Final Report

Comparing Interpretability Metrics for Diagnostic Classification of Neuropsychiatric Disorders Using Clinical MRI Data

CM3203 One Semester Individual Project
40 Credits

Author:
Elise Bailey

Supervisor:
Dr Matthias Treder

May 2020

ABSTRACT

The use of deep learning models in diagnostic classification has been gaining increasing attention due to its outstanding performance over traditional machine learning models, especially in the domain of computer vision. However, such models have a reputation of being highly uninterpretable and are often labelled as ‘black boxes’. This project aims to compare and evaluate different interpretability methods that can be used to increase the transparency of deep learning models. These methods are Layer-wise Relevance Propagation heatmaps, saliency maps and images produced using the Lime toolbox. A convolutional neural network was developed to classify patients with neuropsychiatric disorders from controls, but satisfactory results were not obtained. The interpretability methods were therefore applied to a convolutional neural network trained to classify MRIs based on age. Qualitative and quantitative evaluations were used to show that the methods successfully increase the interpretability of the model and could therefore one day be used by clinicians to increase their trust in deep learning models as a diagnostic tool. More specifically, it was found that Layer-wise Relevance Propagation heatmaps and saliency maps are more successful at increasing interpretability than images produced using the Lime toolbox.

ACKNOWLEDGEMENTS

I would like to thank Dr Matthias Treder, my supervisor, for his help and guidance throughout the project. He ensured the project was enriching and that I was continually learning along the way.
I would also like to thank the team at Stellenbosch University for allowing me access to their Shared Roots data.

Table of Contents

1 INTRODUCTION

Neuropsychiatric disorders are a group of medical conditions that involve both neurology and psychiatry. This means they have a physical component relating to the nervous system as well as a mental component that causes emotional distress and abnormal behaviour. The disorders that the data that will be used comprises of are schizophrenia, posttraumatic stress disorder (PTSD) and Parkinson's disease (PD). Table 1 shows the number of people estimated to be living with these disorders worldwide.

Neuropsychiatric Disorder	Frequency Worldwide
Parkinson's	10 million[1]
Schizophrenia	3.2 million[2]
PTSD	3 in 100 people in a given year[3]

Table 1 The estimated number of people living with Parkinson's and Schizophrenia worldwide and the estimated number of people who suffer with PTSD each year worldwide (as it is curable).

However, the number of people these disorders effect is much greater. For example, 145,000 people in the UK have been diagnosed with Parkinson's, but it is estimated that more than 1 million people are affected by the diseases, either by living with Parkinson's or as a friend, colleague or family member of someone who is.[4] It is therefore incredibly important to continue to increase our understanding of these disorders. By increasing understanding, we can make both diagnostic and treatment advances that would have a direct positive impact on both the present and future sufferers of these disorders.

Clinical diagnosis of these disorders is still a challenge as they are very complex and largely not understood. However, the physical changes in the brain mean that machine learning can be used to diagnose patients using MRI scans. An MRI is a 3D imaging method that uses strong magnetic waves to produce detailed images of the organs and tissues within the body. MRIs are therefore very suited to detecting changes in volume of grey matter (a type of tissue in the brain), a key sign of a neuropsychiatric disorder, as well as other neuroanatomical changes that are characteristic of such disorders.

The first aim of this project is to develop a convolutional neural network (CNN) model that can accurately classify MRIs from healthy patients and MRIs from patient with a neuropsychiatric disorder. A CNN is the most suitable model as they are able to automatically detect the important features of images without any supervision. [5] This means information about which parts of the brain correspond to neuropsychiatric disorders does not need to be associated with the data that is fed into the CNN; it will learn this for itself.

The second aim of the project is to compare different interpretability methods currently available to help gain an understanding of the choices made by the CNN during training. CNNs are notoriously uninterpretable because they consist of many hidden layers that make significant, but complex, computational decisions. Being able to understand these decisions is important to help us gain a better understanding of these disorders and how to diagnose them. This project aims to uncover by how much these different methods increase the transparency of the CNN and therefore the interpretability of the results. One of the biggest challenges is

finding a fair, quantitative way to measure just how interpretable these methods make the models. The overall goal of this project is therefore to objectively and fairly evaluate how well these different methods produce consistent, interpretable and useful information on the diagnostic classification of neuropsychiatric disorders using a CNN.

2 BACKGROUND

2.1 Concepts

MRIs

An MRI is a 3D imaging method that uses strong magnetic waves to produce detailed images of the organs and tissues within the body. MRIs are therefore very suited to detecting changes in volume of grey matter (a type of tissue in the brain), a key sign of a neuropsychiatric disorder, as well as other neuroanatomical changes that are characteristic of such disorders.

Machine Learning

Machine learning is a subset of Artificial Intelligence which uses algorithms that can modify themselves without human intervention in order to improve at tasks with experience. The algorithms use statistics to find and learn patterns from large amounts of data. Machine learning models can then apply what they have learnt to new, unseen data in order to solve problems. Machine learning algorithms often fall into two categories; supervised and unsupervised. Supervised models used labelled data to learn about the past events in order to predict future events. The two types of supervised learning are classification and regression. Unsupervised models look for patterns in un-labelled data, such as clustering, which groups data points that are similar. The approach used in this project is supervised classification which is where the model will learn features of each pre-determined class from the labelled data it is trained on and then use the information it has learnt to predict the classes of unseen data it is tested on.

Deep Learning

Deep learning is a branch of machine learning that was inspired by the human brain. It creates artificial neural networks (ANNs) that are able to learn by experience and acquire skills without human involvement from a large amount of data that is unstructured or unlabelled. The algorithms get better with experience but require more data than classical machine learning algorithms as they have a much larger number of parameters that need to be tuned. The structure of such algorithms mean that they are able to solve non-leaner, complex problems that classical machine learning algorithms cannot.

Artificial Neural Network Structure

Like the human brain, ANNs are made up of neuron nodes that are interconnected like a web. These nodes are made up of input and output units and are where computation happens in the form of an activation function; the input is combined with a set of weights that either amplify or dampen that input, thereby assigning it significance with regard to the task the algorithm is trying to learn. Figure 1 shows the structure of a node in an ANN.

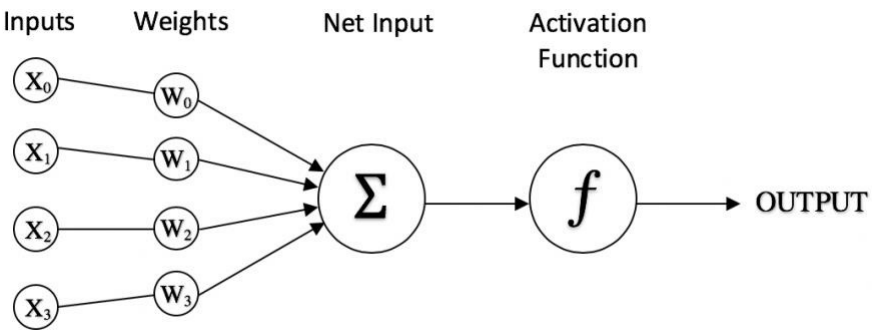

Figure 1: A simple illustration of a node in an artificial neural network

Typically, an ANN has thousands to millions of these neurons, arranged in a series of layers. First is the input layer which receives the data from an external source. Next are a series of hidden layers, each of which group a large number of neurons together. It is these layers that define an algorithm as deep learning and makes it superior to other machine learning algorithms. Each hidden layer can have a different number of neurons and each neuron can be connected to any number of the neurons in the next hidden layer. A layer is called fully connected if all of the neurons are connected to each and every neuron in the next layer. The type of computations performed by the hidden layers and the activation functions used depend on the type of neural network, which in turn depends on the application. Lastly there is an output layer which maps the outputs from the last hidden layer to the desired number of outputs in the output range. Figure 2 shows the different layers of an ANN and how they may be connected.

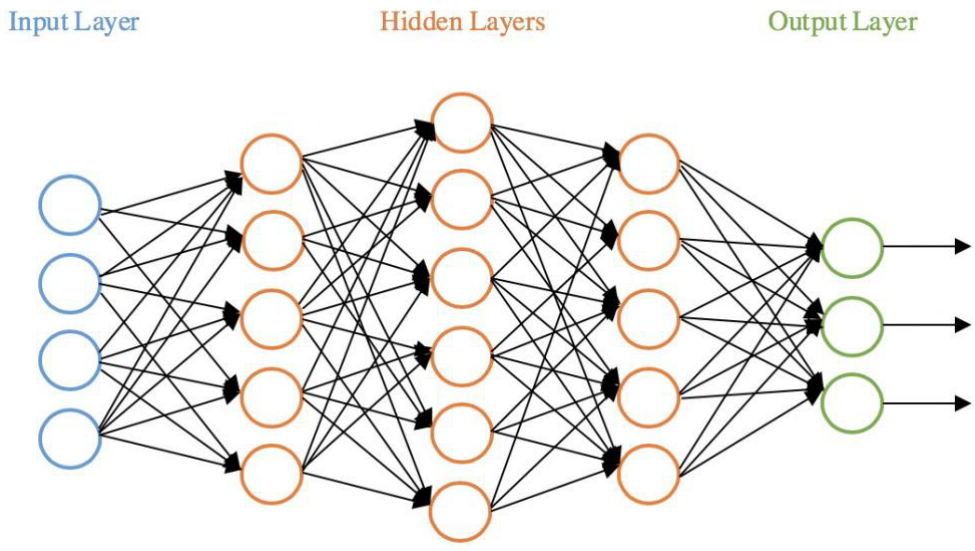

Figure 2 A simple illustration of the structure of an artificial neural network

Activation Functions

Activation functions are what introduce non-linearity to ANNs by mapping the input values to a different, smaller range, for example 0 to 1. This non-linearity is essential for the ANN to learn and make sense of complex problems where there is not a linear relationship between the inputs and outputs. This enables deep learning algorithms to compute and learn any function at all. Below are some examples of activation functions most commonly used in deep learning:

1. Sigmoid

   One of the most widely used non-linear activation function as it transforms the values between the range 0 and 1.

   Function: $f(x) = \frac{1}{1+e^{-x}}$

   Graph:

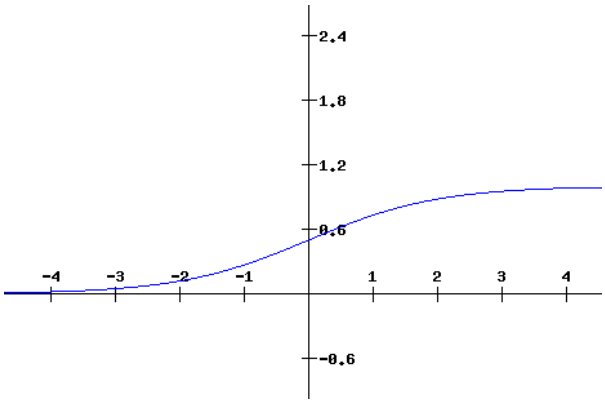

Figure 3 Sigmoid activation graph

2. ReLU (Rectified Linear Unit)

   Does not activate all the neurons at the same time so neurons are only deactivated if the output of the linear transformation is less than 0.

   Function: $f(x) = \max(0, x)$

   Graph:

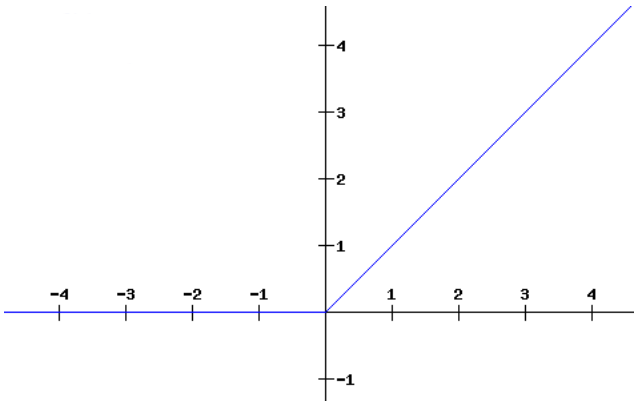

Figure 4 ReLU activation graph

3. Leaky ReLU
   Attempts to fix a common problem with ReLU called “dying ReLU” where neurons become inactive and output 0 for any input. The solution is to have a small negative slope when x < 0.
   Function: $f(x) = \max(0.01x, x)$
   Graph:

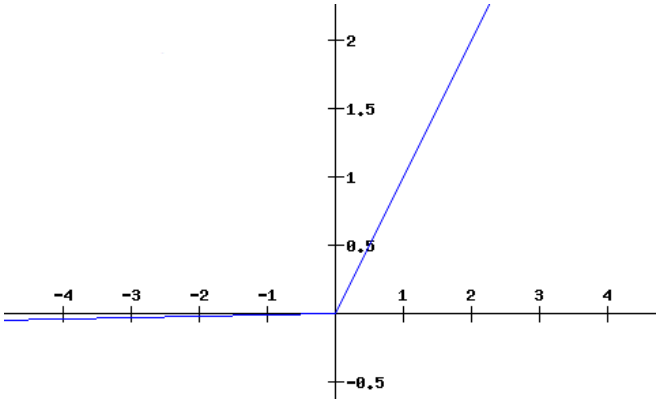

Figure 5 Leaky ReLU activation graph

Convolutional Neural Network (CNN)

A CNN is a deep learning algorithm that is designed to take images as an input, assign learnable weights and biases to parts of the image and then be able to classify them using labels. The layers that make up a CNN are a series of convolutional and pooling layers followed by a number of fully connected layers.

Convolutional Layer

A convolution is a mathematical operation to merge two sets of information.[5] A CNN uses filters to perform convolutions to be able to capture the spatial and temporal dependencies in an image. A feature map is produced by sliding the filter over the input data and performing element-wise matrix multiplication and summing the result at every location. The size of the filter can be specified but is usually 3x3. This process is shown in Figure 6.

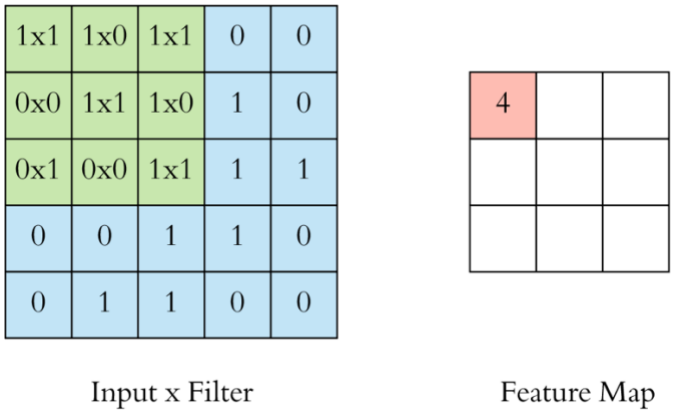

Figure 6 Visualisation of a convolution[5]

Multiple convolutions are performed on an image, each using a different filter and therefore resulting in a different feature map. If you want to have less overlap between the filters as they move across the input you can increase the stride value, which specifies by how much the filter is moved at each step. This makes the resulting feature map smaller, as shown in figures 7 and 8.

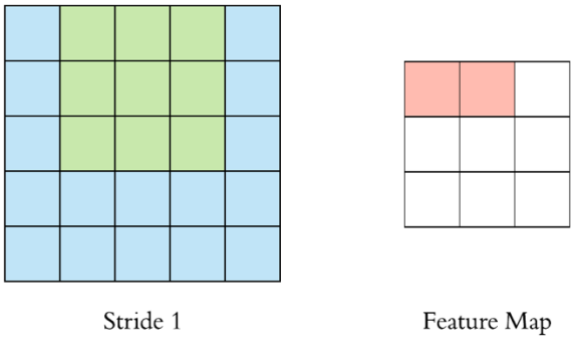

Figure 7 The second position of the filter when the stride value is 1[5]

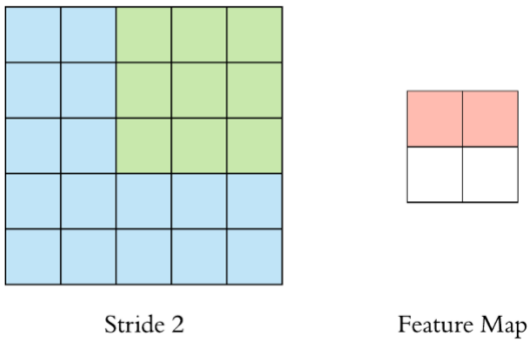

Figure 8 The second position of the filter when the stride value is 2[5]

The size of the feature map is always smaller than the input so if you want to maintain the dimensionality of the input you can use padding to surround the input with zeros, as shown in figure 9.

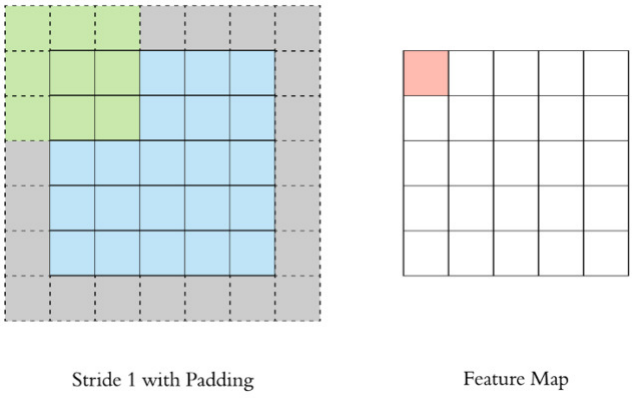

Figure 9 The effect of using padding on the feature map dimensionality[5]

The output of the convolutional layer is then all the feature maps stacked together. This result of the convolutional layer is then passed through an activation function, such as ReLU, in order to incorporate non-linearity and therefore increase the power of the neural network.

Pooling Layer

After a convolutional layer there is usually a pooling layer to reduce the dimensionality and therefore the number of parameters, which both shortens the training time and reduces the chance of overfitting. Max pooling is the most common type. It slides a window over the input and takes the maximum value in the window. Like the filter in the convolutional layer, the window size and stride can be specified.

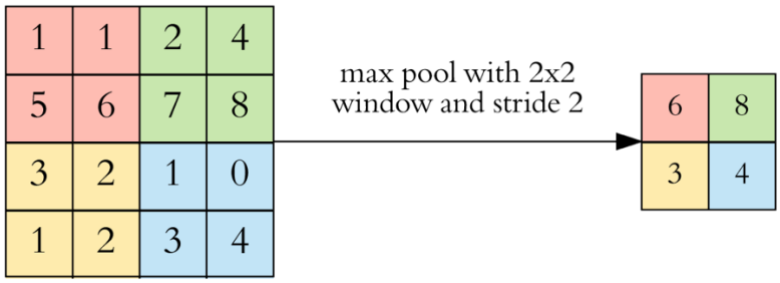

Figure 10 A visualisation of a pooling layer[5]

The main purpose of pooling is to downsize the feature map while keeping the important information. In the example in figure 10 above the resulting feature map is half the size of the input.

Fully Connected Layer

After a series of convolution layers and pooling layers there are one or more fully connected layers which take a flattened, 1D version of the output from the last pooling layer and compute class scores which map each input image to a class label. Figure 12 shows a visualisation of how all the layers of a CNN are put together to form a complete network.

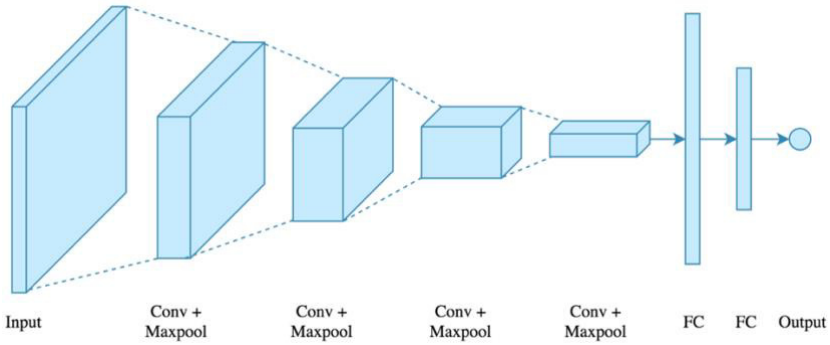

Figure 11 The layers in a CNN[5]

Training a CNN

At the start of training, the CNN will not know anything, meaning it has to learn everything from scratch. This means training large amounts of data can take a long time. More specifically, the learnable weights and filter values are randomised,[6] so the filters do not know what to look for. The way a CNN learns is through a training process called backpropagation which works backwards through the neural network to calculate the gradient of descent for weighting different variables. When training a CNN there are two important hyperparameters to be set; number of epochs and batch size. The number of epochs is the number of times the entire dataset is passed through the neural network. The higher the number of epochs, the more chance the neural network has to learn features and minimise its error, therefore the number of epochs is usually in the hundreds or even thousands. The batch size determines the number of data points to be passed through the network at once, so a lower batch size requires less memory.

CNN Performance

The performance of a CNN is usually measured by train and test accuracy. Accuracy is a metric that measures the number of predictions the model got right (true positives and true negatives).

$$Accuracy = \frac{TP + TN}{Total\ number\ of\ predictions}$$

Equation 1 Accuracy

Accuracy is measured on a scale of 0 to 1, where 1 means the model had 100% accuracy. Train accuracy relates to how well the model performed on the training data, so this should be close to 1.0 by the end of the network's training, while test accuracy relates to how well the model performed on the test data so is the real measure of how accurate the model is on unseen data. In order for accuracy to be a reliable measure of model performance the amount of data samples per class has to be balanced.

Other evaluation metrics that can tell us how well a model performs are precision and recall. Precision tells us the proportion of positive identifications that were actually correct while recall tells us the proportion of actual positives that were correctly identified. They are both calculated using a combination of true positives, false positives and false negatives.

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

Equation 2 Precision

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

Equation 3 Recall

Precision and recall are usually discussed together because there is often an inverse relationship, meaning if one is increased the other is decreased. It is therefore important to evaluate models using both in order to ensure recall is not being sacrificed for precision or vice versa. However, when it comes to developing machine learning models that use health care data, recall is often considered more important because it takes into count false negatives, which are times when the presence of disease has been missed. Such false negatives can therefore have a detrimental effect on the patient. Within healthcare it can be said that it is more important to have more false positives than false negatives, in other words it is better to err on the side of caution, and this can be represented by a lower precision but a higher recall.

One of the most effective ways to visualise the performance of a deep learning model is a confusion matrix. A confusion matrix is a table that shows the proportion of correct and incorrect predictions made for each class and which class the model predicted when its predictions were incorrect i.e. which class it got confused with. This evaluation technique therefore not only tells us how many errors are being made by the model but more importantly the type of errors. Figure 12 explains the values that are used in a confusion matrix.

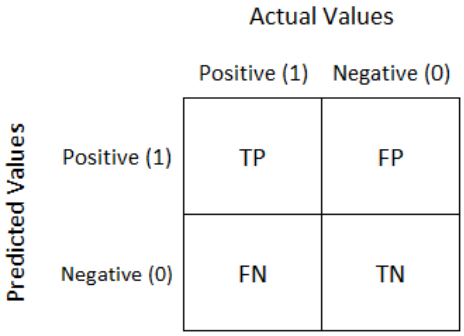

Figure 12 The values in a confusion matrix[7]

CNN Interpretability

In the context of machine learning, interpretability means the ability to explain or to present in understandable terms to humans. Deep learning techniques are known for being largely uninterpretable due to their hidden layers and are often branded as 'black boxes'[8]. This is because we cannot always explain or even identify the logic behind the computations and predictions made by deep learning models. Interpretability of CNN predictions is incredibly important in the context of clinical diagnosis using MRI scans because the clinicians who would be over-seeing the use of deep learning in the context need to be able to fully trust the CNN and its predictions. This can only be achieved by allowing them to see what is leading the CNN to make these predictions so that they can understand it for themselves. It is additionally important to understand why false negative and false positive predictions are made by the CNN so the pre-processing of the data or the model itself can be adjusted accordingly to account for these. Also, so that clinicians can look out for certain patterns that lead to false negatives or false positives and then double-check those predictions themselves. Interpretability methods can help increase the transparency, and therefore interpretability, of CNN predictions to people with no deep learning experience by creating images that highlight which areas of the input image were key influencers in the prediction made.

Interpretability Methods

Heatmaps

Heatmaps are a popular data visualisation technique that represents values as colours. They can therefore be used to increase the interpretability of the results of machine learning models by highlighting which pixels of the input images had a positive contribution to the model's predictions and which pixels had a negative contribution. One mathematical method of achieving this on convolutional neural networks is Layer-wise Relevance Propagation (LRP). LRP uses the weights and biases learnt by the network to propagate the output back through the network to the input layer where the relevance of each pixel to the output can be visualised.[9] When working backwards through the network, LRP follows this basic rule for each neuron:

$$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k$$

Equation 4 LRP rule

Where R is the relevance in the output layer, j and k are neurons of consecutive layers, a is the activation of the neuron and w is the weight between the two neurons. There are different variations of this rule that can be applied depending on the application of both the neural network and the LRP heatmap. Figure 13 shows an example of how these different variations can affect the resulting heatmap.

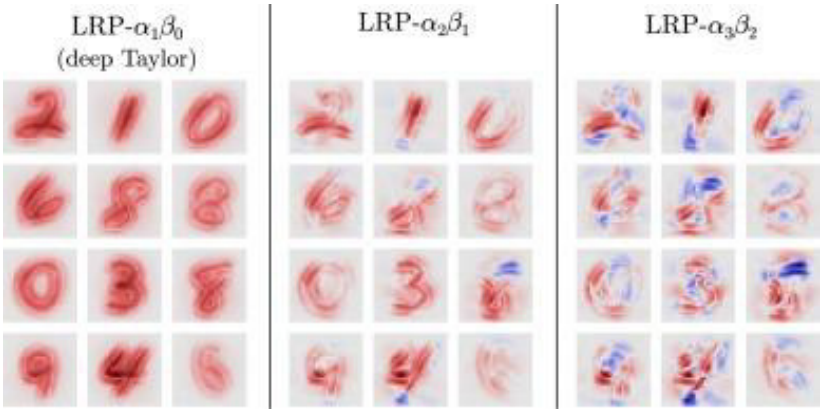

Figure 13 Example of three different LRP variations generated from a CNN trained on the MNIST dataset which aims to classify digits[10]

Heatmaps also have the advantage of being widely used so most people can understand the meaning behind them at first glance without knowing any of the scientific or mathematical background behind them.

Saliency Maps

A second interpretability method that is gaining increasing popularity when it comes to CNN interpretability is saliency maps.[11] They highlight areas of importance in the CNN's prediction by calculating how the prediction changes with respect to small changes in the input pixels and therefore how important each pixel is to the output value. This can also be explained as the gradient of the output prediction with respect to the input image.[12]

$$\frac{\partial output}{\partial input}$$

Equation 5 The gradient used in producing saliency maps

Figure 14 shows and example of a saliency map where yellow indicates a high gradient for the predicted class.

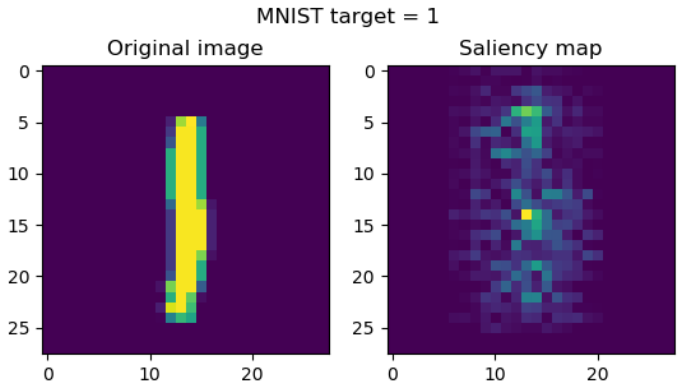

Figure 14 Saliency map example generated from a CNN trained on the MNIST dataset which aims to classify digits [13]

Lime

Lime is a toolbox which aims to 'explain what machine learning classifiers are doing'[14]. It takes a different approach to the previous interpretability methods discussed as it does not aim to understand the model as a whole but instead to understand each individual decision. This is called local model interpretability and is achieved by slightly changing the feature values of a single data point and observing the change in the predicted class. Lime then produces a number of super pixels which had the highest contribution value to the output and can be used to visualised which areas of the input resulted in the model's decision. Figure 15 shows an example of an image produced using the Lime toolbox.

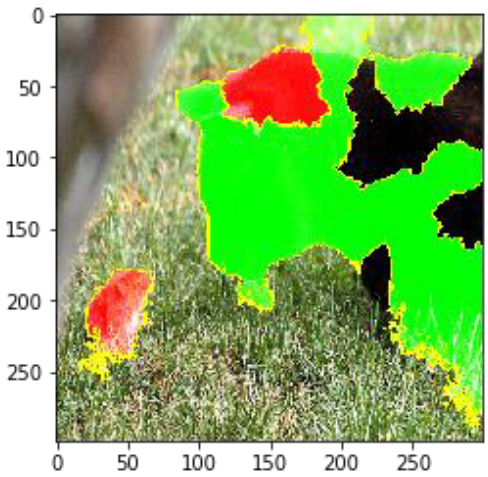

Figure 15 An example image produced using Lime which shows the super pixels that had the largest positive and negative contribution to the predicted class. The true label of the image is 'cat' and the predicted label was 'black bear'.

2.2 Datasets

Cam-CAN dataset

The cam-CAN dataset was initially used whilst waiting to gain access to the Shared Roots dataset. Cam-CAN (Cambridge Centre for Ageing and Neuroscience) is a large scale, collaborative research project that uses epidemiological, cognitive and neuroimaging data to

understand how individuals can best retain cognitive abilities into old age.[15] The dataset contains 653 brain MRIs of subjects aged 18-87 years old. Each subject has recorded attributes such as gender, age, brain measurement, whether they smoke and alcohol intake, however, for this project focus is purely on the age of the subjects. These attributes are all stored in a comma-separated value (CSV) file separate to the MRIs. The MRIs are stored as numpy arrays with a dimension of (181, 217, 181).

Shared Roots dataset

Shared Roots is a project run by Stellenbosch University that aims to understand the commonalities of neuropsychiatric disorders and modifiable risk factors for cardiovascular disease. Like the cam-CAN dataset, the Shared Roots dataset consists of brain MRIs from 288 subjects; 141 patients and 147 controls with a neuropsychiatric disorder. The MRIs are stored as numpy arrays with a dimension of (157, 189, 136)

2.3 Development Environments and Tools

Lengau Cluster

The Shared Roots data that was used is stored on a cluster that is hosted on CHPC (Centre for High Performance Computing) supercomputers. All computations that used this data were therefore be performed on this cluster. This also improved the speed of training the 3D CNN as it has a number of GPUs that could be utilised. Access to the cluster needed to be gained from researchers at Stellenbosch University and then some time was needed to learn the commands and how to navigate it.

Google Colab

Google Colab is a free cloud service that gives access to a python development environment and can be run using either a CPU, GPU or TPU. While working with the cam-CAN data, notebooks were created on Google Colab. A notebook is a file that contains cells of code that can be run independently of each other. This makes it a very good environment for experimenting with CNNs for the first time as the architecture and parameters of the CNN can be changed without having to re-load and pre-process the data again. The option to use a GPU also meant that it was possible to train a CNN in reasonable time.

Tensorflow

Tensorflow[16] is an end-to-end, open source platform for machine learning. It has a comprehensive and flexible set of tools, libraries and resources to enable high-level building of machine learning models.

Keras

Keras[17] is a high-level API written in Python for developing neural networks. It was developed with a focus on enabling fast experimentation and so suits the needs of this project. For this project Tensorflow was used as the backend for Keras.

3 APPROACH

The overall approach throughout this project was that of data analysis and exploration within machine learning. Data science projects like this cannot be architected at the start because information about the data and the best techniques to use are uncovered as the project progresses. As a result, agile methodology was employed as the project was likely to be highly iterative. This enabled changes to pre-processing methods and hyperparameter tuning to take place after the model was complete in order to improve performance.

Before working towards achieving the objectives set out in the initial plan, a new objective was set to build and train a CNN using the Cam-CAN dataset in order to learn more about deep learning implementation. Development and training using the Cam-CAN dataset was achieved using the Google Colab development environment as it is more suited to learning and experimentation. The development of both CNNs was approached in the same way. First, the dataset was explored and pre-processed so that it could be used as input into a CNN. Then the CNN was developed using Keras with a Tensorflow backend and the hyperparameters were finetuned to improve the CNN performance. In order to achieve the second initial objective of training the CNN on the Shared Roots data, the model was uploaded to the Lengau Cluster where it was trained to classify the Shared Roots data as either 'patient' or 'control'. Various network architectures were then implemented and tested in order to see which yields the best model performance and to achieve the third initial objective. Although the focus of this project is on increasing the interpretability of CNN results, a large proportion of time was spent building the CNN models in order to achieve results to which interpretability methods can be applied.

Once satisfactory results were obtained from the CNN, further investigation was completed into the three interpretability methods outlined in section 2.1. The interpretability methods were then applied to the CNN results in order to produce images that aim to increase the interpretability of the results and to achieve the fourth initial objective. Lastly, a method of quantitative evaluation was derived in order to fairly compare the interpretability images and allow conclusions to be drawn regarding how well they increase the interpretability and transparency of the model. Qualitative conclusions were also drawn as there are certain advantages to images that cannot easily be quantified.

The timeline outlined in the initial plan was kept to with only a couple of minor changes. The addition of the extra aim to develop an initial CNN to be trained on Cam-CAN data meant the time allocated to develop and train a CNN model based on the Shared Roots data had to be shared with this new aim. However, the advantage of developing this additional CNN first meant less time was needed on the Shared Roots CNN so it balanced out to mean both could be achieved within the allotted time. Additionally, a quantitative evaluation method of the interpretability toolboxes was not determined before these methods were applied to the CNN as was stated in the timeline in the initial plan. This is because it was necessary to produce the images first to be able to know in which format the data would be in and therefore what methods were suitable for quantifying the images.

The combination of an agile approach and a broad, flexible time plan allowed for successful iterative development of two CNNs with sufficient time to apply a range of interpretability methods and evaluate their effectiveness both qualitatively and quantitively.

4 IMPLEMENTATION

The implementation process of the CNN models using both the Cam-CAN dataset and Shared Roots dataset will be discussed. Then the process of generating images in order to increase the interpretability of these models and how they were evaluated will be outlined.

4.1 3D CNN Using Cam-CAN Dataset

The aim of this section was to build a 3D CNN model that can accurately classify MRI scans in the Cam-CAN dataset into predetermined age groups.

4.1.1 Pre-Processing

The first task was to load the cam-CAN data into Google Colab and pre-process it so that it could be accepted as an input by the CNN. The total size of the MRI numpy files was 8GB and Google Colab gives you a total of 12GB of RAM, so the majority of difficulties encountered at this stage were related to computations exceeding the memory allocations. The first step to a solution was to use the GPU hardware accelerator available within Google Colab. This ensured the notebook did not run out of memory when using the data however the computations still took an unreasonable amount of time considering a significant amount of experimentation would be needed. Although Google Colab has many advantages as a development environment for this project, it became clear that it is not ideal for handling large volumes of data, as if the session times out or has to restart because the RAM capacity was exceed, all the data has to be re-loaded and re-processed which is the part of building a CNN this way that takes the longest. It was therefore decided to only work with half of the data until the CNN was fully functional in order to save time. It was also decided to reduce the size of the MRI numpy arrays by 50% from (181, 217, 181) to (90, 108, 90) to make computations faster.

The labels chosen for this dataset were ‘young’, ‘middle aged’ and elderly’ with age boundaries of 18-43, 44-65, 66-80 respectively. These boundaries ensured that the spread of the data across the classes was even in order to achieve reliable CNN results. Table 2 shows how many samples belonged to each of these classes.

Class Label	Age Range	Number of Samples
Young	18-43	211
Middle Aged	44-65	225
Elderly	66-80	217

Table 2 The class boundaries used and number of data points in each class

The demographics and brain measurements of each subject were stored in a text file which was converted to a CSV file so that it could be loaded as a data frame using pandas in Colab. Each subject had a unique ‘SubCCID’ which was also the name of the file containing their MRI scan. It was therefore simple to find the age that corresponded to each MRI scan in the numpy array and then create a separate numpy array of all the class labels. As this is a multi-class problem, it was necessary to convert the labels into binary form, or one-vs-all, that is

understandable by the CNN. In order to achieve this, the LabelBinarizer function that's available as part of scikit-learn was used which replaces each label with 3 binary digits. The label 'young' is represented by [1 0 0], 'middle aged' is represented by [0 1 0] and 'elderly' is represented by [0 0 1].
At the end of pre-processing, we are left with two numpy arrays which will be accepted by the CNN. The numpy array 'X' holds the arrays of MRI scan data and has dimensions (653, 90, 108, 90). However, an extra dimension is required by Keras to be the colour channel, so an extra axis was added to give the array 'X' a final dimensionality of (653, 90, 108, 90, 1). An example slice sequence of one of the MRI scans stored in 'X' is shown in figure 16. The numpy array 'Y' holds the representations of the labels of each MRI scan and has dimensions (653, 3).

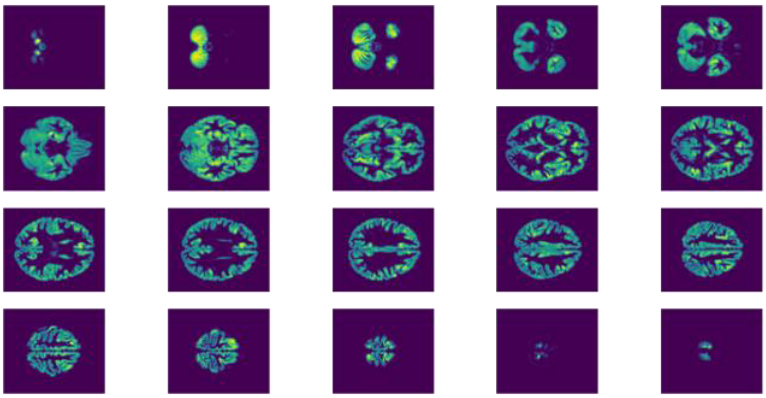

Figure 16 An example MRI scan used as input to the CNN shown as a sequence of slices

The input data then needed to be split into train data and test data. An 80/20 split was achieved using the train_test_split function from scikit-learn as shown in figure 17.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2)
```

Figure 17 The sci-kit learn function used to split input data in train and test data

The variables X_train, X_test, y_train and y_test can then be used as inputs to the CNN.

4.1.2 Convolutional Neural Network Implementation

After the pre-processing of the data, it was no more complicated to build a 3D CNN than a regular 2D CNN as Keras has equivalent functionalities and methods available. Figure 18 shows the code for the initial CNN model.

```
# Convolution layers
x = keras.layers.Conv3D(filters=32, kernel_size=(3,3,3), strides=2, padding = 'same', activation='relu')(inputs)
x = keras.layers.Conv3D(filters=32, kernel_size=(3,3,3), strides=2, padding = 'same', activation='relu')(x)

# MaxPooling
x = keras.layers.MaxPool3D(pool_size=(2,2,2))(x)

# Convolution layers
x = keras.layers.Conv3D(filters=32, kernel_size=(3,3,3), strides=2, padding = 'same', activation='relu')(x)
x = keras.layers.Conv3D(filters=32, kernel_size=(3,3,3), strides=2, padding = 'same', activation='relu')(x)

# MaxPooling
x = keras.layers.MaxPool3D(pool_size=(2,2,2))(x)

# Flatten from 3D to 1D
x = keras.layers.Flatten()(x)

# Output layer
x = keras.layers.Dense(Y.shape[1], activation='softmax')(x)
```

Figure 18 The initial network architecture used

The architecture of this model consists of 2 convolutional layers followed by a max pooling layer. This block is then repeated before a layer to flatten the data from 3D to 1D and then finally a fully connected layer to map each input to its predicted label. It was necessary to have a stride value of 2 in order for the model to run without running out of memory.
A different network architecture was then tested, called LeNet-5 architecture which is well-known for its small memory footprint and therefore fast training times[18] meaning it is well suited to this data and development environment. It consists of a convolutional layer and pooling layer, repeated twice, and then 3 fully connected layers.

```
#LeNet-5 Architecture

#Convolutional 1
x = keras.layers.Conv3D(filters=32, kernel_size=(5,5,5), strides=2, padding = 'same', activation='relu')(inputs)
#Pooling 1
x = keras.layers.AveragePooling3D(pool_size=(2,2,2))(x)

#Convolutional 2
x = keras.layers.Conv3D(filters=64, kernel_size=(5,5,5), strides=2, padding = 'same', activation='relu')(x)
#Pooling 2
x = keras.layers.AveragePooling3D(pool_size=(2,2,2))(x)

x = keras.layers.Flatten()(x)

#Fully connected 1
x = keras.layers.Dense(120, activation='relu')(x)
#Fully connected 2
x = keras.layers.Dense(84, activation='relu')(x)
#Fully connected 3
x = keras.layers.Dense(Y.shape[1], activation = 'softmax')(x)
```

Figure 19 LeNet-5 network architecture

The intention was to test the AlexNet architecture and the VGG-16 architecture as well, however, after compiling the models and seeing the number of parameters it was clear these architectures would not be feasible given the size of the data and the memory constraints of Google Colab. LeNet-5 was therefore accepted as the most suitable architecture for this network and was used for all further work using this model.
In order to train the model, 200 epochs were used with a batch size of 32 as shown in figure 20.

```
#train model
n_epochs = 200
history = model.fit(X_train, y_train, epochs=n_epochs, \
                    batch_size=32, validation_data=(X_test, y_test), verbose=0)
```

Figure 20 Code to train the model

4.2 3D CNN using Shared Roots data

Once the CNN model was finalised using the Cam-CAN data it was easily transferable to be trained using the Shared Roots data. The aim of this section was to build a 3D CNN model that can accurately classify MRI scans in the Shared Roots dataset as either patient with a neuropsychiatric disorder or control.

4.2.1 Pre-Processing

The pre-processing of the Shared Roots data was more complex than that of the Cam-CAN dataset. One reason why is because the data was stored on the cluster so all code to be run using the data had to be uploaded to the cluster as a script and then run as a job. This meant there was less possibility for trial and error. A second reason was because the MRI scans were not stored using a filename that was the same as the subject ID that was used to reference the patient in the demographics file. It was therefore more complicated to assign each MRI a label. The required files first had to be extracted from their folders and then converted from nifti files to numpy arrays and then saved together as one numpy array. The original file name for the nifti files followed a pattern of 'wsubjectID _date_ Memprage_Sag_Memprage_Sag_RMS_brain.nii' so it was possible to extract the subject ID for each MRI scan using the code shown in figure 21.

```
60  def extract_subjects_from_filenames(files):
61      IDs = []
62      for file in files:
63          splitFile = file.split('_2')
64          splitFile = splitFile[0].split('/w')
65          subID = splitFile[1]
66          IDs.append(subID)
67
68      return IDs
```

Figure 21 Code to extract the subject IDs from the filenames

The labels for each subject could then be found in the 'demographics' CSV file using the subject IDs extracted from the MRI file names. The status of each subject can be determined by the 'Patient_Control' column in this file; if it is 1 the subject is a patient with a neuropsychiatric disorder, if it is 2 the subject is a control. Figure 22 shows how the subject statuses were retrieved. The subject IDs used in the demographics folder sometimes included '-' which were not included in the file names, so these had to be ignored.

```
74  def get_subject_status(subIDs, df):
75      n = len(subIDs)
76      status = np.zeros((n, 1))
77      for ix, s in enumerate(subIDs):
78          s = s.lower().replace('-',"")
79          s = s.lower().replace('_',"")
80          df_sub = df[df['SubjectID'].str.replace('-',"")==s]
81          status[ix] = df_sub['Patient_Control']
82      return status
```

Figure 22 Code to get each subjects status

At the end of the pre-processing we are left with one numpy array holding all the numpy arrays of the MRI scans and a second numpy array holding the labels of the scans.

4.2.2 Convolutional Neural Network Implementation

A neural network with the same architecture as was used with the Cam-CAN data was implemented however, some small changes needed to be made as classifying the Shared Roots data is a binary classification problem. The activation used in the final dense layer had to be changed from Softmax to Sigmoid and the loss metric had to be changed from 'categorical_crossentropy' to 'binary_crossentropy'. Dropout was added to test for overfitting as well as lowering the learning rate. The learning rate is a hyperparameter which controls how much the weights of the network are adjusted with respect to loss.[19] The learning rate was reduced from 0.001 to 0.00001 meaning the network is more likely to find local optima but will take a longer time to converge.

4.3 Producing Interpretability Images

It was decided to apply the interpretability methods to the results of the model trained on the Cam-CAN data as it achieved a much better performance and therefore would lead to clearer, more reliable interpretability images.

It was discovered that it would not be possible to produce interpretability images using a 3D model because the toolboxes used expected a 2D model so the 3D Cam-CAN model was adapted to 2D simply by using a 2D slice of each MRI scan as inputs and then changing the Conv3D layers to Conv2D layers. The 2D slice was chosen to be position 35 out of 90 as it shows important structures such as the corpus callosum which plays a role in motor, sensory, and cognitive performances as well as grey and white matter which often change with age. An example 2D slice image is shown in figure 23.

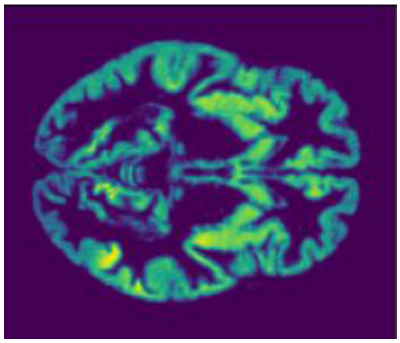

Figure 23 An example 2D slice which was used as input for the 2D CNN

The accuracy graph and confusion matrix for the 2D model can be found in the supplementary material.

The iNNvestigate toolbox[20] was first used to produce a number of LRP heatmaps using different rules for 10 of the MRI scans the network was tested on. The code to achieve this can be found in the supporting material as it is relatively long. The one that seemed best to highlight the important areas of the brain for classification by the model was chosen for further evaluation.

The Keras-Vis toolbox[21] was then used to produce a saliency image for each MRI scan the model was tested on. The code to produce a saliency image for one input image is shown in figure 24.

```
from vis.visualization import visualize_saliency
from vis.utils import utils
from keras import activations

class_idx = 0
indices = np.where(y_test[:, class_idx] == 1)[0]
# Pick a random input
idx = indices[1]

# Utility to search for layer index by name
layer_idx = utils.find_layer_idx(model, 'preds')

# Swap softmax with linear
model.layers[layer_idx].activation = activations.linear
model = utils.apply_modifications(model)

grads = visualize_saliency(model, layer_idx, filter_indices=class_idx, seed_input=X_test[idx])
plt.imshow(grads, cmap='jet')
```

Figure 24 Code to produce a saliency map using the Keras-vis toolbox

Lastly, the Lime toolbox was used to visualise the top 10 super pixels that contributed either positively or negatively to the predicted class. Figure 25 shows the code to achieve this for one data point.

```
explainer = lime_image.LimeImageExplainer()

#Get predicted classes
preds = model.predict(X_test)
pred_bool = np.argmax(preds, axis=1)

#Get explanation for the top 3 labels for a data point
explanation = explainer.explain_instance(X_test[0], model.predict, top_labels=3, /
                                         hide_color=0, num_samples=1000)

#Get top 10 super pixels for predicted class with positive and negative indication
temp, mask = explanation.get_image_and_mask(explanation.top_labels[0], positive_only=False, /
                                            num_features=10, hide_rest=False)
plt.imshow(mark_boundaries(temp / 2 + 0.5, mask))
```

Figure 25 Code to produce an image with the top 10 super pixels using the Lime toolbox

In order to evaluate and compare how well these methods increase the interpretability of the CNN, the average image for each class was generated for both the LRP heatmap and saliency map. This was achieved by calculating the average colour of each pixel for each image in each predicted class. A similar method was also applied to the super pixel maps produced by Lime. If a pixel was red or green in more than 80% of the images for each predicted class, then it would be shown as red or green in the ‘average’ image.

5 RESULTS AND EVALUATION

In this section, first the results obtained from the CNN models that used the Cam-CAN and Shared Roots data will be discussed and the performance of these models will be evaluated. Then the results from applying the three interpretability methods will be shown and evaluated.

5.1 Cam-CAN CNN Results

Accuracy, precision, recall and confusion matrices were used to evaluate the performance of different 3D CNN models trained on the Cam-CAM data set.
Figure 26 shows the accuracy graph of the model when LeNet-5 architecture was used and table 3 shows the precision and recall. Figure 27 shows the confusion matrix.

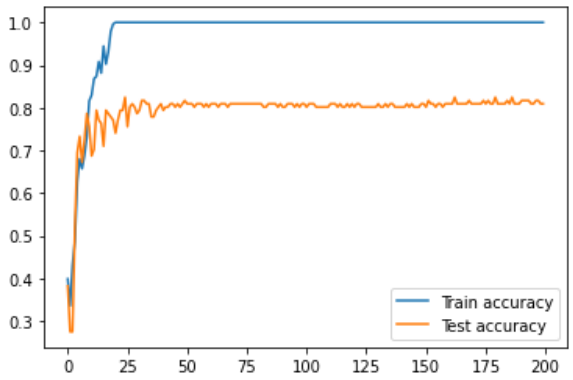

Figure 26 The accuracy graph for the CNN with LeNet-5 architecture trained on the Cam-CAN dataset

Class	Precision	Recall
Young	0.92	0.98
Middle aged	0.62	0.78
Elderly	0.89	0.68

Table 3 The precision and recall for the CNN with LeNet-5 architecture trained on the Cam-CAN dataset

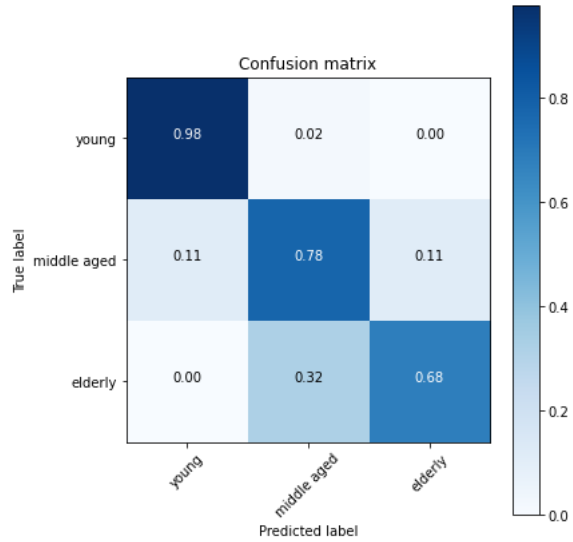

Figure 27 Confusion matrix for the CNN with LeNet-5 architecture trained on the Cam-CAN dataset

Dropout was then added to the model to test for overfitting. Table 4 shows a comparison between the accuracy, precision and recall of the model with and without dropout.

	Maximum Accuracy	Average Accuracy	Average Precision	Average Recall
Without Dropout	0.824	0.809	0.810	0.813
With Dropout	0.878	0.710	0.813	0.813

Table 4 Comparison of the effect of dropout on the accuracy, precision and recall on the CNN with LeNet-5 architecture trained on the Cam-CAN dataset

This table shows that although the maximum accuracy obtained when using dropout was higher than the maximum accuracy obtained when not using dropout, the average accuracy with dropout was significantly lower and there was no difference in average precision and recall. This shows that there was no overfitting occurring when training the model.
The confusion matrix in figure 27 shows that the errors being made by the model are always due to predicting the class next to the correct class. In particular, the model predicts 'middle aged' instead of the true label 'elderly' for 32% of data points with the label 'elderly'. The vast majority of other errors were due to predicting 'young' when the true label was 'middle aged' or predicting 'elderly' when the true label was 'middle aged'. It seems reasonable to allow the model these errors as the classes are continuous rather than distinct and have extremely close boundaries. For example, an MRI belonging to a patient aged 44 is unlikely to show significantly differences than an MRI belonging to a patient aged 43 however the former belongs to the class 'young' and the later belongs to the class 'middle aged'. It is therefore satisfactory that the model confused the two classes in this case.
It is clear from both the confusion matrix and the precision and recall values for each class that the model performed best at classifying data points as 'young', having achieved 92% precision and 98% recall. These values are particularly a success when looking at the problem from a healthcare point of view as recall is higher than precision which is more advantageous in healthcare, as discussed in section 2.1. The precision within the class 'elderly' is satisfactory at 89% however the precision for 'middle aged' is less than satisfactory at 62%. This can likely be explained due to the fact this is the middle class and so the data points at both the upper and lower ends of the boundaries of this class can easily be confused for one of the other classes.
After checking the model for overfitting and analysing the confusion matrix and precision and recall values it seems reasonable to conclude that this is the best performance that can be expected given the number of data points and the RAM available. If more RAM or GPUs were available, it would be possible to test more sophisticated architectures such as AlexNet and GoogLeNet.[22] Having more data available to train the model on would likely increase the precision and recall of the 'middle aged' and 'elderly' classes as the model would have more opportunity to learn defining features of these classes.

5.2 Shared Roots CNN Results

Accuracy graphs were primarily used when testing the model using the Shared Roots data as it did not perform as well as the model which was trained using the Cam-CAN dataset, so precision and recall are not as informative.
Figure 28 shows the accuracy curve of the model with no dropout and default learning rate of 0.001.

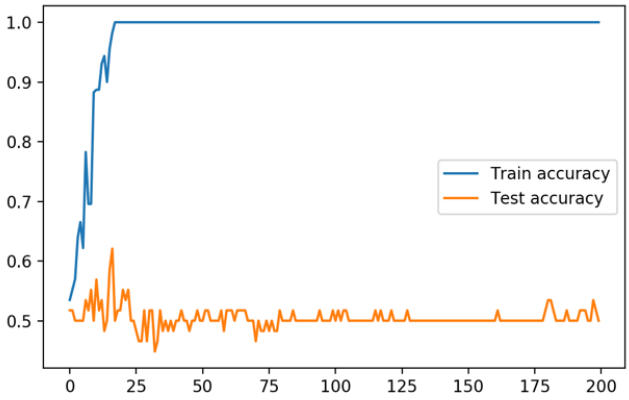

Figure 28 The accuracy graph for the CNN with LeNet-5 architecture trained on the Shared Roots dataset

This curve suggests that the model is overfitting because the train accuracy quickly reaches and plateaus at 1.0 while the test accuracy remains around 0.5 with a single peak above 0.6 which is likely a fluke. Dropout was therefore added in order to reduce the effect of overfitting.
Figure 29 shows the accuracy curve when drop out was added after the two pooling layers.

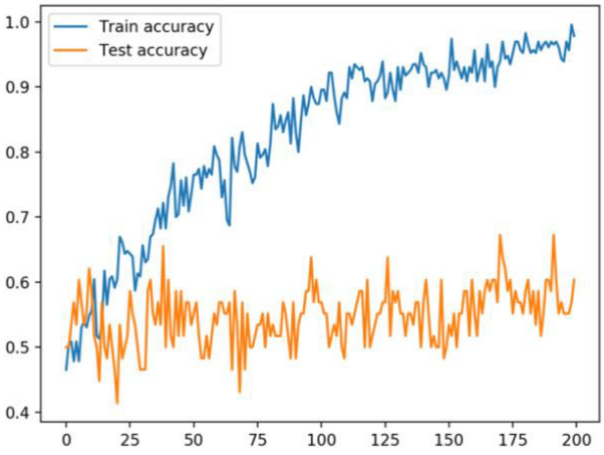

Figure 29 The accuracy graph for the CNN with LeNet-5 architecture trained on the Shared Roots dataset with dropout

The average test accuracy was increased from 0.5 to 0.6 however there is a greater fluctuation in both the train and test accuracy as a result of introducing dropout. There is a relatively small increase in test accuracy which suggests that overfitting was not the only problem stopping the network from achieving a higher accuracy. It was therefore decided to see the effect of lowering the learning rate. Table 5 compares the average accuracy for different combinations of dropout and learning rate values.

	Learning rate = 0.001	Learning rate = 0.00001
Without Dropout	0.500	0.552
With Dropout	0.603	0.500

Table 5 A comparison of the effect of reducing the learning rate on the accuracy of the CNN with and without dropout

This table shows that lowering the learning rate increased the average accuracy when dropout was not used but decreased the average accuracy when dropout was used however it is possible that this is simply by chance.
It was therefore concluded that 0.6 is the highest accuracy achievable using the Shared Roots dataset on this model. There may be multiple explanations as to why such a low accuracy was achieved when training the model on the Shared Roots data, but a relatively high accuracy was achieved when training the model on the Cam-CAN data. The simplest explanation is that there were significantly fewer data points in the Shared Roots dataset; just 288 compared to 653 in the Cam-CAN dataset. This means the network had less chance to learn defining features. Additionally, within the 'patient' class in the Shared Roots data there are multiple neuropsychiatric disorders; schizophrenia, PTSD and Parkinson's. Therefore, the network needs to learn individual features of all of these disorders and differentiate them from the control MRIs which is exceptionally hard given there are only 141 patient MRIs.

5.3 Interpretability Method Results

5.3.1 LRP Heatmaps

Figure 30 shows the results of applying various LRP methods to 3 different inputs and their predicted outputs from the CNN model trained on the Cam-CAN dataset.

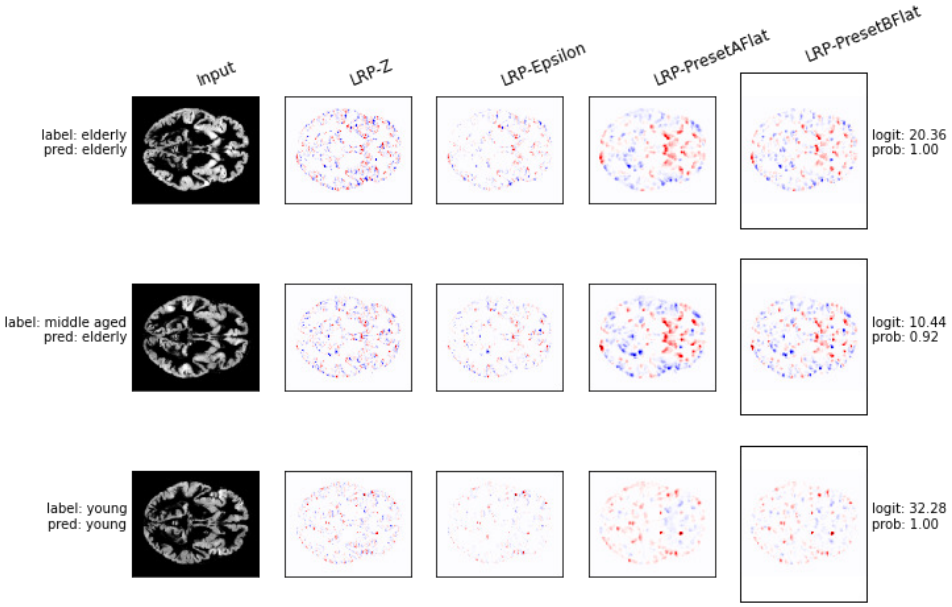

Figure 30 LRP heatmaps using different rules for 3 input images

Each LRP method uses different rules and equations when backpropagating through the network and calculating the relevance of the pixels, resulting in each map visualising the relevance of the pixels in different ways, such as at different layer depths. It was decided to analyse and evaluate LRP-PresetAFlat further as it appeared to consistently and clearly highlight which structures and areas of the brain were highly relevant to the CNNs prediction. This particular method uses α-decomposition which treats the activating and inhibiting activation factors of a layer separately, meaning the resulting map highlights pixels that positively contributed to the output class as well as pixels that negatively contributed. In this instance, pixels that are red had a positive contribution while pixels that are blue had a negative contribution. LRP-PresetAFlat also uses flat weight decomposition which produces maps that represent the relevance of pixels in higher network layers.[23]
As discussed in section 4.3, average heatmaps were produced for the three classes using the input images the CNN predicted to belong to the class. This allows us to see which areas of the brain were most important to the CNN when classifying an image into one class over the others. Figures 31, 32 and 33 show the resulting LRP heatmaps.

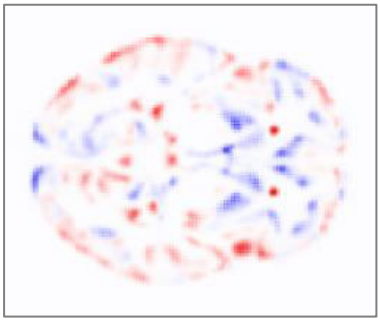

Figure 31 Average LRP heatmap *for class 'young'*

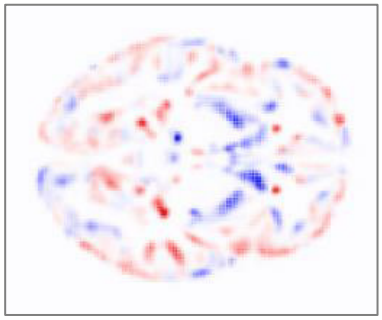

Figure 32 Average LRP heatmap for *class 'middle aged'*

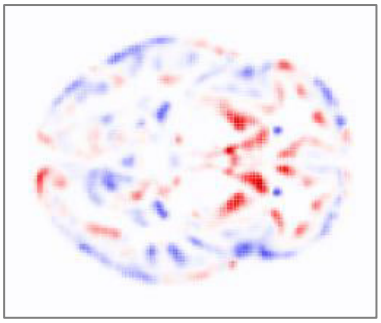

Figure 33 Average LRP heatmap *for class 'elderly'*

These heatmaps clearly show that the neural network has learnt the structure and features of the brain as well as showing which structures contributed positively or negatively to each age class.

5.3.2 Saliency Maps

Figure 34 shows saliency maps that were produced for 3 input images. The brighter the pixel, the more it contributed to the CNNs prediction.

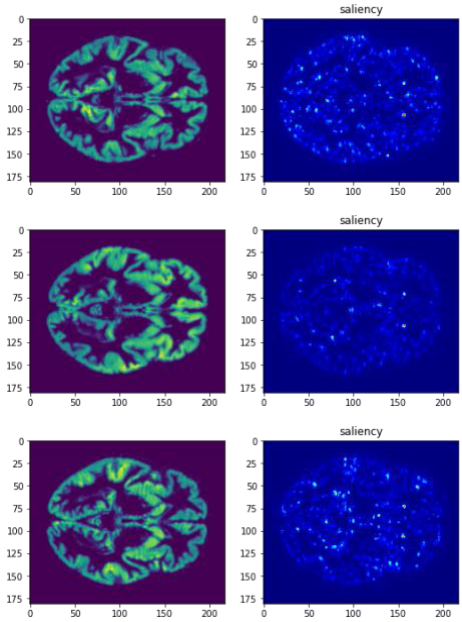

Figure 34 Saliency maps for 3 input images

Figures 35, 36 and 37 show the average saliency map for each predicted class that were generated in the same way as the average LRP heatmaps. In order to highlight the general areas that were of importance to the CNNs predictions a gaussian filter was applied to blur the saliency maps and then the map was overlaid on top of an example input from the corresponding class.

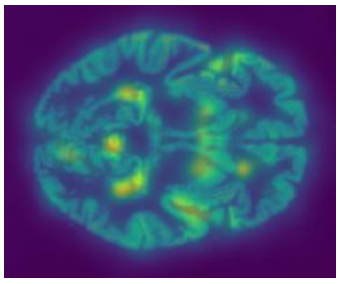

Figure 35 average saliency map *for the class 'young'*

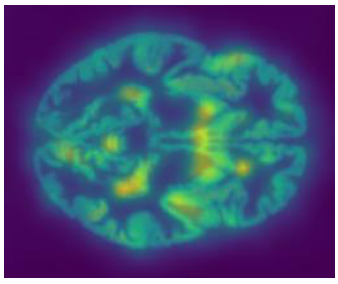

Figure 36 average saliency map *for the class 'middle aged'*

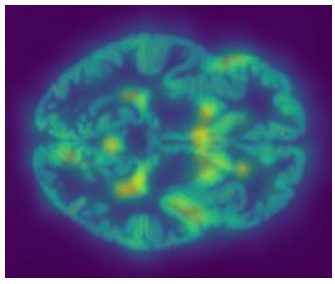

Figure 37 average saliency map *for the class 'elderly'*

5.3.3 Lime Images

Figure 38 is an example image that was produced using Lime which shows the top 10 super pixels which contributed either positively or negatively to the output class.

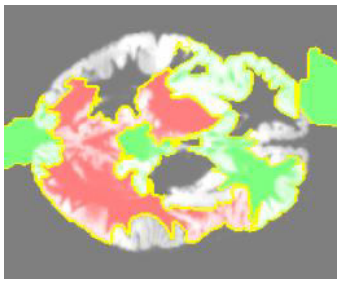

Figure 38 The top 10 super pixels that contributed to the predicted class for an input

This image does suggest that the model has learnt the edges of the brain and some of its structures. Figures 39, 40 and 41 show the resulting images from assigning each pixel as red or green only if it was red or green in more than 80% of the images in that class.

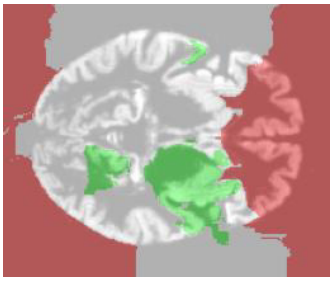

Figure 39 Top super pixels across *the class 'young'*

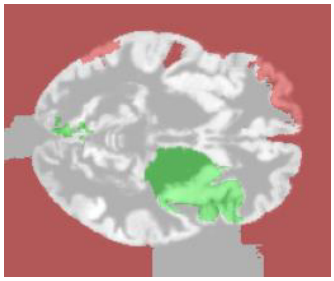

Figure 40 Top super pixels across *the class 'middle aged'*

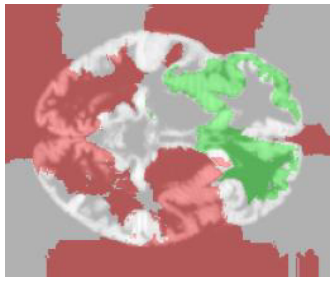

Figure 41 Top super pixels across *the class 'elderly'*

Unfortunately, a significant number of the super pixels appear outside the brain region and are instead focused within the border around the brain. It is possible this is because these regions did not play any part in the CNN prediction and so Lime has interpreted this as these areas having a negative impact on the resulting prediction.

5.4 Evaluation of Interpretability Methods

The interpretability images will be evaluated based on how they could be used by, or be useful to, clinicians as the motivation behind the project was to increase the transparency of machine learning models so that they are more likely to be accepted and used by clinicians in a healthcare environment. It is therefore important to look at these images from a clinician's point of view in order to see their true scientific value and to discuss potential further improvements. The initial aim of the project was to evaluate how well these images increased the interpretability of a CNN when diagnosing neuropsychiatric disorders. However, as mentioned in section 4.3, it was decided to instead produce interpretability images for the Cam-CAN dataset as it achieved much higher accuracy, precision and recall values and therefore the resulting images would have more significance.

5.4.1 Qualitative Evaluation

The interpretability methods all produced very different images, so it is necessary to discuss the conclusions likely to be drawn by clinicians from simply looking at the images.

The most obvious difference between the images is that the LRP heatmaps and the Lime images show whether an area contributed positively or negatively to the CNN's output whereas the saliency maps do not differentiate this factor. This allows the clinician to clearly see the areas that the CNN learnt were different between the age groups. For example, the LRP maps clearly show that the anterior horns of the lateral ventricle had significant relevance to the CNN when classifying MRIs as 'elderly' because the pixels in this area were consistently displayed as red when the predicted class was 'elderly', meaning they had a strong positive contribution. Meanwhile, the same pixels were consistently displayed as blue when the predicted class was 'young' or 'middle aged', meaning they had a negative contribution. This tells clinicians that the CNN learnt that differences in this area is one of the factors that separates the three classes. This is in line with the known effects of ageing on the brain as it has been found that the volume of the lateral ventricles increases with age.[24]

However, the LRP heatmaps and Lime images do not clearly show which pixels and areas were more significant than others because varying intensity of colours is not utilised as it is in the saliency maps. In fact, LRP appears to show the contribution of all edges learnt by the CNN but the lack of varying intensity of colour means it is not possible to see which areas were the most important. In the saliency maps it is clear which areas were more relevant in each of the classes. For example, the anterior horns of the lateral ventricle are brighter in the 'middle aged' and 'elderly' classes which can tell clinicians that they had higher relevance to the CNN when classifying an MRI into one of these classes than they did when classifying an MRI as 'young'. The use of varying brightness can also highlight to clinicians which areas of the brain were more important than others within the classes. This can then allow them to focus their attention to the areas that had higher relevance. For example, in the average saliency map for the class 'young' it is clear that structures towards the back of the brain, such as the choroid plexus, had higher relevance than the anterior horns of the lateral ventricle when the CNN was classifying an image into this class as this area has brighter pixels.

Looking at the images in broader terms, it can be said that both the LRP heatmaps and saliency maps increase the transparency and interpretability of the CNN as even at first glance it is clear which specific areas and structures of the brain had some level of relevance to the CNN when making its predictions. The average Lime images produced show that images produced using this toolbox are of lesser quality and achieve increased interpretability to a lower level due to a large proportion of highlighted areas being outside of the brain region. This problem does not occur when the LRP or saliency methods are used. This has a large impact on the interpretability of the images because it means that less structures within the brain are highlighted and it also can draw attention away from the highlighted areas within the brain, meaning they are not a very effective tool for clinicians.

5.4.2 Quantitative Evaluation

In order to quantitatively evaluate how successfully the individual methods are at increasing the interpretability of the CNN, a correlation value was obtained for each method. These values tell us how visually clear it is that the change in relevance of a certain area effected the CNN's prediction. The idea behind this method of evaluation was that clinicians could use a similar tool to investigate how the relevance of a small area of the brain they are interested in changed within the CNN in relation to age. The area chosen for this evaluation was a rectangular section that contains the anterior horns of the lateral ventricle as this was highlighted by all three interpretability methods as having some relevance of significance to the CNN when classifying the MRIs. Figure 42 highlights this section on one of the input MRIs.

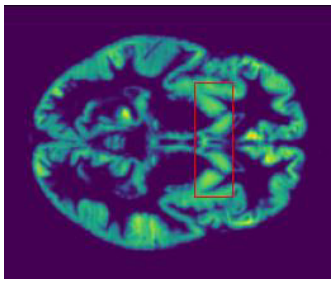

Figure 42 An example input image with the section used for quantitative evaluation highlighted by the red box

Each interpretability method was applied to each input MRI and then the resulting images were cropped down to the area shown above. The images were then divided into the three classes that they were predicted to belong to by the CNN. The number of certain pixel values in this area in each age class were then used to calculate the correlation value.
The interpretability images were all generated from the same CNN model trained on the same data, meaning if the images were equally as good at increasing the interpretability of the CNN, the correlation values should be relatively close. Therefore, the correlation values obtained can be used to evaluate the different methods in comparison to one another.

LRP

Unfortunately, the pixels in the LRP images were stored as RGB values rather than values relating to the relevance of the pixel. To overcome this, the maximum value in the RGB array was calculated and its corresponding colour was recorded as the colour of that pixel, where red represents a positive contribution and blue represents a negative contribution. White pixels represent no contribution so were not included in the calculations. The number of red and blue pixels present in each image of each class were counted and this information was stored in a pandas DataFrame containing the class label of each image (0 for 'young', 1 for 'middle aged' or 2 for 'elderly'), the red pixel count and the blue pixel count. It was then possible to calculate the correlation between the label and the number of each colour pixels and the p-values. Table 6 shows the results.

	Correlation Value	p-value
Red pixel count	0.880	$1.838e^{-43}$
Blue pixel count	-0.880	$1.838e^{-43}$

Table 6 The correlation values and p-values obtained from the LRP heatmaps

Using a significance level of 0.05 for the p-values, the null hypothesis that there is no relation between the number of red or blue pixels in the area and the age class predicted by the CNN can be rejected. This means the strong positive correlation between the red pixel count and the label tells us that the greater the number of red pixels in this area, the more likely the predicted class is to be 'elderly' while the strong negative correlation between the blue pixel count and the label tells us that the greater the number of blue pixels in this area, the more likely the predicted class is to be 'young'. Not only does this increase the transparency of the CNN by proving that the anterior horns of the lateral ventricle play an important role in the CNN's decision no matter what the predicted class is, it also proves a level of consistency in the way the LRP heatmaps present which areas of the brain were relevant to each predicted class. This consistency is important in enabling clinicians to understand the decisions made by the CNN and to trust the predictions it makes.

Saliency

The pixels in the saliency images were stored as single numerical values between 0 and 1 which represent the relevance of that pixel on the predicted class. It was decided to measure the correlation between the average pixel value of each cropped image and the predicted class. The average pixel value of the image will be greater if there are more 'brighter' pixels in the image which would mean that the selected area of the brain was of high importance to the CNN when making its prediction. Table 7 shows the correlation value and p-value obtained.

	Correlation Value	p-value
Average pixel relevance	0.576	$6.001e^{-11}$

Table 7 The correlation values and p-values obtained from the saliency maps

Once again using a significance level of 0.05 for the p-values, the null hypothesis that there is no relation between the average relevance of the pixels in the area and the age class predicted by the CNN can be rejected. Although this is not as strong of a correlation as was achieved with the LRP heatmaps, it does tell us that the higher the number of bright pixels in this area, the more likely the CNN is to predict the label as 'elderly'. This means that the saliency maps aren't quite as effective as the LRP heatmaps in accurately displaying the difference in the relevance of this area of the brain but they do still increase the transparency and interpretability of the CNN by making it visually clear to some extent that the anterior horns of the lateral ventricle have higher relevance when the predicted label is 'elderly' than when it is 'young'.

Lime

The Lime images were stored using only three possible pixel values; 1 for the colour green, -1 for the colour red and 0 for no colour. Therefore, the same method that was used to calculate a correlation value for the LRP images was applied. In the Lime images, however, green pixels represent a positive contribution and red pixels represent a negative contribution. The number of green and red pixels present in each cropped image were counted and then the

correlation between these counts and the corresponding predicted labels was calculated. Table 8 shows the correlation values and the p-values.

	Correlation Value	p-value
Green pixel count	0.400	$2.280e^{-06}$
Red pixel count	-0.252	0.004

Table 8 The correlation values and p-values obtained from the Lime images

As with the LRP and saliency correlations, using a significance level of 0.05 for the p-values, the null hypothesis that there is no relation between the number of green or red pixels in the area and the age class predicted by the CNN can be rejected. Although the null hypothesis can be rejected, the correlations are weak, meaning the relationship between the pixel counts and the predicted label is close to negligible. As the Lime images were produced using the same information as was available to the LRP heatmap method, no blame can be given to the CNN itself for such a difference in correlation values between the Lime images and the LRP heatmaps. We can therefore conclude that the Lime images are not as successful in visually increasing the interpretability of the CNN as the anterior horns of the lateral ventricle do not have differencing levels of relevance within these images like they do within both the saliency maps and the LRP heatmaps.

5.4.3 Evaluation Conclusions

After qualitatively and quantitatively evaluating the three interpretability methods, it is possible to conclude that LRP heatmaps are the best method for increasing the transparency and interpretability of CNNs for clinicians. This is due to them showing positive and negative contributions that are consistent within the predicted classes but differing across the predicted classes. This allows clinicians to be confident in the conclusions they can draw when using the heatmaps as a tool to understand which areas of the brain were important to the CNN when making its decision.

Saliency maps are also an effective method for increasing interpretability, but they contain less information than the LRP heatmaps so fewer conclusions can be drawn by clinicians when using them as a tool to understand the CNNs predictions. They are, however, effective at informing clinicians which areas of the brain were more important than others which the LRP heatmaps are lacking.

It can also be concluded that Lime images are the least effective at increasing the interpretability of the CNN, mainly due to the majority of the highlighted areas falling outside of the brain in the average images produced. This means even fewer conclusions about the CNN can be drawn by clinicians using these images.

Although the methods were not applied to MRIs of neuropsychiatric disorders, the results from this evaluation are proof of concept that LRP and saliency methods could be applied to this form of data to obtain interpretable results about the CNNs decision making process.

6 FUTURE WORK

6.1 Shared Roots CNN Optimisation

Although the aims of the project have been met, the later aims relating to the interpretability of a CNN were not achieved using the intended dataset. This was due to a CNN trained on the Shared Roots data not achieving an accuracy level higher than 60%. Therefore, any future work should first be focused on trying to improve the performance of this model. This would involve potentially weeks of hyperparameter tuning and experimenting with different architectures. It would also be necessary to check whether the task is definitely computationally possible as it may be the case that there are not enough consistent structural differences within the brain for a CNN to be able to classify MRIs of patients with a neuropsychiatric disorder from MRIs of control subjects. It should also be tested whether changing the prediction classes would improve accuracy. For example, by aiming to classify the MRIs into their exact neuropsychiatric disorders; Parkinson's, PTSD and schizophrenia, rather than into patient or control.

6.2 Apply Interpretability Methods to Shared Roots CNN

If an accuracy level of greater than 80% is achieved, then we can confidently assume that the CNN has successfully learnt the features of the brain that separate these disorders from healthy controls. The interpretability methods that were explored in this project can then be applied to such CNN trained on the Shared Roots data as it has been shown that they are all successful in increasing the transparency and interpretability of CNNs to at least a small extent. The results of this would have greater clinical applications than the results achieved using the Cam-CAN age dataset as diagnosing neuropsychiatric disorders is a more important clinical task than predicting age. If it can be shown that such methods can increase the interpretability of a CNN in this context that is one step towards deep learning models being trusted and accepted by clinicians and consequently being used for diagnosis in clinical settings.

6.3 Further study of Interpretability Methods

It would also be beneficial to carry out further study and explorations of the interpretability methods to ensure that they are being used to their full potential. This would involve experimenting with different variations of the methods, such as the many different LRP variations available. A large-scale study could then be carried out in order to evaluate and compare these methods in a clinical setting. Further quantitative evaluation of both the CNN and the interpretability methods could be achieved by asking clinicians to highlight areas of specific MRIs they would primarily look at when diagnosing the patient and calculate the percentage of regions the interpretability methods also highlighted as being important to the CNN when making its prediction. This would tell us how effective and accurate the CNN and interpretability methods are compared to clinicians and therefore whether including them in the diagnostic process would be of significant benefit.

7 CONCLUSIONS

There were initially two aims of this project, with an additional aim added at the start of the implementation. These are reiterated below.

- Develop a CNN to classify MRI scans into age groups
- Develop a CNN to classify MRI scans of neuropsychiatric disorders
- Evaluate the interpretability of a range of Python libraries for interpreting deep learning models

The first CNN which was trained on the Cam-CAN dataset was developed successfully, having obtained greater than 80% accuracy, precision and recall.
The second aim was met in the sense that a CNN was developed and trained on the Shared Roots data, however it did not achieve a satisfactory performance, having accuracy of only 60%. Although there was no explicit aim to achieve a certain accuracy, it was necessary to achieve a performance level where valuable information could be extracted via the use of interpretability methods and this level was not achieved.
The third aim was also met but it was not achieved using the Shared Roots data as was originally planned. However, it was still possible to evaluate the three interpretability methods initially outlined using the Cam-CAN dataset. There were no expectations of how well the methods would successfully increase the interpretability of the CNN, however based on the results achieved, it is reasonable to say that the methods did achieve this to a reasonably high extent. The evaluation of the methods showed that LRP was the most successful at clearly highlighting the areas of the brain that had high relevance to the CNN when making its predictions but also it was successful at highlighting the differences in important areas between the classes. It was also shown that saliency maps increased the transparency of the CNN significantly by visually drawing attention to the areas of the brain which consistently contributed to the CNN's predictions. Lime was a method very different to LRP and saliency and it was discovered that it may not be as suitable as the others for this application. The Lime method highlighted the area in the input image around the brain more than it highlighted areas within the brain and it was not successful at highlighting the areas that were consistently relevant to the CNN, as was shown by the weak correlation value obtained. These conclusions can be drawn as a result of both quantitative and qualitative evaluation of the three interpretability methods, therefore successfully obtaining the third and final aim of the project. This project can be used as proof of concept that interpretability methods can be used to increase the transparency and interpretability of deep learning models and it can confidently be assumed that the same would be true if the methods were applied to MRIs of neuropsychiatric disorders.

8 REFLECTION OF LEARNING

I am pleased not only with the outcome of my project but also with all the skills, both technical and soft, that I have learnt along the way. At the start of the project I had no practical experience with deep learning, but I now feel confident in programming a CNN and believe it would be possible to apply the knowledge I have gained to any other machine learning or deep learning task. I have also gained a deeper understanding of the maths and technology behind neural networks which will be incredibly valuable as I continue working with deep learning in the future. Although interpretability of deep learning models is a fairly niche subject, I believe it holds the key to deep learning models being used for everyday diagnostic classification, which is a milestone I am passionate about being achieved. I therefore believe the knowledge about this subject I have gained from this project will give me a great advantage when undertaking further research in the area in the future.

This experience also gave me invaluable insight into the data science research process, most importantly how to stay motivated throughout. I learnt early on that data exploration and development of machine learning models is not a linear process and it is inevitable to be faced with many unexpected problems that need to be overcome. It is therefore important not to get disheartened when you have spent a large amount of time on a project but don't have much to show for it yet. I found that even after a long period of time of obtaining no useful results, things would eventually start to come together, and meaningful results would be obtained in the end.

I have also been able to build and improve upon my soft skills significantly, with this being the first large data science report I have written. I have learnt the correct basic structure of such reports and the tone that they should be written in. Time management was another important skill I was able to utilise and build upon. As this was such a large project it was important to set small, achievable goals along the way. I also had to make important decisions regarding when to move on from trying to improve the accuracy of my CNNs in order to focus on other parts of the project and to meet deadlines. This was always a tough decision as it is very tempting to keep making small changes to hyperparameters or even testing potentially endless combinations of layers in order to see the effects on the model's performance but it was important for me to learn when it was best to move on to other tasks in relation to the amount of tasks and time remaining.

Overall, I believe I handled the hurdles and difficult decisions I was faced with both professionally and successfully and am therefore proud with not only the final product of my project, but also the knowledge and skills I learnt throughout the project.

9 REFERENCES

[1] Naqvi, E., *Parkinson's Disease Statistics*. [online] Parkinson's News Today. Available at: https://parkinsonsnewstoday.com/parkinsons-disease-statistics/

[2] Mentalhelp.net. *Schizophrenia Symptoms, Patterns And Statistics And Patterns*. [online] Available at: https://www.mentalhelp.net/schizophrenia/statistics/

[3] Knott, D., *Post-Traumatic Stress Disorder - PTSD. PTSD Info*. [online] Patient.info. Available at: https://patient.info/mental-health/post-traumatic-stress-disorder-leaflet/

[4] Parkinsons.org.uk. *What Is Parkinson's? | Parkinson's UK*. [online] Available at: https://www.parkinsons.org.uk/information-and-support/what-parkinsons

[5] Dertat, A., 8 November 2017. *Applied Deep Learning - Part 4: Convolutional Neural Networks*. [online] Medium. Available at: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

[6] Deshpande, A., *A Beginner's Guide To Understanding Convolutional Neural Networks*. [online] Available at: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

[7] Narkhede, S., 9 May 2018. Understanding Confusion Matrix. [online] Medium. Available at: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

[8] Pandey, P., 2 April 2019. *Interpretable Machine Learning*. [online] Medium. Available at: https://towardsdatascience.com/interpretable-machine-learning-1dec0f2f3e6b

[9] Lindwurm, E., 15 December 2019. *Indepth: Layer-Wise Relevance Propagation*. [online] Medium. Available at: https://towardsdatascience.com/indepth-layer-wise-relevance-propagation-340f95deb1ea

[10] Montavon, G., Samek, W. and Müller, K., 24 June 2018. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73, pp.1-15.

[11] Stewart, M., 19 March 2020. *Guide To Interpretable Machine Learning*. [online] Medium. Available at: https://towardsdatascience.com/guide-to-interpretable-machine-learning-d40e8a64b6cf

[12] Raghakot.github.io. *Saliency Maps - Keras-Vis Documentation*. [online] Available at: https://raghakot.github.io/keras-vis/visualizations/saliency/

[13] MachineCurve. *Visualizing Keras CNN Attention: Saliency Maps – Machinecurve*. [online] Available at:

https://www.machinecurve.com/index.php/2019/11/25/visualizing-keras-cnn-attention-saliency-maps/

[14] Marco Tulio Ribeiro and Sameer Singh and Carlos Guestrin, 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016* (pp. 1135–1144).

[15] Cam-can.org. *Cam-CAN*. [online] Available at: https://www.cam-can.org

[16] TensorFlow. *Tensorflow*. [online] Available at: https://www.tensorflow.org

[17] Keras.io. *Keras Documentation*. [online] Available at: https://keras.io

[18] Rosebrock, A., 1 August 2016. *Lenet - Convolutional Neural Network In Python - Pyimagesearch*. [online] PyImageSearch. Available at: https://www.pyimagesearch.com/2016/08/01/lenet-convolutional-neural-network-in-python/

[19] Zulkifli, H., 21 January 2018. *Understanding Learning Rates And How It Improves Performance In Deep Learning*. [online] Medium. Available at: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10

[20] Alber M., and contributors, 13 August 2018. iNNvestigate neural networks! [online] Available at: https://arxiv.org/pdf/1808.04260.pdf

[21] Kotikalapudi, R., and contributors, 2017. keras-vis. https://github.com/raghakot/keras-vis.

[22] Das, S., 16 November 2017. *CNN Architectures: Lenet, Alexnet, VGG, Googlenet, Resnet And More*. [online] Medium. Available at: https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

[23] Lapuschkin, S., Binder, A., Montavon, G., Muller, K. and Samek, W., 17 November 2016. *LRP Toolbox For Artificial Neural Networks 1.2.0 – Manual*. [online] Available at: http://heatmapping.org/files/lrp_toolbox/documents/manual.pdf

[24] Parija, Bijayalakhmi & Sahu, Niranjan & Rath, Shakti & Padhy, Rabindra. (2014). Age-related changes in ventricular system of brain in normal individuals by computed tomography scans. *Siriraj medical journal.* 66. 225-230.