MDD patient is returned to the doctor for his reference. For CNN model and CNN-RNN model, images of connectivity matrices will be shown.

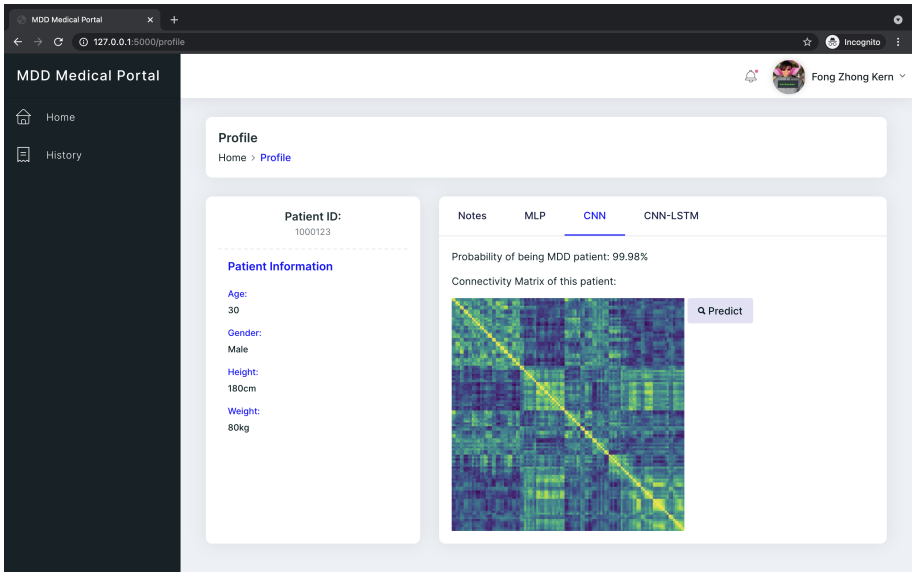

Figure 12: Image of model prediction based on patient data

A more complete set of screenshots and instructions to use will be provided in the user guide. Please refer that document for more information.

6.2 Summary and Discussion of Software Quality

6.2.1 Documentation and Maintainability

In terms of documentation, the main Python file keras_flask.py is well documented as that Python file contains all the code logic for our server. Each function in the file is documented with its aims and the expected input and output. In any circumstance there is any error happening when the back end is served, the programmer should be able to identify it easily. In terms of maintainability, a well documented code can be maintained easily. Besides, when designing the server, the design principles of Fail Fast and Do Not Repeat are implemented to ensure code maintainability and extensibility.

6.2.2 Security

In our software, there are multiple notable security flaws, some of each includes having a secured connection to the web application. In our product, the login link is a public IP address along with the port numbers that connects the user to the web server. In terms of cybersecurity, we are considered vulnerable as any user can perform a cyber-attack onto the web server using the public ip and ports. Flooding the domain with an overwhelming influx of packets that causes oversaturation onto the bandwidth is an example of a denial-of-service attack (DDOS) that the perpetrator could carry out to crash the web server.

6.2.3 Robustness

The robustness of our software is dependent on how well our web application handles wrong input and also mishandling of the user interface such as random clicking. In terms of handling wrong input, our web application will return an error message to the user that wrong input data has been fed in and the user has to contact the system administrator. When there is a mishandling of the user interface, the user will be redirected to an error 404 page where it will state that the page does not exist.

6.2.4 Usability

The usability of the web application achieves the expectation because a user guide with screenshots of the application and explanation is provided to the users. Besides, the design of the web application is the typical design of a medical dashboard. In terms of a Mobile view, our web application does not support a Mobile View because it would only be used in the hospital as it involves sensitive information. Hence the web application is not design for mobile view.

6.2.5 Scalability

The application is easily scalable because the back end server is hosted on Amazon EC2, which is a Cloud Service Platform that is meant for hosting large application. If we were to scale the application when more and more hospitals join to use this system, we could increase the machine image of our Amazon EC2 instance so that it can handle more concurrent requests to the server and the users would not experience any latency. Note that the cost of hosting such huge application will be expensive. In short, the web application in particular the server is highly scalable because we are hosting it using a service provider on the cloud.

6.3 Sample source codes

Source codes for our deep learning models (MLP, CNN and CNN-RNN) will be included within the appendix section.

7 Critical discussion on software project

7.1 Initial phase

During our initial project proposal stage, we adopted the waterfall project methodology as the project milestones, goal and objectives were clear and concise to us. We understood our tasks well which were to produce images of raw connectivity matrix of all patients and produce a CNN model that is able to classify all the patients with high prediction accuracy that would be hosted on a web server as a web application. With that goal in mind, the team understood that in order to achieve a high prediction accuracy on our proposed CNN model, we had to explore and implement optimization methods such as hyperparameter tuning, K-fold cross validation and perform leave-one-out during our model development phase. We have also expected that we may be unsuccessful in developing the model as the team members are not experts in the field of deep learning. Hence, it was ranked as the most probable risk to occur within our risk register from our project proposal with a large impact towards the project. However, that did not incur de-motivation among the members but in turn increased our drive in tackling the challenges.

7.2 Development phase

During the commencement of the project, we assigned and agreed upon our roles throughout the project, identified the project scope and milestones. Within the second week of the semester, we started coding on our CNN prototype from the proposal as our starting point and decided to improve from that basic model to achieve a higher accuracy as the prototype accuracy had a rough accuracy of 36%. What we did was that we took the raw connectivity matrices values and plot an image based on that and then fit those images into our CNN model. We thought that our CNN should take RGB images as its input. The accuracy is not acceptable because it is worse than flipping a coin. When we dig deeper into the model, we realized that our model was overfitting because the predicted values for our testing dataset are all the same. Upon coming up with the hypothesis that the model was overfitting, we test it out by reducing the initial layers of our CNN and got the model improving to the testing accuracy of 44% and this accuracy is the highest we could get for that period of time.

Therefore, we consulted our supervisor Prof Ting and Dr Fuad and shared with them our issues. After the discussions, we realized that we have misunderstood our supervisor during our weekly meetings which in turn has caused us 2 weeks of wasted efforts. We were not supposed to use the plotted images but to use the raw connectivity matrices as inputs to the CNN. At this stage, we chose to deviate away from our initial proposed project goal and chose to explore more deep learning models in hopes to find a model that is able to provide a high accuracy prediction as we are no longer bounded by an image data. Hence, instead of feeding the plotted connectivity matrices. to the CNN model as input, rather we converted the connectivity matrices into vectors and used these vectors to feed our CNN model and MLP model to train it.

The lesson learnt and the reflection gained here is that we should always communicate with our supervisor for guidance when we are really stuck on some parts of the projects. Our supervisor is always there to help guide us through as he has more experience in this field. Besides, from this incident we

have also realised how shallow our knowledge on the field of Deep Learning is. We would need to put in more hard work to make sure that our project is on the right track.

In the following 3 weeks, we were able to discover research papers that discusses on using MLP to classify misdiagnosis of Alzheimer's disease and vascular dementia. These papers are extremely helpful to us as we were implementing MLP during that period of time. During the 3 weeks, we were also given advice by the supervisor to explore on CNN-RNN architecture as he explains that it may be suitable to fit our input data. Our supervisor shared with us that if we were to train this CNN-RNN model, we would need to further pre-process our initial data in order to give a series dynamic connectivity matrices for each patient. We took up the challenge and stepped out of our comfort zone to make this work because implementing CNN-RNN is something that we have not done before even though we have taken a Deep Learning unit before. The lesson learnt here is that there will only be improvement when we struggle to complete something. Those struggles are what make us stronger and challenge us to do better.

With hard work and perseverance, within the next 3 weeks, we were able to implement all CNN, MLP and CNN-RNN models successfully, however we encountered our most anticipated risk. Our CNN and MLP were unable to achieve our expected accuracy even with K-fold cross validation implemented while our CNN-RNN model were unable to classify MDD patients from control patients. The CNN-RNN model was also very difficult and challenging to train as it requires a dynamic connectivity matrix which is obtained by slicing the time series data from fMRI images. This causes the number of trainable parameters to increase quickly if we were to use a large filter size to extract its features. As we raised this issue to our supervisor, he informed us that our final product will not be judged on achieving a high prediction accuracy. We were relieved as we thought that our project will be considered a failure if we did not produce high accuracy models.

Although we have spent most of the time developing a model, we were quite satisfied with the final accuracy of our models. This is because we were able to achieve accuracy that is slightly lower than our supervisor's models. This is a great example of where hard work pays off if we persist and persevere, and most importantly work as a team. In the last 3 weeks of the semester, we spent time deploying our trained deep learning model to server, designing the user interface of our web application, performing some software testings and also working on our reports and presentations.

7.3 Overall Comments

In short, we would consider the project a success as we were able to progress and learnt throughout the journey, most importantly stepping out of our comfort zone to learn something new and have also experienced the actual process of deploying a deep learning model to production. These experiences are extremely valuable to us as we are graduating soon and will join the working world. By learning how to deploy the deep learning model to production we could understand the end to end process of building an application with deep learning models and also have a glimpse of the job scope of a machine learning engineer.

In retrospect, some issues that should have been avoided was that we have allocated too much time in improvising the models to ensure robustness. As such, we reduced the amount of effort applied on other parts of the web application such as the front end and back end development. This is an oversight on our part because we did not balance out our workload well. What could have been done better is to stick to our project planning more closely and seek help immediately from the supervisor when we need it so that our internal deadlines are not exceeded and we meet our requirements.

Last but not least, having seen our final products and also looking back at our initial proposal, even though there are some differences especially the choice of models, we are glad that we were able to complete this project on time and achieved our initial aim of contributing to the medical field. By using our deep learning model, we are able to help MDD psychiatrists to predict preliminarily whether a person is a MDD patient not by using merely 1, but 3 Deep Learning models that could better help them make judgement.

8 Conclusion

Brain disorders are one of the most important health issues that our society faces. This can be proven as neurological disorders are the leading cause of disability adjusted life years (DALY) and the second leading cause of deaths worldwide in the year 2016 (Feigin et al., 2019), there is a lot of work to be done to ensure early and precise diagnosis are carried out by health professionals. Such ailments affects ones ability to swallow, move, remember, speak, breathe and feel. Some of the major neurological disorders occur in humans due to the result of dying or damaged nerve cells, there are currently no cure to such illnesses. However in cases of MDD, health professionals would need to diagnose a patient by identifying symptoms such as mood swings, lose interest or pleasure from daily activities for majority of the time or both, and interference with other psychiatric illnesses. However, such symptoms can be easily hid away by patients especially if they believe that they are in fear of losing something.

In this report, we used a similar methodology that was gathered and researched through past works of other researchers that are related to other neurological diseases. Despite the fact that the end product of our implementation may not be perfect, the project has been deemed a success by us as we were able to grow and learned throughout the project. We were able to provide a product that has the capability to classify patients that are suffering from MDD. Improving the prediction accuracy through our deep learning models by employing a broader public dataset is one of the future goals that we'd want to pursue. This project gave us the opportunity to gain expertise within the deep learning field as well as development of web application. The skills attained throughout the project would certainly be beneficial to all of us in the future, irrespective of the sector within computer science that would be pursued.

9 References

Taquet, M., Geddes, J. R., Husain, M., Luciano, S., amp; Harrison, P. J. (2021). 6-month neurological and psychiatric outcomes in 236379 survivors of COVID-19: a retrospective cohort study using electronic health records. The Lancet Psychiatry, 8(5), 416–427. https://doi.org/10.1016/s2215-0366(21)00084-5

World Health Organization. (2020, January 30). Depression. World Health Organization. https://www.who.int/news-room/fact-sheets/detail/depression.

Kennedy, S. H. (2008). Core symptoms of major depressive disorder: relevance to diagnosis and treatment. Dialogues in Clinical Neuroscience, 10(3), 271–277. https://doi.org/10.31887/dcns.2008.10.3/shkennedy

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Berg, A. C. (2015). Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3), 211-252.

Elster, A. D. (2019). Invention of MRI. Questions and Answers in MRI. http://mri-q.com/who-invented-mri.html.

Glover G. H. (2011). Overview of functional magnetic resonance imaging. Neurosurgery clinics of North America, 22(2), 133–vii. https://doi.org/10.1016/j.nec.2010.11.001

Khosla, M., Jamison, K., Ngo, G. H., Kuceyeski, A., Sabuncu, M. R. (2019). Machine learning in resting-state fMRI analysis. Magnetic resonance imaging, 64, 101-121.

Strother, S. C. (2006). Evaluating fMRI preprocessing pipelines. IEEE Engineering in Medicine and Biology Magazine, 25(2), 27-41

Gavrilescu, M., Stuart, G. W., Rossell, S., Henshall, K., McKay, C., Sergejew, A. A., ... Egan, G. F. (2008). Functional connectivity estimation in fMRI data: influence of preprocessing and time course selection. Human brain mapping, 29(9), 1040-1052.

Castellazzi, G., Cuzzoni, M. G., Cotta Ramusino, M., Martinelli, D., Denaro, F., Ricciardi, A., Vitali, P., Anzalone, N., Bernini, S., Palesi, F., Sinforiani, E., Costa, A., Micieli, G., D'Angelo, E., Magenes, G., amp; Gandini Wheeler-Kingshott, C. A. (2020). A Machine Learning Approach for the Differential Diagnosis of Alzheimer and Vascular Dementia Fed by MRI Selected Features. Frontiers in Neuroinformatics, 14. https://doi.org/10.3389/fninf.2020.00025

Yan, W., Calhoun, V., Song, M., Cui, Y., Yan, H., Liu, S., Fan, L., Zuo, N., Yang, Z., Xu, K., Yan, J., Lv, L., Chen, J., Chen, Y., Guo, H., Li, P., Lu, L., Wan, P., Wang, H., ... Sui, J. (2019). Discriminating schizophrenia using recurrent neural network applied on time courses of multi-site FMRI data. EBioMedicine, 47, 543–552. https://doi.org/10.1016/j.ebiom.2019.08.023

Shahriman, W. N., Phang, C. R., Numan, F., amp; Ting, C. M. (2020). Classification of Brain Functional Connectivity using Convolutional Neural Networks. IOP Conference Series: Materials Science and Engineering, 884, 012003. https://doi.org/10.1088/1757-899x/884/1/012003

Gao, S., Calhoun, V. D., Sui, J. (2018). Machine learning in major depression: From classification to treatment outcome prediction. CNS neuroscience therapeutics, 24(11), 1037-1052.

Ke, H., Chen, D., Shah, T., Liu, X., Zhang, X., Zhang, L., Li, X. (2020). Cloud-aided online EEG classification system for brain healthcare: A case study of depression evaluation with a lightweight CNN. Software: Practice and Experience, 50(5), 596-610.

Ioffe, S., Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). PMLR.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-

1958.

Manav M (May 1 2021). Convolutional Neural Networks (CNN). https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

Phi M. (25 Sep 2018) Illustrated Guide to LSTM's and GRU's: A step by step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Feigin, V. L., Nichols, E., Alam, T., Bannick, M. S., Beghi, E., Blake, N., Culpepper, W. J., Dorsey, E. R., Elbaz, A., Ellenbogen, R. G., Fisher, J. L., Fitzmaurice, C., Giussani, G., Glennie, L., James, S. L., Johnson, C. O., Kassebaum, N. J., Logroscino, G., Marin, B., ... Vos, T. (2019). Global, regional, and national burden of neurological disorders, 1990–2016: a systematic analysis for the Global Burden of Disease Study 2016. The Lancet Neurology, 18(5), 459–480. https://doi.org/10.1016/s1474-4422(18)30499-x

Sidén, Per. (2020). Scalable Bayesian spatial analysis with Gaussian Markov random fields. 10.3384/diss.diva-165872.

Mozaffari, Ladan Mozaffari, Ahmad Azad, Nasser. (2014). Vehicle speed prediction via a sliding-window time series analysis and an evolutionary least learning machine: A case study on San Francisco urban roads. Engineering Science and Technology, an International Journal. 6. 10.1016/j.jestch.2014.11.002.

Hingarajiya, A. (2018). dropways/deskapp. GitHub. https://github.com/dropways/deskapp.

Phan, D. (2020). How to Deploy a Flask App on AWS EC2 Instance. Twilio Blog. https://www.twilio.com/blog/deploy-flask-python-app-aws.

10 Appendix

Final report contribution

Sections	**Authors**
Cover	Choo Jun Yi, Fong Zhong Kern, Khoo Teng Ian
Introduction	Fong Zhong Kern
Background	Fong Zhong Kern
Methodology	Khoo Teng Ian, Choo Jun Yi
Project Management	Fong Zhong Kern
Outcomes	Choo Jun Yi, Khoo Teng Ian, Fong Zhong Kern
Conclusion	Fong Zhong Kern

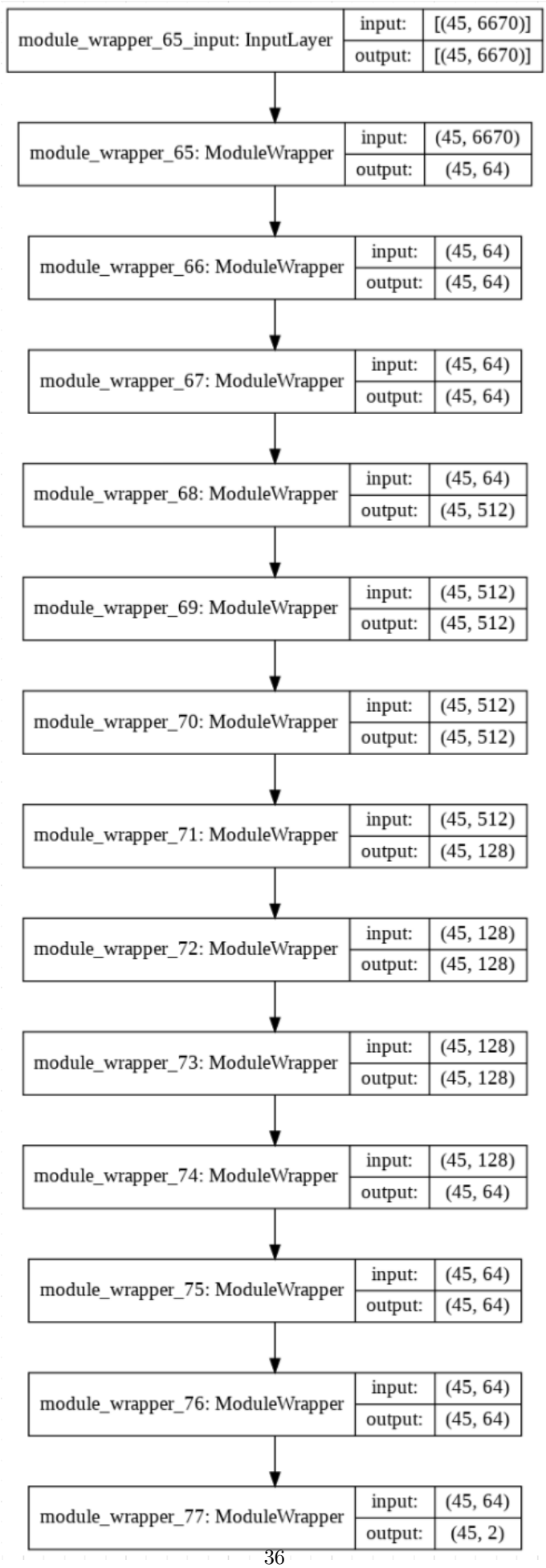

Figure 13: MLP configuration

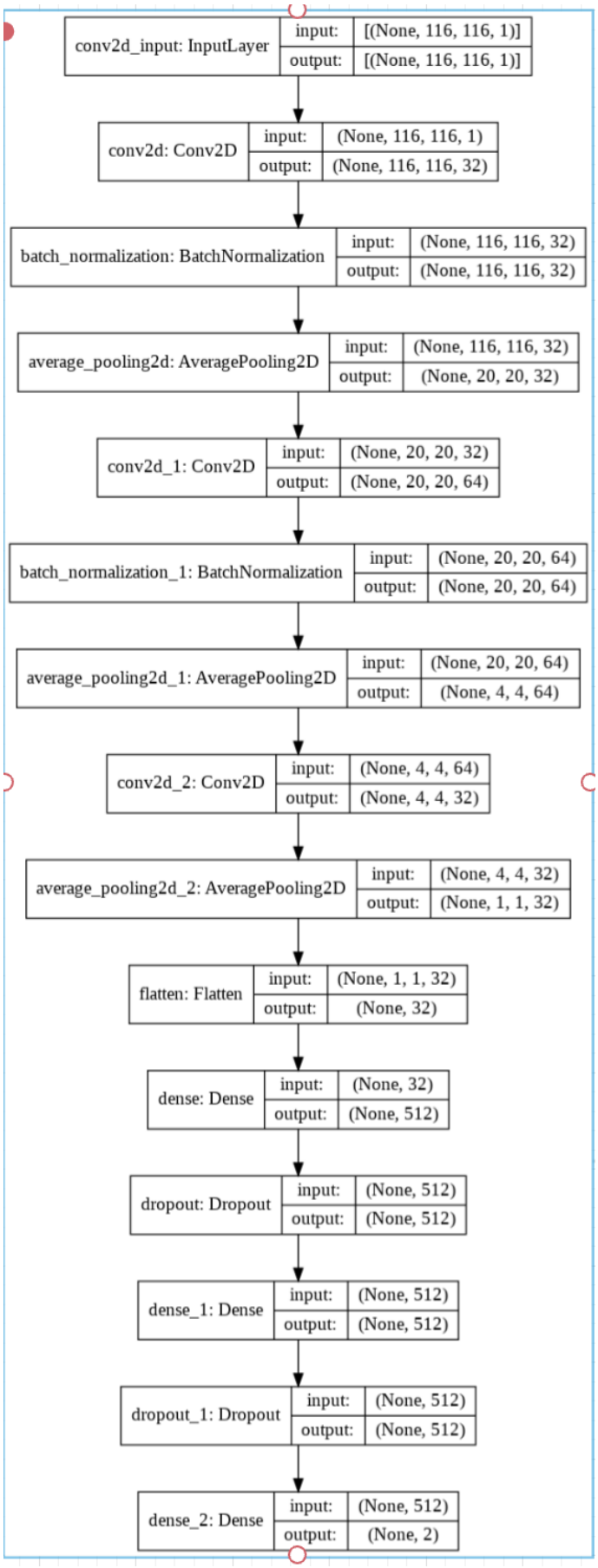

Figure 14: CNN configuration

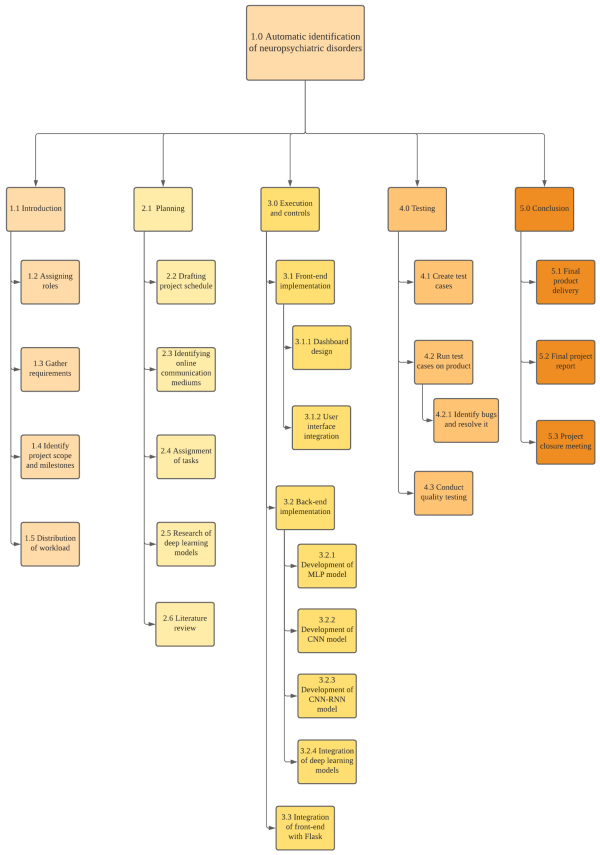

Figure 15: Work breakdown structure

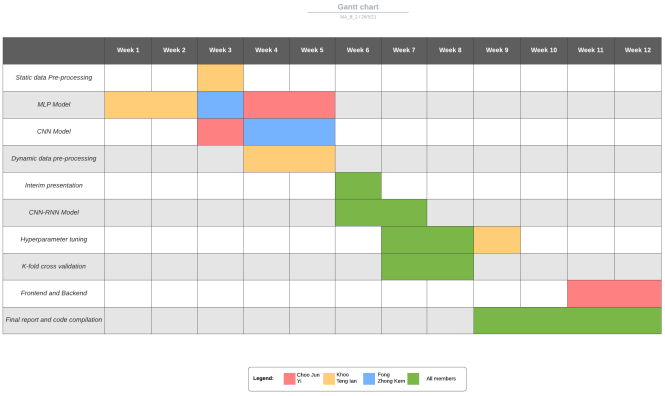

Figure 16: Updated gantt chart from project proposal

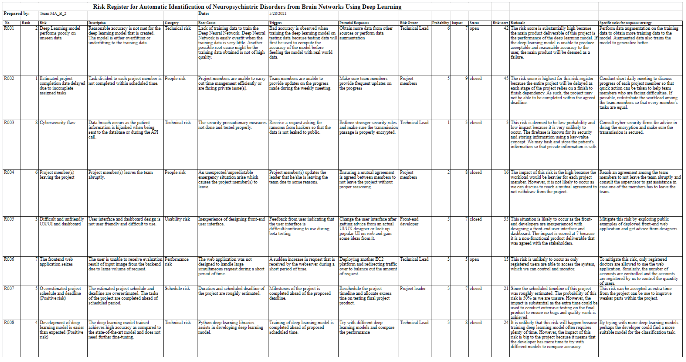

Figure 17: Updated risk register from project proposal

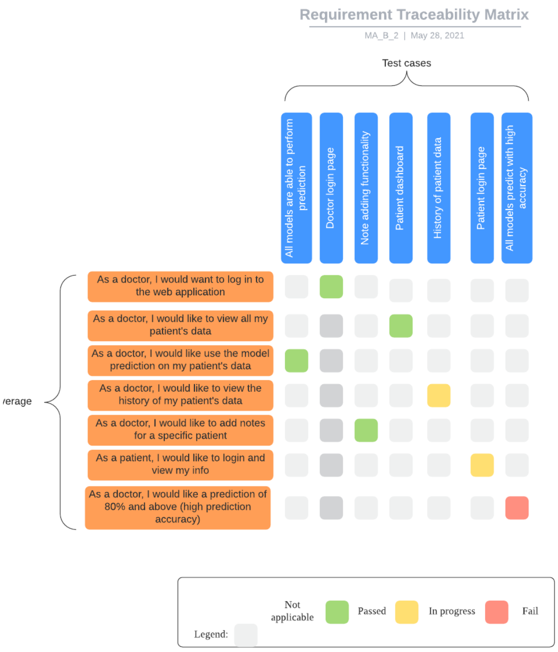

Figure 18: Requirement traceability matrix of our project

-0.875	0.5857142746	0.377777791	[0.01, 4, 64, 512, 128, 64, 'tanh', 'tanh', 'sigmoid', 'tanh', 5, 0.6]
-0.85	0.6928571403	0.4222222298	[0.006833390623096185, 4, 512, 256, 64, 512, 'sigmoid', 'tanh', 'sigmoid', 'tanh', 5, 0.4]
-0.825	0.4928571403	0.4666666746	[1.3096821649319518e-06, 3, 32, 32, 32, 512, 'tanh', 'relu', 'relu', 'sigmoid', 5, 0.5]
-0.825	0.7071428537	0.3333333403	[0.007239993466098751, 4, 256, 256, 64, 256, 'sigmoid', 'tanh', 'sigmoid', 'sigmoid', 5, 0.2]
-0.825	0.7999999881	0.3333333403	[0.0073215159700503974, 4, 64, 256, 64, 128, 'sigmoid', 'tanh', 'sigmoid', 'tanh', 5, 0.3]
-0.825	0.8285714388	0.3777777836	[0.01, 4, 128, 512, 64, 512, 'tanh', 'relu', 'relu', 'tanh', 6, 0.5]
-0.8	0.6071428716	0.3333333388	[0.009622705323316145, 4, 128, 512, 512, 256, 'sigmoid', 'relu', 'sigmoid', 'tanh', 5, 0.4]
-0.8	0.6142857194	0.5111111224	[0.01, 4, 64, 512, 64, 64, 'tanh', 'sigmoid', 'relu', 'sigmoid', 5, 0.5]
-0.8	0.7000000119	0.3777777895	[0.01, 4, 256, 512, 512, 512, 'sigmoid', 'tanh', 'relu', 'sigmoid', 6, 0.6]
-0.8	0.7714285731	0.5111111224	[0.007085792773778615, 4, 512, 256, 64, 512, 'sigmoid', 'tanh', 'sigmoid', 'sigmoid', 5, 0.1]
-0.8	0.8214285851	0.5555555701	[0.01, 4, 64, 512, 32, 64, 'sigmoid', 'tanh', 'sigmoid', 'tanh', 6, 0.1]
-0.775	0.5571428537	0.3777777851	[0.00802837402023977, 4, 64, 256, 256, 64, 'sigmoid', 'relu', 'sigmoid', 'relu', 5, 0.6]
-0.775	0.5785714269	0.5111111298	[0.009204085208172917, 3, 32, 32, 64, 64, 'tanh', 'sigmoid', 'tanh', 'tanh', 10, 0.5]
-0.775	0.6214285672	0.4444444597	[0.001, 4, 64, 32, 32, 64, 'relu', 'relu', 'relu', 'relu', 5, 0.2]

Figure 19: Top results of Hyperparameter Tuning MLP model

-0.85	0.8357142806	0.5555555701	[0.01, 3, 2, 4, 3, 3, 64, 32, 32, 6, 512, 512, 0.0, 6]
-0.825	0.7428571343	0.6000000119	[0.01, 3, 2, 5, 4, 4, 32, 64, 32, 6, 512, 512, 0.4, 5]
-0.825	0.9428571343	0.4888889074	[0.01, 3, 2, 8, 6, 7, 64, 32, 64, 6, 128, 256, 0.0, 10]
-0.8	0.6428571343	0.577777791	[0.01, 3, 2, 8, 6, 8, 64, 128, 64, 6, 128, 64, 0.4, 8]
-0.8	0.7714285731	0.4000000134	[0.01, 3, 2, 3, 3, 3, 32, 32, 32, 6, 512, 32, 0.2, 5]
-0.8	0.7857142925	0.3777777895	[0.01, 3, 2, 8, 7, 7, 64, 32, 32, 6, 256, 256, 0.0, 8]
-0.8	0.8357142806	0.577777791	[0.01, 3, 2, 8, 6, 7, 128, 64, 64, 6, 512, 128, 0.5, 10]
-0.8	0.85	0.4444444567	[0.01, 3, 2, 7, 5, 7, 64, 128, 64, 6, 128, 512, 0.4, 10]
-0.8	0.8571428537	0.3333333418	[0.01, 3, 2, 7, 5, 6, 64, 32, 32, 6, 256, 512, 0.2, 10]
-0.775	0.5571428657	0.4888889074	[0.01, 2, 2, 8, 8, 8, 32, 32, 128, 4, 512, 512, 0.0, 5]
-0.775	0.7071428657	0.5111111283	[0.01, 3, 2, 8, 7, 7, 64, 64, 32, 6, 256, 512, 0.4, 5]

Figure 20: Top results Hyperparameter tuning CNN Model

-0.775	0.649999994	0.5333333433	[0.01, 3, 2, 3, 8, 8, 32, 64, 64, 6, 32, 512, 0.5, 5]
-0.75	0.6428571463	0.4666666806	[0.009649062598134578, 3, 2, 7, 7, 8, 32, 64, 32, 6, 64, 32, 0.4, 5]
-0.75	0.6642857194	0.4444444567	[0.01, 3, 2, 4, 3, 6, 64, 64, 64, 6, 32, 512, 0.2, 5]
-0.75	0.7285714269	0.3555555642	[0.01, 3, 2, 5, 6, 8, 32, 64, 64, 6, 64, 256, 0.4, 5]
-0.75	0.75	0.4666666806	[0.01, 3, 2, 5, 8, 7, 32, 128, 64, 6, 512, 512, 0.5, 5]
-0.75	0.7714285731	0.4000000089	[0.01, 3, 1, 5, 8, 8, 32, 32, 64, 6, 512, 512, 0.2, 10]
-0.75	0.7785714269	0.3777777866	[0.01, 3, 1, 6, 6, 8, 128, 32, 32, 6, 128, 32, 0.5, 8]
-0.75	0.8714285731	0.3555555671	[0.01, 3, 2, 6, 4, 7, 64, 64, 128, 6, 64, 128, 0.5, 6]
-0.725	0.6571428597	0.4222222403	[0.01, 3, 1, 8, 7, 5, 64, 64, 32, 6, 512, 128, 0.5, 5]
-0.725	0.6785714328	0.4888889015	[0.01, 2, 2, 7, 8, 3, 32, 64, 32, 6, 32, 64, 0.5, 5]
-0.725	0.7285714388	0.5333333552	[0.01, 3, 2, 6, 5, 8, 32, 32, 32, 6, 64, 512, 0.5, 6]
-0.725	0.7571428537	0.4000000119	[0.01, 3, 2, 8, 8, 8, 32, 128, 32, 6, 512, 32, 0.5, 5]

Figure 21: Top results Hyperparameter tuning on our CNN-LSTM Model

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, BatchNormalization, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import load_model
# package used to load MATLAB files
import scipy.io as sio
# numerical prcomputing package for arrays
import numpy as np
# package to visualize the data as plots
import matplotlib.pyplot as plt
```

Figure 22: Libraries used for MLP

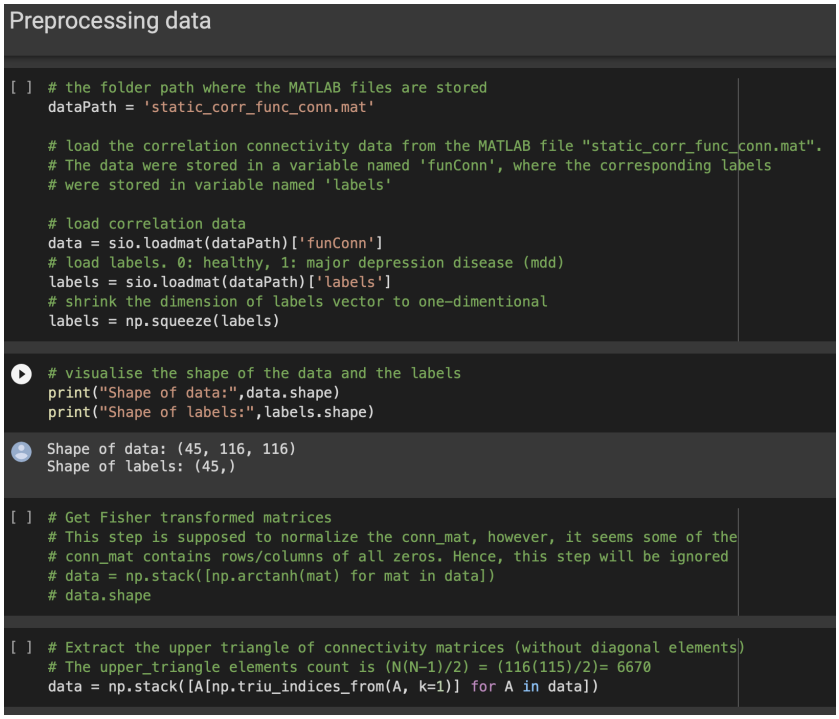

Figure 23: Data preprocessing for MLP

```
# define model with the best tuned hyperparameters
def build_model():
    model = Sequential()
    model.add(Dense(64, activation='tanh', kernel_initializer='he_normal', input_shape=(6670,)))
    model.add(BatchNormalization(momentum = 0.9))
    model.add(Dropout(0.6))
    model.add(Dense(512, activation='tanh', kernel_initializer='he_normal'))
    model.add(BatchNormalization(momentum = 0.9))
    model.add(Dropout(0.6))
    model.add(Dense(128, activation='sigmoid', kernel_initializer='he_normal'))
    model.add(BatchNormalization(momentum = 0.9))
    model.add(Dropout(0.6))
    model.add(Dense(64, activation='tanh', kernel_initializer='he_normal'))
    model.add(BatchNormalization(momentum = 0.9))
    model.add(Dropout(0.6))
    # model.add(Dense(4, activation='relu', kernel_initializer='he_normal'))
    model.add(Dense(num_classes, activation='softmax'))
    # compile the model
    opt = Adam(learning_rate=0.01)
    model.compile( optimizer=opt,
                  loss='categorical_crossentropy',
                  metrics=['accuracy'] )
    return model
```

Figure 24: Build function for our MLP model

```
] print("Test  Accuracy: {0:.2%}".format(np.mean(test_accs)))
  print("Train Accuracy: {0:.2%}".format(np.mean(train_accs)))

  Test  Accuracy: 60.00%
  Train Accuracy: 79.44%
```

Figure 25: Testing Accuracy of MLP model

```
#from nilearn import datasets
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
plt.style.use('ggplot')

#nilearn - neuroimaging tailored library
#from nilearn.input_data import NiftiMapsMasker
#from nilearn import plotting

#sklearn - basic ML tools
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn import metrics

#keras - for NN models
from keras.models import Model, Sequential, load_model
from keras import optimizers
# from keras.utils import plot_model
from keras import utils
from sklearn.metrics import roc_curve
from keras.layers import Dense, Dropout, Conv2D, MaxPool2D, Flatten, BatchNormalization, AveragePooling2D
from PIL import Image
import os
from keras.preprocessing.image import img_to_array


#scipy- statistical analysis tools
from scipy.stats import ttest_1samp
from scipy import interp
import scipy.io as sio

no_folds = 5
K_SEED = 68231
```

Figure 26: Libraries used in our CNN model

```
# load data from matlab file
dataPath = 'static_corr_func_conn.mat'
# load correlation data
data = sio.loadmat(dataPath)['funConn']
# load labels. 0: healthy, 1: major depression disease (mdd)
labels = sio.loadmat(dataPath)['labels']
# shrink the dimension of labels vector to one-dimentional
labels = np.squeeze(labels)
# find the total number of classes from the labels vector
num_classes = len(np.unique(labels))
```

```
labels=labels.flatten()
labels
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1], dtype=uint8)
```

```
data = np.expand_dims(data,-1)
data.shape
```

```
(45, 116, 116, 1)
```

Figure 27: Data Preprocessing for CNN

```
# this is the variables that we found to optimise our CNN model
# params = [0.01, 3, 2, 4, 3, 3, 64, 32, 32, 6, 512, 512, 0.0, 6]
params = [0.01, 5, 4, 4, 32, 64, 32, 6, 512, 512, 0.4, 5]

kernel_1 = (params[1],params[1])
kernel_2 = (params[2],params[2])
kernel_3 = (params[3],params[3])
filter_1 = params[4]
filter_2 = params[5]
filter_3 = params[6]
pool_size = (params[7],params[7])
dense_1 = params[8]
dense_2 = params[9]
dropout = params[10]
batch_size = params[11]
```

Figure 28: Defining the best parameters to use in our CNN

```
# Creation of CNN model
# The model hyper-parameter is not tuned and just using a random value
def build_model():
    model = Sequential()
    model.add(Conv2D(filter_1, kernel_size= kernel_1, strides = (1,1), padding='same', activation='relu', input_shape=(116,116,1)))
    model.add(BatchNormalization(momentum=0.9))
    model.add(AveragePooling2D(pool_size= pool_size, padding='same'))
    model.add(Conv2D(filter_2, kernel_size = kernel_2, strides = (1,1), padding='same', activation='relu'))
    model.add(BatchNormalization(momentum=0.9))
    model.add(AveragePooling2D(pool_size= pool_size, padding='same'))
    model.add(Conv2D(filter_3, kernel_size= kernel_3, strides = (1,1), padding='same', activation='relu'))
    model.add(AveragePooling2D(pool_size= pool_size, padding='same'))
    model.add(Flatten())
    model.add(Dense(dense_1, activation='relu'))
    model.add(Dropout(dropout))
    model.add(Dense(dense_2, activation='relu'))
    model.add(Dropout(dropout))
    model.add(Dense(num_classes, activation='softmax'))
    # Compile the model
    model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')
    # see the number of trainable and non-trainable hyperparamters
    return model
```

Figure 29: Build function for our CNN model

```
#from nilearn import datasets
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
import csv
warnings.filterwarnings("ignore")
plt.style.use('ggplot')

#nilearn - neuroimaging tailored library
#from nilearn.input_data import NiftiMapsMasker
#from nilearn import plotting

#sklearn - basic ML tools
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn import metrics

#keras - for NN models
from tensorflow.keras.models import Model, Sequential, load_model
from keras import optimizers
# from keras.utils import plot_model
from keras import utils
from sklearn.metrics import roc_curve
from tensorflow.keras.layers import Dense, Flatten, MaxPooling3D,BatchNormalization, Dropout, ConvLSTM2D, AveragePooling3D
from PIL import Image
import os
from keras.preprocessing.image import img_to_array


#scipy- statistical analysis tools
from scipy.stats import ttest_1samp
from scipy import interp
import scipy.io as sio

no_folds = 5
K_SEED = 68231
```

Figure 30: Libraries used in our CNN-LSTM model

```
file_name = '/content/drive/My Drive/conn_mat_20.csv'
# load correlation data
conn_mat = [0] * 45
for i in range(45):
    conn_mat[i] = [0] * 20
    for j in range(20):
        conn_mat[i][j] = [0] * 116

row_count = 0
matrix_count = 0
patient = 0
# total_count = 0

with open(file_name, mode = 'r') as f:
    data = csv.reader(f, delimiter = ',')
    for row in data:
        lst = []
        for item in row:

            lst.append(float(item)) # list of 116 items

        conn_mat[patient][matrix_count][row_count] = lst

        row_count += 1
        #total_count += 1
        if row_count >= 116:
            row_count = 0
            matrix_count += 1
            if matrix_count >= 20:
                matrix_count = 0
                patient += 1

data = conn_mat
data = np.expand_dims(data, -1)
# the folder path where the MATLAB files are stored
# dataPath = 'mdd_fmri_data.mat'
dataPath = '/content/drive/My Drive/mdd_fmri_data.mat'

# load correlation data
labels = sio.loadmat(dataPath)['labels']
# shrink the dimension of labels vector to one-dimentional
labels = np.squeeze(labels)
# find the total number of classes from the labels vector
num_classes = len(np.unique(labels))
```

Figure 31: Data Preprocessing for our CNN-LSTM model

```
# this is the variables that we found to optimise our CNN model
params = [0.01, 5, 4, 4, 32, 64, 32, 6, 512, 512, 0.4, 5]
kernel_1 = (params[1],params[1])
kernel_2 = (params[2],params[2])
kernel_3 = (params[3],params[3])
filter_1 = params[4]
filter_2 = params[5]
filter_3 = params[6]
pool_size = (1,params[7],params[7])
dense_1 = params[8]
dense_2 = params[9]
dropout = params[10]
batch_size = params[11]
```

```
# Creation of CNN_RNN model
# The model hyper-parameter is not tuned and just using a random value
def build_model():
    model = Sequential()
    model.add(ConvLSTM2D(filter_1, kernel_size= kernel_1, strides = (1,1), padding='same', activation='relu', input_shape=(20,116,116,1), return_sequences= True))
    model.add(BatchNormalization(momentum=0.9))
    model.add(AveragePooling3D(pool_size= pool_size, padding='same'))
    model.add(ConvLSTM2D(filter_2, kernel_size = kernel_2, strides = (1,1), padding='same', activation='relu', return_sequences= True))
    model.add(BatchNormalization(momentum=0.9))
    model.add(AveragePooling3D(pool_size= pool_size, padding='same'))
    model.add(ConvLSTM2D(filter_3, kernel_size= kernel_3, strides = (1,1), padding='same', activation='relu', return_sequences= True))
    model.add(AveragePooling3D(pool_size= pool_size, padding='same'))
    model.add(Flatten())
    model.add(Dense(dense_1, activation='relu'))
    model.add(Dropout(dropout))
    model.add(Dense(dense_2, activation='relu'))
    model.add(Dropout(dropout))
    model.add(Dense(num_classes, activation='softmax'))
    # Compile the model
    model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')
    # see the number of trainable and non-trainable hyperparamters
    return model
```

Figure 32: Hyper parameters used, as well as the building of our CNN-LSTM model

```
print("Test  Accuracy: {0:.2%}".format(np.mean(test_accs)))
print("Train Accuracy: {0:.2%}".format(np.mean(train_accs)))
```

```
Test  Accuracy: 53.33%
Train Accuracy: 61.11%
```

Figure 33: Testing Accuracy of our CNN-LSTM model

```
from sklearn.model_selection import StratifiedKFold
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import ModelCheckpoint

no_folds = 5
K_SEED = 68231
num_classes = 2

val_accs = np.zeros(no_folds)
train_accs = np.zeros(no_folds)
test_accs = np.zeros(no_folds)
best_train_accs = []
best_test_accs = []

model_path = "./"

kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=K_SEED)
i_fold = 0
for train_index, test_index in kf.split(data,labels):

    train_data, test_data, y_train, y_test, = data[train_index], data[test_index], labels[train_index], labels[test_index]

    #  Create one-hot encode of integer labels
    train_labels = to_categorical(y_train, num_classes)
    test_labels = to_categorical(y_test, num_classes)

    # Build model
    model = build_model()
    # Use Keras to train the model.
    filename = model_path + 'best_mlp_model.hdf5'
    checkpointer = ModelCheckpoint(filepath=filename, monitor='accuracy', verbose=0,
                                   save_best_only=True)
    history = model.fit(x=train_data,
                        y=train_labels,
                        epochs=100,
                        batch_size= 5,
                        verbose=0,
                        callbacks=[checkpointer])

    train_acc_epochs = history.history['accuracy'] #best train score
    best_epoch = np.argmax(train_acc_epochs)
    train_accs[i_fold] = train_acc_epochs[best_epoch]
```

Figure 34: First part of our Training Function

```
        del model
        model = load_model(filename)
        _,test_accs[i_fold] = model.evaluate(test_data, test_labels, verbose=0)

        i_fold += 1

        best_train_accs.append(np.mean(train_accs))
        best_test_accs.append(np.mean(test_accs))
```

Figure 35: Second part of our training function