



# **Cloud-Based System for detection of Alzheimer's Disease using Deep Learning on MRI Images**

Josh Crotty

20096881@mail.wit.ie

Department of Computer Science and Mathematics

B.Sc. (Hons) Software Systems Development

Supervisor: Dr. Bernard Bulter

*Submitted in partial fulfilment of the requirements for the degree of B.Sc. (Hons) Software  
Systems Development*

South East Technological University

February 13, 2025

# Contents

<b>I. Plagiarism Declaration</b>	<b>3</b>
<b>II. Abstract</b>	<b>4</b>
<b>1 Introduction</b>	<b>5</b>
1.1 Background . . . . .	5
<b>2 Research Objectives</b>	<b>6</b>
<b>3 Literature Review</b>	<b>7</b>
<b>4 Dataset</b>	<b>10</b>
4.1 Image Preprocessing . . . . .	11
<b>5 Methodology</b>	<b>12</b>
5.1 Software System Stack . . . . .	13
5.2 Workflow . . . . .	14
5.3 Model Architectures . . . . .	15
5.3.1 ResNet-50 initial setup & model implementation . . . . .	16
5.4 Models Evaluation and Performance . . . . .	17
5.4.1 Evaluation Metrics . . . . .	17
5.4.2 Performance of proposed models . . . . .	19
<b>6 Tools and Technologies</b>	<b>20</b>
6.1 Part One - Experimentation . . . . .	20
6.2 Part Two - Deployment . . . . .	21
<b>7 Project Plan</b>	<b>22</b>
7.1 Semester One . . . . .	22
7.1.1 Gantt Chart . . . . .	22
7.2 Semester Two . . . . .	24
7.2.1 Gantt Chart . . . . .	24
7.2.2 Homepage Wireframe . . . . .	25
7.2.3 Patient Page Wireframe . . . . .	25
7.2.4 Personas and Use Cases . . . . .	26
7.3 Challenges and Mitigation Strategies . . . . .	27
<b>8 Conclusion</b>	<b>28</b>
<b>9 Bibliography</b>	<b>29</b>
<b>10 Appendix A</b>	<b>31</b>
<b>11 Appendix B</b>	<b>33</b>

## List of Figures

1	Unprocessed middle slice from each class label in the axial plane. . . . .	10
2	A sample of an MRI slice in three planes (axial, coronal, and sagittal) with preprocessing steps applied. (Row 1) Original MRI; (Row 2) N4 Bias Correction; (Row 3) Affine Transformation; and (Row 4) Skull Stripping. . . . .	11
3	Frameworks and System Architecture that is being used for data analysis and training for this project. . . . .	13
4	Diagram of the deep learning system workflow for experimentation and training. . . . .	14
5	ResNet-50 Model Architecture . . . . .	15
6	Convolutional and Identity Blocks in ResNet-50. These building blocks use skip connections to allow for deeper networks which avoids the degradation problem. . . . .	15
7	ResNet-50 loss curves and accuracy . . . . .	19
8	Workload schedule for Semester One, showing weekly task allocations and milestones. . . . .	22
9	Workload schedule for Semester Two, showing weekly task allocations and milestones. . . . .	24
10	Homepage wireframe for the proposed web application. . . . .	25
11	Patient page wireframe for the proposed web application. . . . .	25
12	User persona of Dr. Carter who specialises in Alzheimer’s Disease and progression. . . . .	26
13	User persona of Ms. Sarah Foley who specialises in Medical Imaging. . . . .	26
14	User persona of Mr. Joe Blogs with ongoing progression of Alzheimer’s Disease. . . . .	26
15	Initial EDA Analysis Figures. . . . .	31
16	Evaluation Metrics ResNet-50. . . . .	32

## List of Tables

1	Details of the ADNI dataset. CN: cognitively normal; MCI: mild cognitive impairment; AD: Alzheimer’s disease. . . . .	10
2	Comparison of model evaluation metrics and ResNet-50 hyperparameters . . . . .	19

## List of Listings

1	ResNet-50 model initialisation using MONAI and device agnostic setup. . . . .	16
2	ResNet-50 optimiser and loss function setup code. . . . .	16
3	ResNet-50 dataloader and transforms python code. . . . .	33
4	ResNet-50 training loop code. . . . .	34
5	ResNet-50 test loop code. . . . .	35

## I. Plagiarism Declaration

I hereby certify that this assignment is all my own work and contains no plagiarism. By submitting this assignment, I agree to the following terms: Any text, diagrams or other material copied from other sources have been clearly acknowledged and referenced as such in the text by the use of citations including the author's name and date of publication [e.g. (Byrne, 2008)] in the text proceeding the referenced material. These details are then confirmed by a fuller reference in the bibliography. I have read the sections on referencing and plagiarism in the handbook or in the SETU Quality Manual and I understand that only assignments which are free of plagiarism will be awarded marks. I further understand that SETU has a plagiarism policy which can lead to the suspension or permanent expulsion of students in serious cases (SETU, 2023).

Signed: 

Date: February 13, 2025

## II. Abstract

This project is suggesting a use case for a system for the identification of Alzheimer's Disease (AD) that could be deployed within a cloud environment. Having such a system in place could help with the diagnosis and potentially speed up the procedure of identifying cognitive impairment within patients' MRI brain scans. Apart from this, the project also proposes the effects of data augmentation and transformations applied to training data and the effects it has on evaluation metrics for image classification. Different model architectures will be compared based on evaluation metrics for the task of image classification of (AD) and also the effects of using pre-trained weights from medical and non-medical domains.

A deep learning environment is proposed for the experimentation of different model architectures and results will be compared using a cross validation method to give a more robust metric of accuracy. Also, the project gives details on the deployment options that are available for the trained models which will be utilised in Semester Two where most of the development will take place.

# 1 Introduction

Alzheimer's Disease is a neurological disease that impairs the neural connections and pathways in the brain. It is caused by the abnormal build-up of proteins in and around brain cells. One of the proteins involved is called amyloid, which form deposits of plaque around brain cells, the other protein is called tau, deposits of which form tangles within brain cells [1]. This protein accumulation disrupts cellular function and results in a decrease in neurotransmitter chemicals, which are essential for effective communication between neurons.

Identifying Alzheimer's Disease (AD) symptoms early on is essential for patient screening, diagnosis, and care as it offers patients the opportunity to plan for the future and make appropriate lifestyle changes that could help maintain their quality of life for longer. Early detection of Alzheimer's disease can be difficult in clinical practice due to a number of factors, such as time constraints for clinicians, the difficulty of accurately diagnosing Alzheimer's pathology, and the tendency of patients and healthcare providers to dismiss symptoms as a normal part of ageing [2].

Recently, there has been growing research in applying Machine Learning and Deep Learning techniques to medical imaging. Specifically, advanced imaging modalities such as MRI, PET, and X-ray are being utilised to enhance the classification and identification of diseases. Deep Learning models, in particular, have demonstrated remarkable results, especially through the use of popular architectures like Convolutional Neural Networks (CNNs) [3]. These models leverage their ability to automatically learn hierarchical features from imaging data, significantly improving diagnostic accuracy and efficiency. This project aims to combine both, MRI medical imaging of AD, CN patients with Deep Learning models in order to correctly classify the correct labels for the images.

## 1.1 Background

Compared to countries like the United States, where healthcare is heavily integrated with technology and supported by a robust infrastructure and funding, Ireland's healthcare system has lagged in technological advancements. A study by the Economic and Social Research Institute (ESRI) in 2021 highlighted significant gaps in Ireland's Health Information Systems (HIS) and data infrastructure. Although initiatives like eHealth Ireland [4] have made strides in digital healthcare transformation, they are still limited by funding. In 2021, ESRI estimated that less than 0.8 percent of Ireland's public health budget was allocated to eHealth and health technologies [5]. This figure is considerably lower than in peer nations, where health technology budgets can reach up to 3 percent of total healthcare spending. The funding shortfall inhibits Ireland's capacity to fully leverage health technologies for more efficient patient care and accurate diagnostics, which are critical in areas like Alzheimer's disease diagnosis. Alzheimer's disease (AD) poses a significant challenge to the healthcare sector because early and precise diagnosis is crucial for patient care and planning because symptoms can develop subtly and progress rapidly, timely intervention enables patients to seek early intervention [6]. However, due to the limited investment in healthcare technologies in Ireland, resources for diagnostic tools like machine learning (ML) and deep learning (DL) systems remain underdeveloped. This underinvestment results in a reduced ability to adopt emerging technologies that are transforming AD diagnostics.

## 2 Research Objectives

The primary objective would be to identify and evaluate suitable model architectures for classifying Alzheimer's disease, along with understanding the effects of data augmentation and transfer learning with regard to the models evaluation metrics (accuracy, sensitivity, specificity, etc.). There are many different model architectures available, such as ResNet, DenseNet, and Vision Transformers (ViT), just to name a few, each offering their own unique learning capabilities. Selecting suitable models, while also comparing the evaluation metrics, can be tricky and requires experimentation. This project will focus on the comparison of selected models along with the effects of augmentation and transfer learning, compiling, and analysing each of the metrics to determine the most effective architecture for accurate Alzheimer's disease classification.

To achieve this, the following research questions are proposed:

- **RQ1:** What is the performance impact of specific data transformations (e.g., rotation, scaling, and cropping) on the evaluation metrics (accuracy, sensitivity, specificity, and ROC AUC score.) of deep learning models in Alzheimer's disease classification?
- **RQ2:** How do different deep learning architectures (e.g., 3D CNN, EfficientNet, variation of a custom hybrid model[s]) compare in terms of (accuracy, sensitivity, specificity, and AUC-ROC) when classifying Alzheimer's disease in MRI images?
- **RQ3:** To what extent does transfer learning, using pre-trained models from non-medical and medical domains, improve the evaluation metrics (accuracy, sensitivity, specificity, and AUC-ROC) of deep learning models for Alzheimer's detection in MRI images?

The goal of these questions is to show how well different model architectures and classification methods work for Alzheimer's disease, which could lead to improvements in diagnostic approaches.

### 3 Literature Review

With the rapid advancements in deep learning in recent years, Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs) have emerged as focal points in computer vision tasks. This trend was started by Krizhevsky, Sutskever, and Hinton groundbreaking 2012 paper [7], which introduced a deep convolutional neural network capable of classifying 1.3 million high-resolution images in the Large Scale Visual Recognition Challenge 2010 (LSVRC-2010) ImageNet dataset into 1,000 classes. This model significantly outperformed the previous state-of-the-art methods on the same benchmark, This made ImageNet as a gold standard for evaluating image classification models. The success of this paper not only sparked the development of other CNN-based models which we will discuss in more detail later, but also marked the beginning of a new era for deep learning research.

A number of new techniques and technologies have been tried and tested in the medical field. Specifically, advancements in data analysis and deep learning have enabled the development of accurate, potentially life-saving models that can assist doctors and physicians in diagnosing disease and identifying new health conditions. One such strategy by Reshan et al. [8] implemented a MobileNet CNN based model to detect pneumonia in X-ray images. The model had 13 million parameters, 28 layers, and input layer was  $224 \times 224 \times 3$ . With this model architecture they then trained the model with two datasets, the first consists of 5,856 images which contains 4,273 pneumonia labelled images, and 1,583 normal control labelled images. The second dataset, composed of 1,431 pneumonia labelled images and 1,431 normal control images. After 32 epochs, they achieved a binary classification accuracy of 90.85% and precision and recall were 95.28% and 91.41% respectively.

Another study by Gai et al. [9] compared two prominent deep learning approaches for identifying lung cancer from 3D CT scans: the Data-efficient Image Transformer (DeiT-S) [10] and the EfficientNet3D architectures were used. DeiT-S is a Vision Transformer (ViT) model designed to improve data efficiency by leveraging distillation through attention, enabling it to learn effectively from smaller datasets compared to traditional ViTs. This study is relevant to the current research as it explores the performance of different deep learning models in medical imaging. The findings indicated that while both approaches had advantages under specific conditions, the EfficientNet3D CNN-based architecture outperformed the DeiT-S transformer model. EfficientNet3D achieved the highest accuracy of 93.38% and an AUC of 98.1% when trained with self-supervised learning (SSL), compared to DeiT-S, which achieved an accuracy of 87.75% and an AUC of 95.57%. The authors highlighted limitations of current Vision Transformer (ViT) models in medical imaging, noting that they require significantly larger datasets to perform effectively compared to CNN-based models. This poses a challenge in the medical domain, where large datasets are often unavailable. These findings underscore the suitability of CNN architectures like EfficientNet3D for medical imaging tasks, particularly when data availability is limited.

A study by Jain et al. [11], similar to the work by Gai et al. [9], compared CNN, ResNet, and Vision Transformers (ViT) for multi-classification of chest diseases. The key difference in this study was the use of a much larger dataset. Their dataset consisted of the freely available NIH Chest X-ray dataset, which includes 112,120 X-ray images from 30,805 patients, with disease labels for each image. The authors used a subset of 85,000 images for their performance comparison. The results



showed that both CNN architectures performed similarly, with validation accuracies of 92.68% for CNN and 93.34% for ResNet. Comparing this to the Vision Transformer models, ViT-v1 and ViT-v2, achieved validation accuracies of 92.89% and 92.95%, respectively.

The ViT models used in this study included ViT-v1/32, ViT-v2/32, and ViT-ResNet/16. ViT-v1/32, the base variant, utilizes a  $32 \times 32$  input patch size, while ViT-v2/32 also uses a  $32 \times 32$  patch size but with improvements in efficiency and generalisation compared to ViT-v1. The ViT-ResNet/16 variant, on the other hand, uses a smaller  $16 \times 16$  input patch size and combines the Vision Transformer's attention mechanism with the feature extraction capabilities of ResNet. It was pre-trained on the ImageNet-21k dataset, which is a form of self-supervised learning, offering a large and diverse dataset for pre-training and improving model performance. Despite these differences, ResNet still outperformed the standalone Vision Transformers in terms of validation accuracy.

The authors also introduced a hybrid ViT-ResNet model, which combined the Vision Transformer's state-of-the-art attention mechanism with ResNet's feature extraction capabilities. This model yielded a validation accuracy of 94.07%, similar to ResNet, but with 10% faster training times and a significantly improved ROC curve. The authors highlighted that while CNNs and ResNet provide robust results, ViTs excel in diagnostic accuracy when pre-trained on large datasets such as ImageNet-21k. These findings align with the conclusions in the study by Gai et al. (2024) [9], where the authors also noted that a large dataset is crucial for maximising the performance of ViT models.

The literature that was discussed up to this point above shows the effectiveness and strengths of both CNN-based networks and the newer ViT models. While ViT models generally underperformed compared to CNNs in most cases, the authors attributed this to the limited availability of data for training. Smaller datasets, which are common in medical imaging, pose significant challenges for ViT models that typically require large amounts of data to realise their full potential. CNN-based models demonstrate superior performance on smaller datasets, making them more suitable for medical imaging applications where data scarcity is a common limitation.

The following literature shows the effects of data transformations being applied to training sets and compares evaluation results with and without data transformations and augmentations. Data augmentation is a set of techniques that are used to increase the size and or quality of the dataset by means of image rotations, scaling, random cropping, and variations in pixel values to insert variations within the training dataset. Data augmentation can potentially help in two ways first by generating more data from limited amount of data within smaller datasets such as the medical domain in which the data used in this project is from and, secondly it helps the model to generalise on unseen data which prevents over fitting.

This [12] study investigates the effectiveness of data augmentation techniques for a standard CNN based sequential model with convolutional, pooling, dropout, flatten, and dense layers. The models input is of  $150 \times 150$  pixel image dimensions. The architecture is assessed with and without data augmentation methods to determine the impact of data augmentation on model performance. To compare the performance the dataset the authors are using is the Oxford 102 flower dataset which is divided into 102 flower categories, each category containing approximately 40 and 258 images.

In total the dataset has 8,189 images. The augmentation pipeline the authors are proposing includes four sequential steps which include rescaling, rotating, flipping, and converting to greyscale. The results from the study show improvements to model accuracy by applying the proposed augmentation pipeline to the input images, the models accuracy increased by 3.33%.

The following authors [13] are proposing a new method for the classification of foods. Their paper discusses previously tested methods on food classification but they mention a common problem with other comparative models used in the same domain and that is, food classification can be challenging when dealing with foods that are similar in shape. The authors from this paper propose a high-accuracy food image classification model with data augmentation and feature enhancement through the use of a vision transformer (AlsmViT) which can accurately handle foods with similar shapes. In their paper they are using two food image datasets, Food-101 and Vireo-172, respectively. They present a data augmentation pipeline that includes random resizing and cropping, horizontal flips, and TrivialAugmentWide, which applies a single augmentation to each image. Additional transformations include Gaussian blur to slightly alter image details, greyscale to focus on shape over colour, solarisation to add strong colour noise and focus on shapes, colour jitter to adjust brightness, and finally random erasing, which masks random regions of an image which forces the model to learn features outside the masked area. Their findings concluded that using the augmentation pipeline accuracy increased from 89.9% to 95.2% when testing results without the augmentation and transformation on the Food-101 dataset and accuracy improved from 89.2% to 94.3% on the Vireo Food-172 dataset.

One drawback from the data augmentation and transformation techniques discussed above don't include the impact of utilising a pre-trained model alongside data augmentation and transformation techniques. This is essentially true for smaller datasets like that of the study by [12] which uses the Oxford 102 dataset which is relatively small. Exploring the effectiveness of data augmentation and transformation techniques on smaller datasets while also combined with pre-trained models, could yield interesting insights. This raises the question of, how effective are augmentation techniques in improving classification accuracy for medical datasets with limited samples, such as MRI images?

## 4 Dataset

The dataset used for this project was obtained from Alzheimer's Disease Neuroimaging Initiative (ADNI) study [14], in which it actively supports the investigation and development of treatments that slow or stop the progression of Alzheimer's disease (AD). Researchers at over 60 clinical sites in the USA and Canada collect data to study the progression of AD in the human brain across normal ageing (CN), mild cognitive impairment (MCI), and Alzheimer's disease and dementia (AD). The ADNI study was first started in 2004 as a public/private partnership led by Principal Investigator Michael W. Weiner. The goal of the ADNI study is to evaluate whether chemicals found in blood, along with magnetic resonance imaging (MRI) and PET scans, can be used together with clinical measures (cognitive and neuropsychological tests), to assess the brain's structure and function over the course of three disease states (cognitively normal/unimpaired, mild cognitive impairment, dementia).

Dataset	Image Scans	CN	MCI	AD	Male	Female	Age
ADNI1: Complete 1Yr 1.5T	2294	705	1113	476	1341	953	75 ± 6.6

Table 1: Details of the ADNI dataset. CN: cognitively normal; MCI: mild cognitive impairment; AD: Alzheimer's disease.

The unprocessed dataset is quite imbalanced, with 705 CN, 1113 MCI, and 476 AD images for each label, respectively. Because of this, class balancing will be applied to the dataset. Along with removing the MCI data. Initial model development will be a binary classification model which will focus on the CN and AD class labels. Later on, the introduction of the MCI data could be considered for further training and evaluation of model performance.

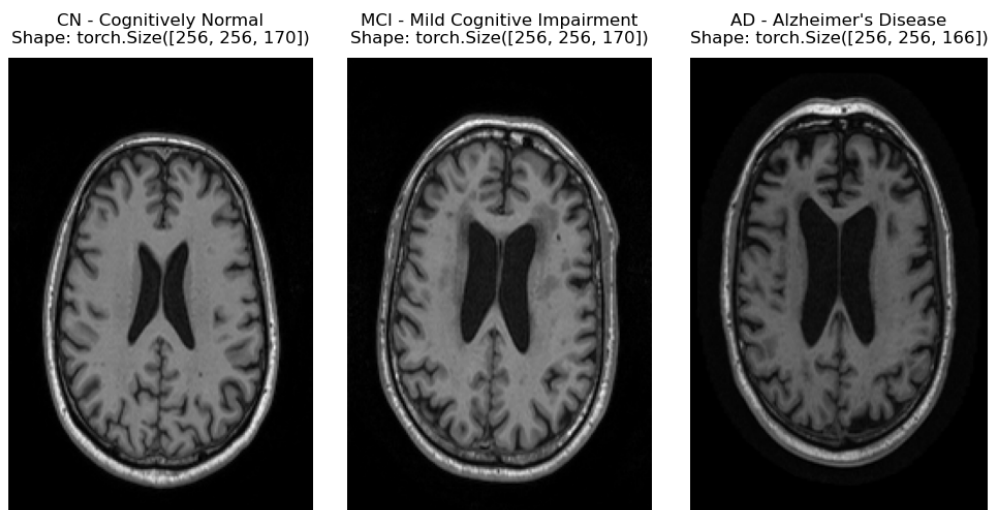


Figure 1: Unprocessed middle slice from each class label in the axial plane.

## 4.1 Image Preprocessing

There are several preprocessing steps applied to the images before any model can be trained. These preprocessing steps ensure the images are consistently formatted, free of artifacts, and correctly aligned, this will enhance the models ability to learn relevant features. Each of the preprocessing steps will be discussed in the order of application below. Figure 2 shows the output of preprocessing steps.

The bias field is a smooth, low-frequency signal that can distort MRI images by intensity variations in tissues, especially in older MRI machines. It can be caused by various factors such as differences in scanner parameters or variations in patient anatomy. N4 Bias Correction helps to reduce these intensity variations.

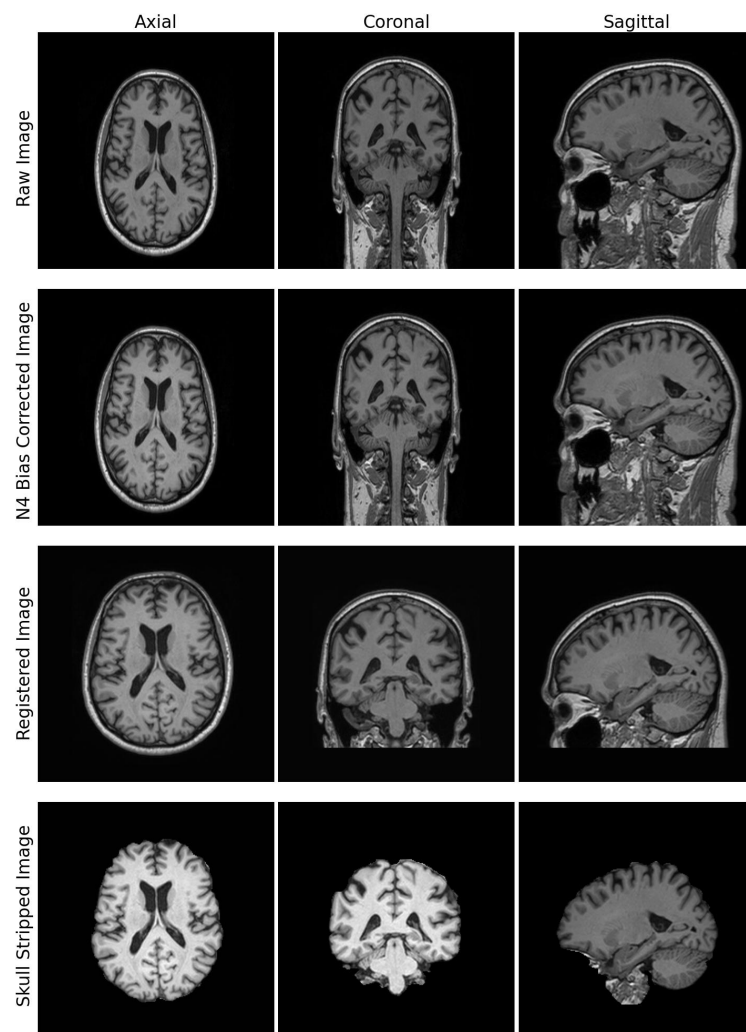


Figure 2: A sample of an MRI slice in three planes (axial, coronal, and sagittal) with preprocessing steps applied. (Row 1) Original MRI; (Row 2) N4 Bias Correction; (Row 3) Affine Transformation; and (Row 4) Skull Stripping.

Image registration is an important component of medical image analysis, it enables structural and functional correlation between images obtained from various modalities, including MRI, CT, and PET. In brain imaging, aligning scans from multiple subjects into a single reference frame is crucial for accurately comparing anatomical data across individuals.

Skull stripping is a technique used to remove non-anatomical structures, such as the skull, from brain images while preserving the brain tissue. This process reduces the amount of data to be analysed and enables deep learning models to concentrate on the relevant anatomical information, enhancing their learning ability.

The FMRIB Software Library (FSL) [15], is a widely used preprocessing tool and offers the capability to apply Affine Registration and Skull Stripping. It will be used in this project to ensure precise alignment of the images for effective analysis [16].

## 5 Methodology

This section outlines the methodology used for developing and evaluating the proposed deep learning system for Alzheimer's disease detection. The methodology is structured to provide an overview of the tools, frameworks, and workflows used, along with the design and evaluation of the proposed model architectures.

The section begins with an overview of the software stack used for implementing the deep learning system, including libraries and frameworks. Following this, the workflow of model experimentation and analysis is discussed, accompanied by a diagram to provide a visual representation of the proposed system and its components and how they connect to one another.

Finally, the methodology will finish on the model architectures and evaluation metrics in detail. These metrics, selected based on reviewed research papers from Section 3, align with common practices in medical image classification and general image classification tasks. The selected metrics will form the basis for evaluating the baseline model trained on the proposed dataset.

### 5.1 Software System Stack

Matplotlib is a low-level graph plotting library in python that serves as a visualisation utility. Matplotlib will be used for plotting model metric results and plotting MRI images. And also, for gaining visual insights into the dataset. FSL [15] is a comprehensive library of analysis tools for FMRI, MRI, and diffusion brain imaging data. For this project, the FSL toolkit provides a range of useful tools for preprocessing MRI and FMRI image data.





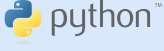

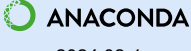
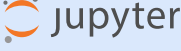


Deep Learning Frameworks	Visualisation & Image Preprocessing
 Version 1.4.0	 6.0.7.14, 8th October 2024
 Cuda version 11.8	 Version 3.9.1
Development Environment	
 Version 3.9	
 2024.02-1	 2024.02-1
Hardware and OS Components	
 Cuda version 11.8	 Build 22H2
CPU - Ryzen 7 5800x 8 Cores	GPU - Nvidia RTX 3080 10Gb
SSD - 1TB Samsung 990 Pro	RAM - 32GB 3600Mhz

Figure 3: Frameworks and System Architecture that is being used for data analysis and training for this project.

The deep learning frameworks used in this project are PyTorch and MONAI. PyTorch, developed by Facebook’s AI Research lab, is an open-source framework that supports CUDA for GPU acceleration this will improve the training times for large models and datasets. MONAI, built on PyTorch by NVIDIA and MITK, is designed for medical imaging datasets, providing optimised tools for training and inference. By leveraging CUDA, MONAI benefits from faster computations, which is important when it comes to medical imaging such as MRIs as these images can get large.

The development environment consists of Anaconda, Jupyter notebooks, GitHub, and Python. Anaconda simplifies dependency management and integrates necessary libraries, making it easier to work with frameworks like PyTorch and MONAI. Jupyter notebooks offer an interactive environment for coding and experimentation, while Python and GitHub handle programming and version control.

The system architecture includes a Windows 10 desktop with 32GB of RAM, a 1TB SSD, and an Nvidia RTX 3080 GPU with 10GB of video memory. The GPU is the main processing unit for training, with CUDA enabling efficient data transfer and batch processing. This setup will accelerate training and reduces processing time for large medical imaging datasets such as the one we will be using (ADNI).

## 5.2 Workflow

The system workflow follows a structured methodology, guiding the process from the raw dataset through to model evaluations. This approach ensures efficiency and consistency, covering each phase from data preprocessing to result compilation and analysis. The deep learning system, introduced in Section 5.1, provides the foundational software stack for the workflow. The next key component is dataset preprocessing. Before training and analysis, images undergo several preprocessing steps to standardise format, remove artifacts, and align features, optimising them for model training. Details of specific image preprocessing steps was covered in Section 4.1.

Cross-validation is a technique for evaluating model performance on unseen data by dividing the dataset into multiple folds, or subsets. In the testing environment, 5-fold cross-validation will be applied, where one fold serves as the validation set and the model trains on the remaining four folds. This process repeats five times, with each fold taking a turn as the validation set. The evaluation metrics from each iteration are then averaged, providing a more robust estimate of the models overall performance. Any chosen metric can be applied, and averaging across these folds yields a reliable and robust final performance result.

At the end of the workflow, results will be compiled and compared across different data transformations, transfer learning approaches, and model architectures. This comparison will help to address the research questions outlined in Section 2.

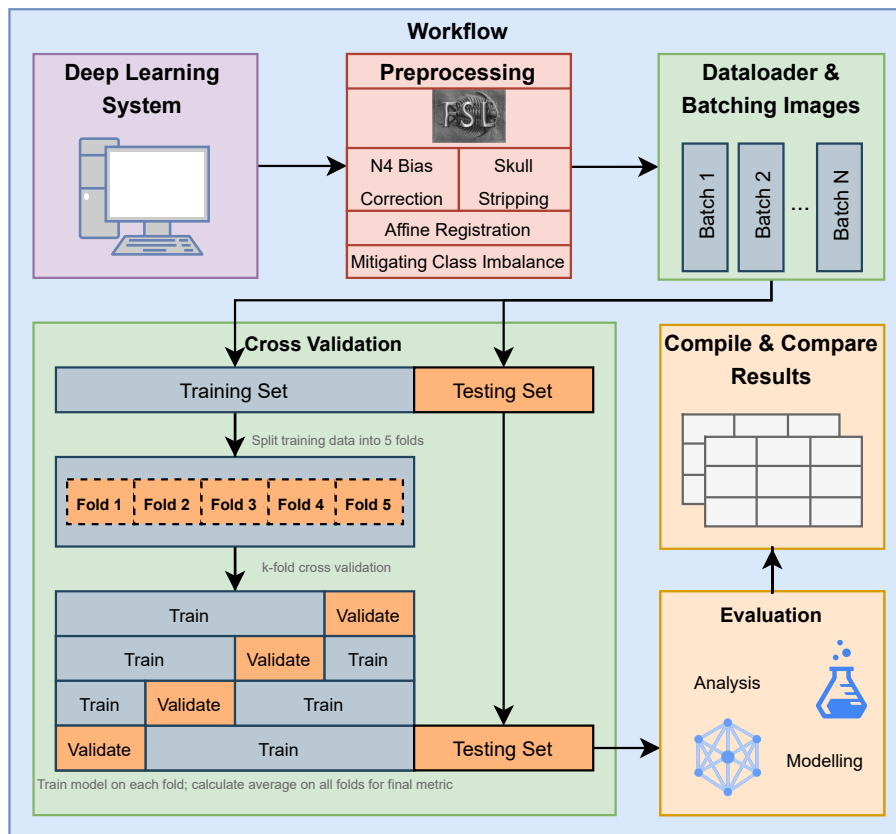


Figure 4: Diagram of the deep learning system workflow for experimentation and training.

### 5.3 Model Architectures

The first model to be trained is ResNet-50 which was developed by He et al. in 2015 [3] at Microsoft Research Asia. Residual networks get their name from the residual blocks that build up a residual network such as ResNet.

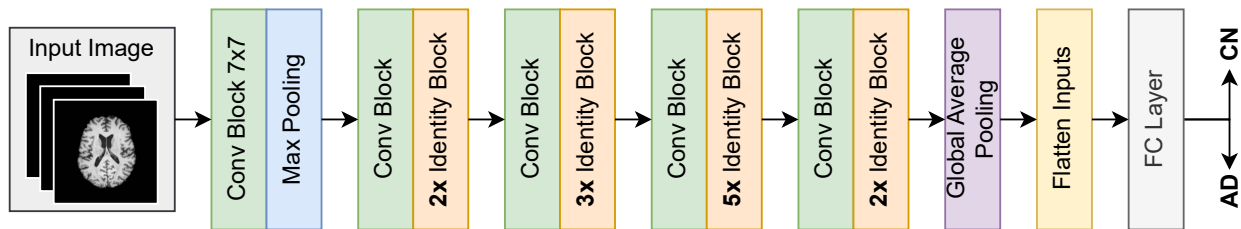


Figure 5: ResNet-50 Model Architecture

ResNet models allow you to train very deep neural networks and avoiding the problem of vanishing gradients by using shortcut connections, which skip one or more layers and help propagate gradients through the network more effectively. Before residual networks, adding more layers to a network didn’t always improve the model performance, in some cases, it actually led to poorer results. This phenomenon, known as the degradation problem, occurred because deeper networks struggled to learn meaningful representations and gradients would diminish or explode during training, making optimisation difficult.

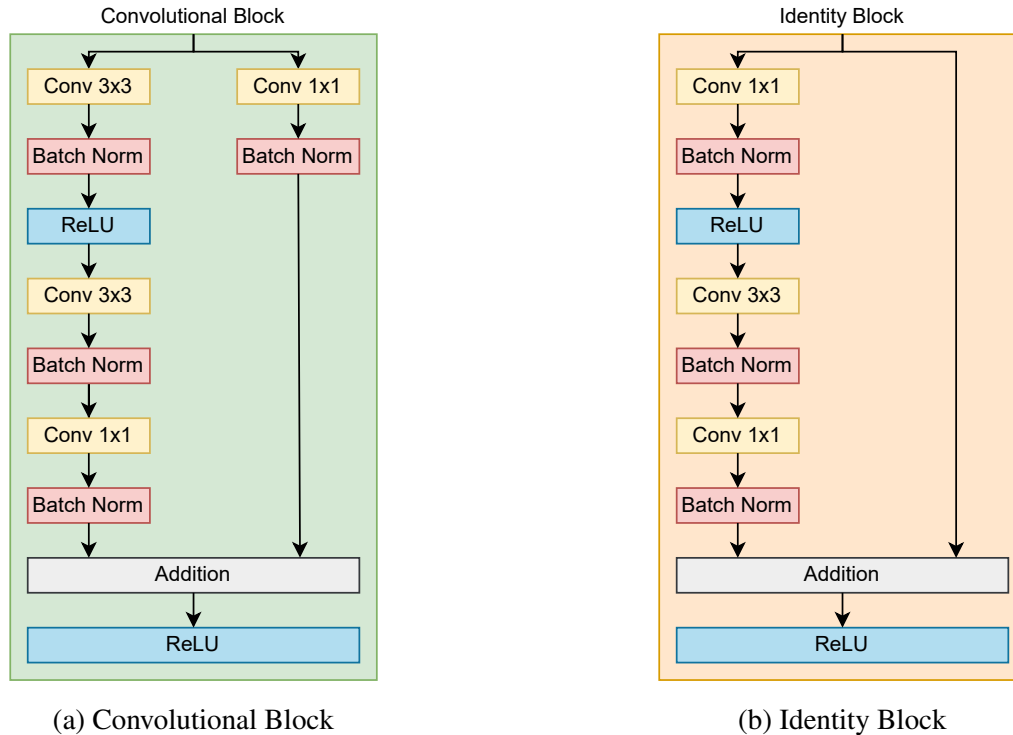


Figure 6: Convolutional and Identity Blocks in ResNet-50. These building blocks use skip connections to allow for deeper networks which avoids the degradation problem.



### 5.3.1 ResNet-50 initial setup & model implementation

The code in Listing 1 initialises a ResNet-50 model based on the MONAI 3D implementation. It also includes a device-agnostic setup, which will automatically use the CUDA device (GPU) if available, or default to the CPU.

#### ResNet-50:

```
1 # Define device
2 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
3
4 # Initialize ResNet
5 model = ResNet(
6     block="bottleneck",
7     layers = (3, 4, 6, 3), # ResNet-18 -> (2, 2, 2, 2) ResNet-50 -> (3, 4, 6, 3)
8     block_inplanes=(64, 128, 256, 512),
9     spatial_dims=3,
10    n_input_channels=1,
11    num_classes=2
12 ).to(device)
```

Listing 1: ResNet-50 model initialisation using MONAI and device agnostic setup.

Another important point regarding the ‘layers’ parameter is that it corresponds to the number of layers in each ResNet block. Specifically, for ResNet-50, as shown in Figure 5, there are 5 blocks, with the middle blocks having 3, 4, 6, and 3 layers respectively.

#### Loss function and Optimiser:

```
1 # Loss function and optimizer
2 loss_function = torch.nn.CrossEntropyLoss()
3 optimizer = torch.optim.Adam(model.parameters(), lr=1e-4)
```

Listing 2: ResNet-50 optimiser and loss function setup code.

The loss function being used is cross entropy loss along with the Adam optimiser with a learning rate of  $1 \times 10^{-4}$  for the baseline ResNet-50 model.

For the remaining code snippets see Appendix B, for training and evaluation python code.

## 5.4 Models Evaluation and Performance

Several metrics have been developed to assess the effectiveness of deep learning techniques in medical imaging, tailored to specific tasks such as classification and segmentation. For image segmentation tasks, widely used metrics include the Dice coefficient and Jaccard index, which measure the overlap between predicted and actual segmentations, along with surface distance measures that evaluate boundary accuracy. In image classification tasks, metrics such as accuracy, precision, recall, and F1 score are commonly used to provide a comprehensive view of model performance [17]. These metrics not only gauge overall correctness but also offer insights into the models ability to minimise false positives and negatives, critical for reliable diagnostics.

### 5.4.1 Evaluation Metrics

In medical applications, defining the right evaluation metrics is difficult. It is important to minimise both false positives and false negatives, but, a false negative where a condition is incorrectly identified as healthy can have a serious consequences such as delayed treatment. Accuracy being a common metric and providing a high-level overview of performance, it may not be the most suitable indicator for medical applications. Accuracy is limited because it measures the percentage of correct predictions it doesn't check for any errors such as an imbalance of the dataset being used. Accuracy can offer a broad view of model performance, but additional metrics, discussed below, are necessary to better understand the models predictions and ensure it meets the standards in healthcare settings.

#### Accuracy

Formulae 1: Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where *TP* and *TN* are the numbers of true positives and true negatives, respectively, and *FP* and *FN* are the false positives and false negatives.

#### Recall

Recall, or sensitivity, measures the models ability to correctly identify positive cases, such as patients with Alzheimer's disease (AD). In medical contexts, failing to identify a positive case (false negative) can lead to delayed treatment or insufficient intervention, potentially worsening the patient's condition. Thus, recall is crucial for minimising false negatives.

Formulae 2: Recall

$$\text{Recall (or TPR)} = \frac{TP}{TP + FN}$$

Where *TP* and *FN* are the numbers of true positives and false negatives, respectively.

## Specificity

Specificity, on the other hand, assesses the model's ability to correctly identify negative cases, such as Cognitively Normal (CN) individuals, among all actual CN labels. Together, recall and specificity provide a more nuanced understanding of the models performance.

### Formulae 3: Specificity

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Where *TN* and *FP* are the numbers of true negatives and false positives, respectively.

## AUC-ROC Score

The Area Under the Receiver Operating Characteristic Curve (AUC-ROC) offers a broad measure of the model's discrimination ability. A higher AUC-ROC score indicates that the model is better at distinguishing between Alzheimer's disease (AD) and Cognitively Normal (CN) images, reflecting its overall effectiveness. The ROC curve is a plot of the True Positive Rate (TPR) (also known as recall or sensitivity) against the False Positive Rate (FPR).

The AUC (Area Under the Curve) calculates a single value. An AUC of 1.0 means perfect classification, compared to an AUC of 0.5 where it means no discriminatory ability (no better than random guessing). The formulas for calculating TPR and FPR are shown below.

### Formulae 4: Recall

$$\text{Recall (or TPR)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Where *TP* and *FN* are the numbers of true positives and false negatives, respectively.

### Formulae 5: False Positive Rate

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

Where *FP* and *TN* are the numbers of false positives and true negatives, respectively.

### 5.4.2 Performance of proposed models

The performance of the trained model was evaluated with the metrics proposed in section 5.4. The results are shown in Table 2, and shows good results with the baseline ResNet-50. It is also important to note that these metrics were not evaluated with cross validation. Semester Two will focus on the training of other models and also using cross validation to get a robust calculation of metrics.

**Table 2**

ResNet-50 model evaluation metrics and the hyperparameters used for training.

Models	Accuracy (%)	Recall	Precision	F1-Score	ROC Score
3D ResNet-50	90.03	0.84	0.90	0.87	0.97
ResNet-50 Hyperparameters					
Optimiser	Adam				
Loss Function	Cross-Entropy Loss				
Learning Rate	$1 \times 10^{-4}$				
Batch Size	4				
Training Time (minutes)	55.64				

Table 2: Comparison of model evaluation metrics and ResNet-50 hyperparameters

Figure 7 below shows the train and test loss curves along with test accuracy. More detailed metrics can be found in Appendix A.

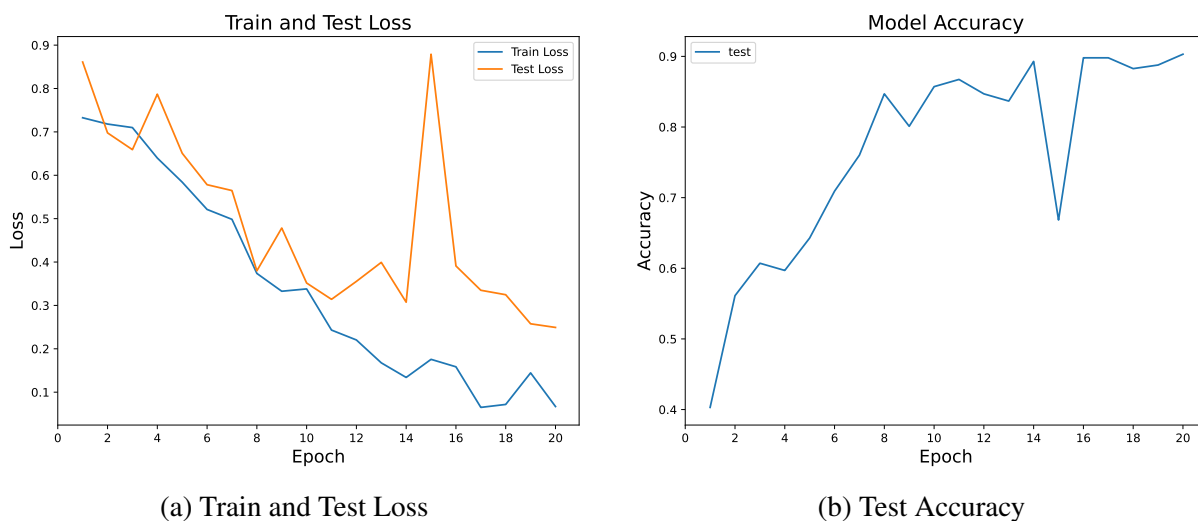


Figure 7: ResNet-50 loss curves and accuracy

## 6 Tools and Technologies

The tools and technologies can be categorised into two parts which in their own way are interconnected. Part one is the experimentation stage. Initial stages of the data analysis, model experimentation, and model evaluation metrics will be done at this phase. Part two will be focused on application development and deployment. This will connect the model to the application so end users can interact with the trained model.

### 6.1 Part One - Experimentation

Part one is comprised of data exploratory analysis, data preprocessing and model experimentation and development. A range of tools will be used for getting the dataset in a form that's ready for modelling and also evaluating models performance.

#### Python

Python is the programming language that will be used for the Data Analytics, Data Exploratory Analysis, and Model Development. Python is an interpreted language which means code is run through the interpreter for execution rather than compiling the code to machine code. It is a high level, object-oriented language and allows for rapid development because of it's ease of use.

#### PyTorch

PyTorch is a fully featured Python based framework for building and deploying deep neural networks. Deep neural networks are an off branch of machine learning that is commonly used in applications like image recognition and natural language processing (NLP). PyTorch also has intuitive support for Graphical Processing Units (GPUs), more precisely, support for NVIDIA's (CUDA) platform. This will off load the heavy tasks to the GPU such as training deep models which will increase training times.

#### Jupyter

Jupyter is a web based interactive development environment. The idea behind Jupyter is that you work within a notebook, which allows you to write and execute code in a cell by cell manner. Cells can contain text, code, and even markdown. Jupyter notebooks are useful when you want to demonstrate your code or keep track of experiments in a tidy manner.

#### Medical open network for AI (MONAI)

MONAI [18] is a consortium-led PyTorch-based framework for deep learning in healthcare. MONAI extends PyTorch to support medical data, with a particular focus on imaging, and provide purpose-specific AI model architectures, transformations and utilities that streamline the development of medical AI models.

## FSL

FSL [15] is a comprehensive library of analysis tools for FMRI, MRI, and diffusion brain imaging data. For this project, the FSL toolkit provides a range of useful tools for preprocessing MRI and FMRI image data. Two tools from the FSL application will be used: the first is FLIRT (FMRIB's Linear Image Registration Tool), a fully automated, robust, and accurate tool for linear (affine) intra- and inter-modal brain image registration. The second tool is BET (Brain Extraction Tool), which removes non-brain tissue from whole-head images. BET can also estimate the inner and outer skull surfaces, as well as the outer scalp surface, provided that high-quality T1 and T2 images are available.

## 6.2 Part Two - Deployment

Part two will focus on the usability of the model, users will be able to upload an image to the model for inference and processing. Below will be an list of tools that will be utilised for the deployment and the development of the system pipeline.

### React

React.js is a JavaScript library for building efficient and flexible fronted applications. It's component architecture allows developers to create UI elements such as forms, tables, and navigation bars with usability in mind. With reacts state management you also get the flexibility of of manipulating and updating UI components with immediate effect to the user.

### Node.js

Node.js is an open-source, cross-platform JavaScript runtime environment and library for running web applications outside the client's browser. It provides an event-driven, non-blocking I/O model, which makes it highly efficient and scalable. This will enable the frontend (react) to connect to the backend (database and PyTorch model API endpoint).

### TorchServe

TorchServe is tool that works with PyTorch that enables you to deploy trained models at scale. With it's ease of use, and extensive documentation, getting a model deployed can be integrated into the production pipeline with little effort.

### Streamlit

Streamlit lets you transform Python scripts into interactive web applications for the user to interact with your trained models. Streamlit is built with Python, which enables fast and interactive prototyping for data driven web applications.

## 7 Project Plan

This section outlines the overall project structure, goals, and methodologies that will be followed throughout the duration of the project. The project is divided into two main phases, Semester One and Semester Two. The project plan also discusses challenges faced during semester one and also potential challenges in semester two. Gantt charts were also created at the start of semester one, the reasoning behind it was to get a better understanding of what sections of the report along with the project plan needed to be completed by, this has a sense of urgency for certain sections of both the project report and also the project research and development.

### 7.1 Semester One

Semester One will be focused on building a experimentation and software framework to carry out the development and testing of models in Semester Two. Furthermore, I will be researching existing literature related to the project’s domain in order to explore relevant techniques and models that could be applied to this project. This includes reviewing current work in similar areas and also studying methodologies, models used, and common challenges encountered in similar projects and research. This initial phase aims to provide insights that will help with model selection and also with data preprocessing and transformations.

The dataset used in this project was obtained on May 2nd, 2024, following an initial access request submitted on April 15th, 2024. The request included the project’s scope and intended use of the dataset. Access to this dataset was crucial for the feasibility of the project. While other datasets, such as the Open Access Series of Imaging Studies (OASIS), are available, they were not necessary as access to the ADNI dataset mentioned in section 4 was granted.

#### 7.1.1 Gantt Chart

The Gantt chart below (Figure 8) illustrates the week-by-week breakdown of tasks for Semester One, highlighting the workload distribution and achieved project milestones.

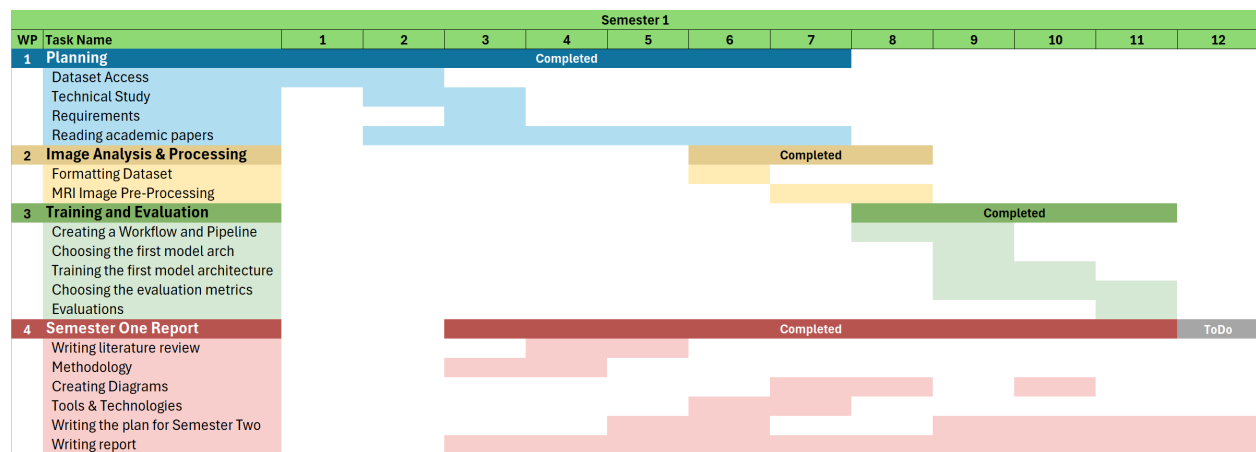


Figure 8: Workload schedule for Semester One, showing weekly task allocations and milestones.

As illustrated in Figure 8, the chart outlines the key work packages and the associated tasks that were necessary to complete each package. Milestones reached during this semester are clearly marked as "completed", showcasing the progress made. This demonstrates that the project has not only aligned to the planned schedule but has also met the objectives for Semester One.

The chart also shows the approach undertaken in managing the project, with tasks sequenced to achieve steady progress. Each milestone reached reflects significant steps toward the overall project goals, including data preparation, model evaluation, and initial experimentation. This progress indicates that the project is well aligned with the semester timeline, and sets a strong foundation for the upcoming work in Semester Two, which will focus more on the comparison of different model architectures and transformations with respect to model performance. Additionally, a react based web application will be designed and developed to allow users to create patient profiles, upload MRI scans, and use the cloud deployed models for predictions.



## 7.2 Semester Two

In Semester Two, the primary focus will be on the full development and evaluation of the models. This phase will involve compiling the results of the models to address the research questions outlined in Section 2. Following the model evaluations, the next stage will focus on the usability of the developed models. Two deployment methods are proposed for this phase: The first option involves deploying the model using TorchServe onto a cloud service such as Amazon Web Services (AWS). The second option is to run the model in a Streamlit environment. The choice between these two options will depend on time availability and the progress made during the semester. The first option also requires full-stack development, for which I have chosen React along with Node.js. Having both options ensures flexibility in terms of time and workload management, allowing me to adjust based on project progress.

### 7.2.1 Gantt Chart

The Gantt chart below (Figure 9) illustrates the week-by-week breakdown of tasks for Semester Two, highlighting the workload distribution and project milestones that have yet to be achieved such as selecting models to train and also deploying onto a cloud environment.

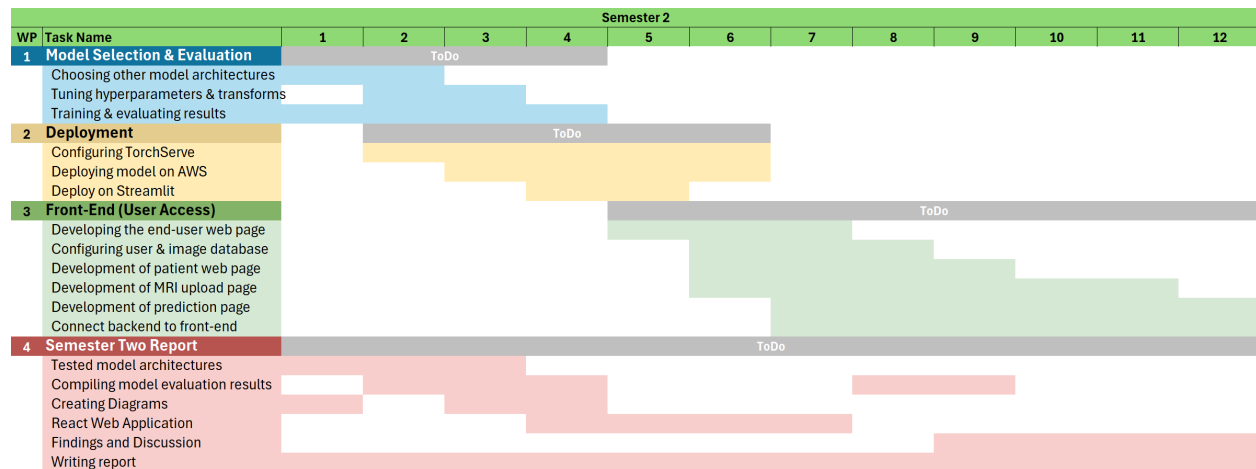


Figure 9: Workload schedule for Semester Two, showing weekly task allocations and milestones.

To get a better understanding of how the web application might function, wireframes provide a visual representation of its potential layout and structure. These wireframes help outline the design and organisation of key elements such as navigation, user inputs, and how predictions from the deployed model will be presented to the user.

The wireframes developed are intentionally kept simple to serve as a foundational guide for the application’s development in Semester Two. These designs represent early stage concepts of potential layouts but play an important role in visualising the user interfaces for the project.

### 7.2.2 Homepage Wireframe

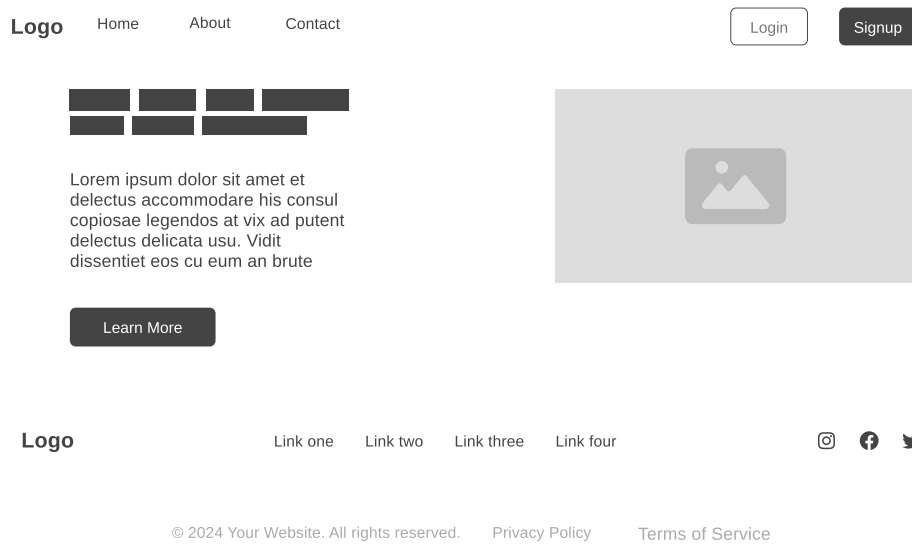


Figure 10: Homepage wireframe for the proposed web application.

### 7.2.3 Patient Page Wireframe



Figure 11: Patient page wireframe for the proposed web application.

### 7.2.4 Personas and Use Cases

**Dr. Emily Carter**

**Role:** Neurologist

**Specialty:** Alzheimer's Disease

**Experience:** 12 years specialising in neurological disorders, with a focus on Alzheimer's disease.

**Emily's Use Cases**

- Upload MRI scans for patients suspected of having Alzheimer's.
- Analyse results using the AI model to identify signs of Alzheimer's disease.
- Add and review patient notes for ongoing diagnosis and treatment planning.
- Access historical patient records to monitor disease progression over time.

**Emily's Features**

- Intuitive dashboard with patient profiles.
- Easy upload and integration of MRI scans.
- Quick access to model-generated insights with predictions for each class.
- Option to print or share reports with other healthcare colleagues.

Figure 12: User persona of Dr. Carter who specialises in Alzheimer’s Disease and progression.

**Sarah Foley**

**Role:** Radiology Technician

**Specialty:** Medical Imaging

**Experience:** 6 years of experience as a Radiology Technician, specialising in medical imaging procedures such as MRI, CT, and X-ray scans.

**Sarah's Use Cases**

- Upload MRI scans to the system after imaging sessions.
- Check scan quality to ensure they meet the requirements for analysis.
- Upload patient data, including MRI scans and ongoing notes, to ensure experienced physicians have all necessary information to conduct their analysis.

**Sarah's Features**

- Simplified workflow for uploading and tagging MRI scans.
- Notifications for successful uploads or errors in scan quality.
- Role-based access to ensure only technical tasks are accessible.
- Enable the assignment of doctors and physicians to patient profiles in the database, granting them appropriate access to patient data.

Figure 13: User persona of Ms. Sarah Foley who specialises in Medical Imaging.

**Joe Blogs**

**Role:** Patient

**Specialty:** Patient Under Observation

**Experience:** Joe Blogs has been living with Alzheimer's for 3 years, managing symptoms and following a treatment plan provided by his physician.

**Joe's Use Cases**

- View MRI scan results with simplified visuals or summaries provided by the doctor.
- Read doctors' notes or treatment plans shared with him.
- Download personal medical reports for reference or sharing with family members.

**Joe's Features**

- Limited, read-only access to personal medical information.
- Simplified interface with no technical jargon.
- Secure login to ensure privacy.

Figure 14: User persona of Mr. Joe Blogs with ongoing progression of Alzheimer’s Disease.

### 7.3 Challenges and Mitigation Strategies

As part of my research into tooling and frameworks, setting up and configuring the required software posed several challenges to ensure it met project needs. Some tools, such as the FSL library for image preprocessing, required a Linux-based system. To avoid using a completely separate system, I opted to use the Windows Subsystem for Linux (WSL) with an Ubuntu image, which allowed easier integration without the need for dual-booting or virtualisation.

However, I encountered a significant hurdle with storage. The original dataset was over 85GB, and my system had only approximately 50GB of free space available. Reinstalling Windows and the software stack to a larger drive was not a feasible option due to time constraints and also breaking the workflow.

To overcome this, I utilised an SSD cloning tool to transfer the original boot drive to a larger SSD, effectively increasing available storage without changing the software environment. This provided the necessary capacity to handle the dataset and ensured smooth preprocessing for subsequent steps in the project.

Another challenge that I will be facing is using MONAI for the first time, although I have never used it before, it builds on top of PyTorch's development best practices. MONAI also has detailed documentation and some tutorials on getting started. Because of this, initial development may progress slowly; being aware of these challenges will help me stay focused and adapt as I work through the project plan and development process.

## 8 Conclusion

This report presents the groundwork for the development of a deep learning framework aimed at training and evaluating models for the detection of Alzheimer's Disease using MRI-based imaging. It also discusses the dataset being used and some data exploratory analysis to get a better understanding of the data being used.

The deployment strategy was briefly discussed with a cloud-based solution with either Streamlit or a custom application built using React and Node.js

This report will follow through to Semester Two where other models will be trained and evaluated more robustly along with the development of a web-based application for deploying the trained models.

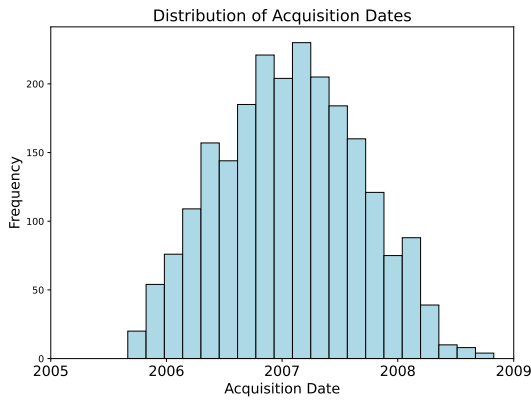
## 9 Bibliography

- [1] Michael A. DeTure and Dennis W. Dickson. “The neuropathological diagnosis of Alzheimer’s disease”. In: *Molecular Neurodegeneration* 14.1 (Aug. 2019), p. 32. ISSN: 1750-1326. DOI: 10.1186/s13024-019-0333-5. URL: <https://doi.org/10.1186/s13024-019-0333-5> (visited on 11/04/2024).
- [2] A. P. Porsteinsson et al. “Diagnosis of Early Alzheimer’s Disease: Clinical Practice in 2021”. en. In: *The Journal of Prevention of Alzheimer’s Disease* 8.3 (July 2021), pp. 371–386. ISSN: 2426-0266. DOI: 10.14283/jpad.2021.23. URL: <https://doi.org/10.14283/jpad.2021.23> (visited on 11/04/2024).
- [3] Kaiming He et al. *Deep Residual Learning for Image Recognition*. arXiv:1512.03385. Dec. 2015. DOI: 10.48550/arXiv.1512.03385. URL: <http://arxiv.org/abs/1512.03385> (visited on 11/04/2024).
- [4] eHealth Ireland. *eHealth Strategy for Ireland*. 2023. URL: <https://assets.gov.ie/16174/092e7c62f97b472b83cdb6dfdcdf5c7.pdf> (visited on 06/11/2024).
- [5] ESRI et al. *Developments in healthcare information systems in Ireland and internationally*. en. Tech. rep. ESRI, June 2021. DOI: 10.26504/sustat105. URL: <https://www.esri.ie/publications/developments-in-healthcare-information-systems-in-ireland-and-internationally> (visited on 11/06/2024).
- [6] Bruno Dubois et al. “Timely Diagnosis for Alzheimer’s Disease: A Literature Review on Benefits and Challenges”. en. In: *Journal of Alzheimer’s Disease* 49.3 (Dec. 2015). Ed. by Andrew Saykin, pp. 617–631. ISSN: 13872877, 18758908. DOI: 10.3233/JAD-150692. URL: <https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/JAD-150692> (visited on 11/06/2024).
- [7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks”. en. In: *Communications of the ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3065386. URL: <https://dl.acm.org/doi/10.1145/3065386> (visited on 11/19/2024).
- [8] Mana Saleh Al Reshan et al. “Detection of Pneumonia from Chest X-ray Images Utilizing MobileNet Model”. en. In: *Healthcare* 11.11 (Jan. 2023). Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, p. 1561. ISSN: 2227-9032. DOI: 10.3390/healthcare11111561. URL: <https://www.mdpi.com/2227-9032/11/11/1561> (visited on 10/10/2024).
- [9] Lulu Gai et al. “Comparing CNN-based and transformer-based models for identifying lung cancer: which is more effective?” en. In: *Multimedia Tools and Applications* 83.20 (June 2024), pp. 59253–59269. ISSN: 1573-7721. DOI: 10.1007/s11042-023-17644-4. URL: <https://doi.org/10.1007/s11042-023-17644-4> (visited on 11/15/2024).
- [10] Hugo Touvron et al. *Training data-efficient image transformers & distillation through attention*. arXiv:2012.12877. Jan. 2021. DOI: 10.48550/arXiv.2012.12877. URL: <http://arxiv.org/abs/2012.12877> (visited on 11/17/2024).
- [11] Ananya Jain et al. *A Comparative Study of CNN, ResNet, and Vision Transformers for Multi-Classification of Chest Diseases*. arXiv:2406.00237. May 2024. URL: <http://arxiv.org/abs/2406.00237> (visited on 11/14/2024).
- [12] Prajit Sengupta, Anant Mehta, and Prashant Singh Rana. “Enhancing Performance of Deep Learning Models with a Novel Data Augmentation Approach”. In: *2023 14th International*

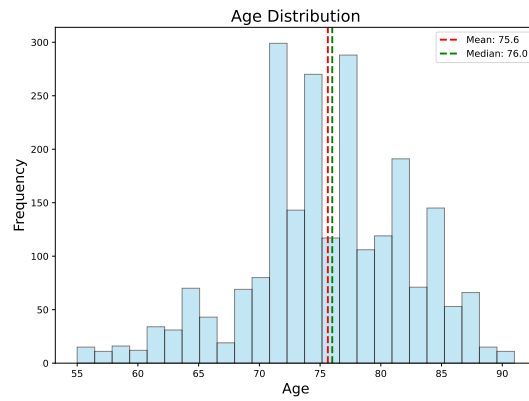
- Conference on Computing Communication and Networking Technologies (ICCCNT)*. ISSN: 2473-7674. July 2023, pp. 1–7. DOI: 10.1109/ICCCNT56998.2023.10308298. URL: <https://ieeexplore.ieee.org/document/10308298/?arnumber=10308298> (visited on 11/27/2024).
- [13] Xinle Gao, Zhiyong Xiao, and Zhaohong Deng. “High accuracy food image classification via vision transformer with data augmentation and feature augmentation”. en. In: *Journal of Food Engineering* 365 (Mar. 2024), p. 111833. ISSN: 02608774. DOI: 10.1016/j.jfoodeng.2023.111833. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0260877423004314> (visited on 11/27/2024).
- [14] R C. Petersen et al. “Alzheimer’s Disease Neuroimaging Initiative (ADNI)”. In: *Neurology* 74.3 (Jan. 2010), pp. 201–209. ISSN: 0028-3878. DOI: 10.1212/WNL.0b013e3181cb3e25. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2809036/> (visited on 10/11/2024).
- [15] Mark Jenkinson et al. “FSL”. eng. In: *NeuroImage* 62.2 (Aug. 2012), pp. 782–790. ISSN: 1095-9572. DOI: 10.1016/j.neuroimage.2011.09.015.
- [16] William R. Crum et al. “Registration of challenging pre-clinical brain images”. en. In: *Journal of Neuroscience Methods* 216.1 (May 2013), p. 62. DOI: 10.1016/j.jneumeth.2013.03.015. URL: <https://pmc.ncbi.nlm.nih.gov/articles/PMC3683149/> (visited on 10/23/2024).
- [17] Alexander Selvikvåg Lundervold and Arvid Lundervold. “An overview of deep learning in medical imaging focusing on MRI”. In: *Zeitschrift für Medizinische Physik*. Special Issue: Deep Learning in Medical Physics 29.2 (May 2019), pp. 102–127. ISSN: 0939-3889. DOI: 10.1016/j.zemedi.2018.11.002. URL: <https://www.sciencedirect.com/science/article/pii/S0939388918301181> (visited on 11/06/2024).
- [18] M. Jorge Cardoso et al. *MONAI: An open-source framework for deep learning in healthcare*. arXiv:2211.02701. Nov. 2022. URL: <http://arxiv.org/abs/2211.02701> (visited on 10/11/2024).

# 10 Appendix A

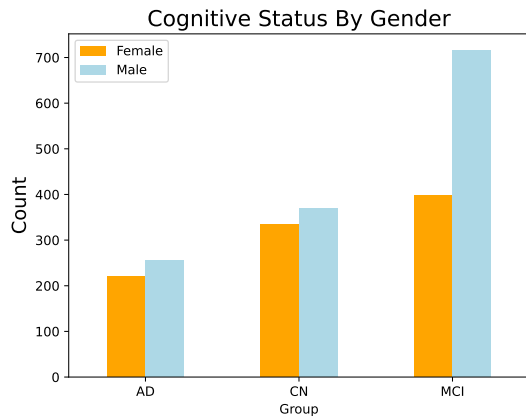
## Initial EDA Analysis:



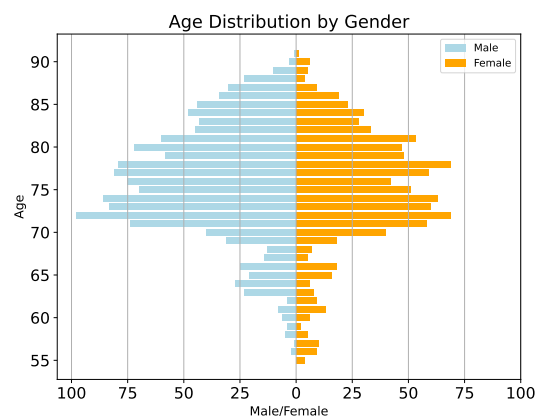
(a) Acquisition Dates



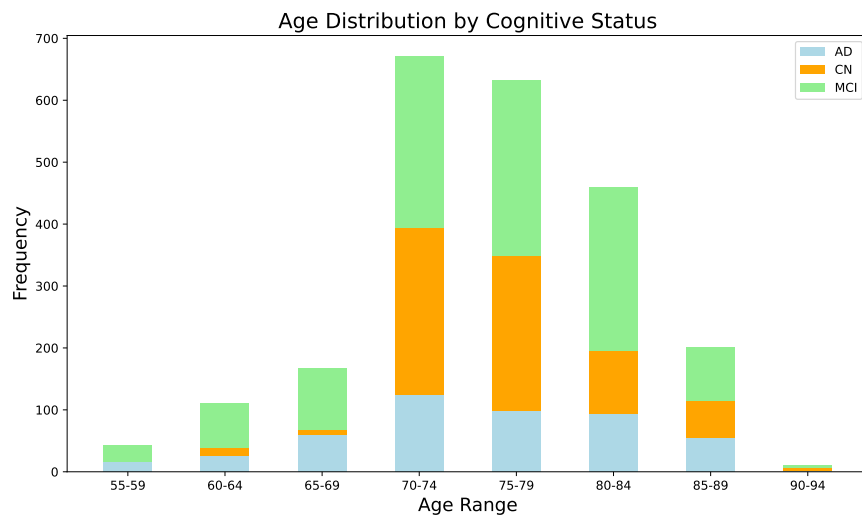
(b) Age Distribution



(c) Cognitive Status By Gender



(d) Age Distribution By Gender



(e) Age Distribution by Cognitive Status

Figure 15: Initial EDA Analysis Figures.



**ResNet-50 Metrics:**

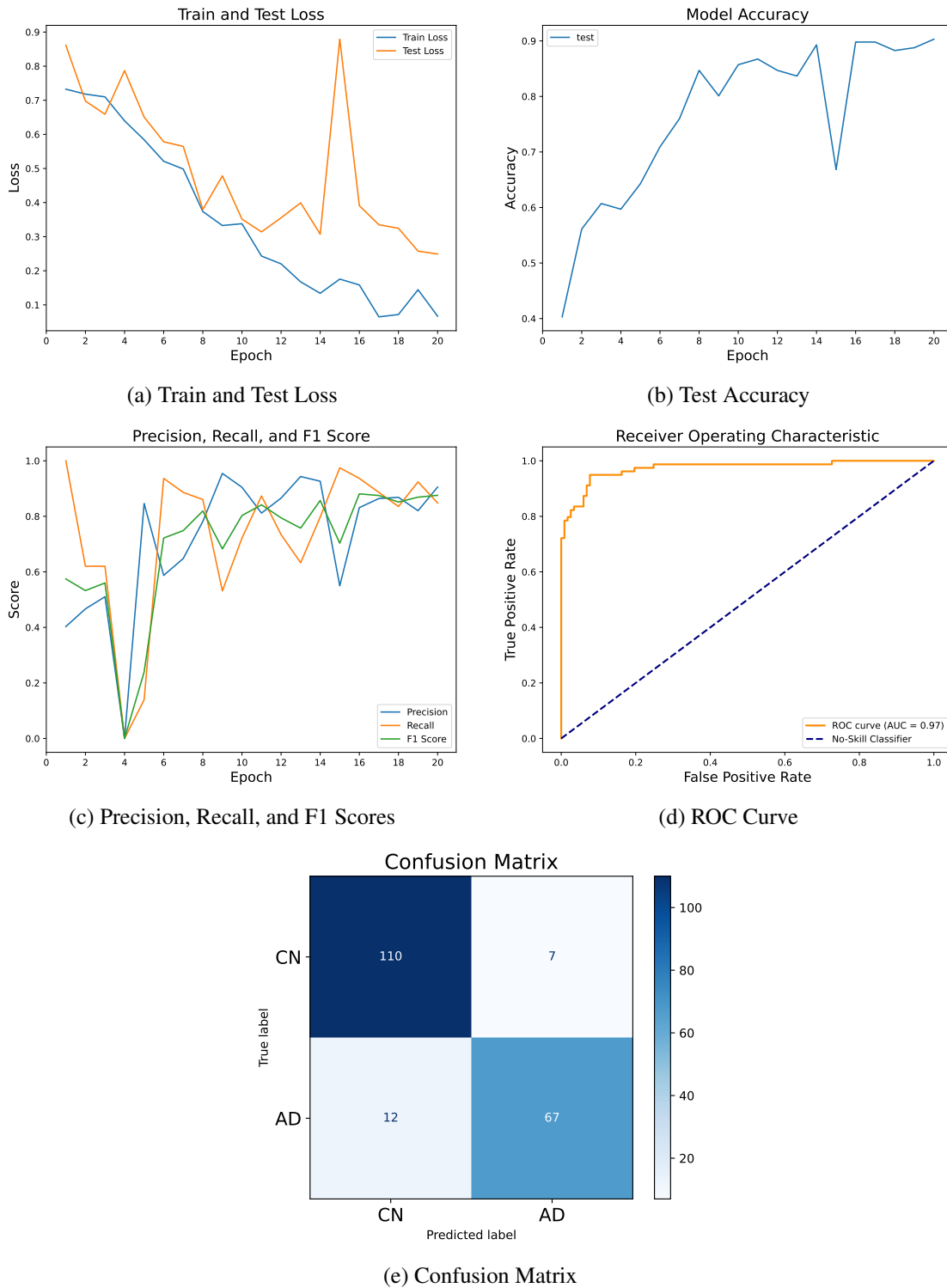


Figure 16: Evaluation Metrics ResNet-50.

## 11 Appendix B

### Dataloader and Data Transforms python code:

```

1  ad_dir, cn_dir = os.path.join("data/AD"), os.path.join("data/CN")
2
3  def get_file_list(data_path, label):
4      return [{"image": os.path.join(data_path, f), "label": label} for f in
5              ↪ os.listdir(data_path) if f.endswith(".nii")]
6
7  data_list = get_file_list(ad_dir, 1) + get_file_list(cn_dir, 0)
8  np.random.shuffle(data_list)
9
10 # Split into training and validation sets (20/80 split)
11 n_val = int(len(data_list) * 0.2)
12 train_data, val_data = data_list[:-n_val], data_list[-n_val:]
13
14 # Define basic transforms (no augmentation)
15 train_transforms = Compose([
16     LoadImaged(keys=["image"]),
17     EnsureChannelFirstd(keys=["image"]),
18     Orientationd(keys=["image"], axcodes="RAS"),
19     CenterSpatialCropd(keys=["image"], roi_size=(50, 96, 96)),
20     Spacingd(keys=["image"], pixdim=(1.0, 1.0, 1.0), mode="bilinear"),
21     ScaleIntensityd(keys=["image"]),
22     EnsureTyped(keys=["image"]),
23 ])
24 val_transforms = Compose([
25     LoadImaged(keys=["image"]),
26     EnsureChannelFirstd(keys=["image"]),
27     Orientationd(keys=["image"], axcodes="RAS"),
28     CenterSpatialCropd(keys=["image"], roi_size=(50, 96, 96)),
29     Spacingd(keys=["image"], pixdim=(1.0, 1.0, 1.0), mode="bilinear"),
30     ScaleIntensityd(keys=["image"]),
31     EnsureTyped(keys=["image"]),
32 ])
33
34 # Create datasets using CacheDataset for caching
35 train_ds = CacheDataset(data=train_data, transform=train_transforms)
36 val_ds = CacheDataset(data=val_data, transform=val_transforms)
37
38 # Create DataLoaders
39 train_loader = DataLoader(train_ds, batch_size=4, shuffle=True, num_workers=4)
40 val_loader = DataLoader(val_ds, batch_size=4, shuffle=False, num_workers=4)

```

Listing 3: ResNet-50 dataloader and transforms python code.

**Training loop python code:**

```
1 epochs = 20
2
3 for epoch in range(epochs):
4     print(f"Epoch {epoch + 1}/{epochs}")
5
6     # Training phase
7     model.train()
8     epoch_loss = 0
9     with tqdm(train_loader, desc=f"Training Epoch {epoch+1}/{epochs}",
10             ↪ unit="batch") as pbar_train:
11         for batch in pbar_train:
12             inputs, labels = batch["image"].to(device),
13             ↪ batch["label"].to(device)
14             outputs = model(inputs)
15             loss = loss_function(outputs, labels)
16             optimizer.zero_grad()
17             loss.backward()
18             optimizer.step()
19
20             epoch_loss += loss.item()
21             del inputs, labels, outputs
22             gc.collect()
23             torch.cuda.empty_cache()
24
25             pbar_train.set_postfix(loss=loss.item(), epoch_loss=epoch_loss /
26             ↪ len(train_loader))
27
28     avg_train_loss = epoch_loss / len(train_loader)
29     print(f"Training loss: {avg_train_loss:.4f}")
```

Listing 4: ResNet-50 training loop code.

**Test loop python code:**

```
1     # Validation phase
2     model.eval()
3     val_loss = 0
4     val_correct = 0
5     total_samples = 0
6     all_val_labels, all_val_preds = [], []
7     with torch.inference_mode():
8         with tqdm(val_loader, desc=f"Validating Epoch {epoch+1}/{epochs}",
9             ↪ unit="batch") as pbar_val:
10            for val_batch in pbar_val:
11                val_inputs, val_labels = val_batch["image"].to(device),
12                ↪ val_batch["label"].to(device)
13                val_outputs = model(val_inputs)
14
15                val_loss += loss_function(val_outputs, val_labels).item()
16                val_correct += (val_outputs.argmax(dim=1) ==
17                ↪ val_labels).sum().item()
18                total_samples += val_labels.size(0)
19
20                all_val_labels.extend(val_labels.cpu().numpy())
21                all_val_preds.extend(val_outputs.softmax(dim=1).cpu().numpy()[:,
22                ↪ 1])
23
24                pbar_val.set_postfix(val_loss=val_loss / (total_samples + 1),
25                ↪ accuracy=val_correct / total_samples)
26
27            avg_val_loss = val_loss / len(val_loader)
28            accuracy = val_correct / total_samples
29            print(f"Validation loss: {avg_val_loss:.4f}, Accuracy: {accuracy:.4f}")
30
31            if accuracy > best_metric:
32                best_metric = accuracy
33                best_metric_epoch = epoch + 1
34
35        print(f"Training complete. Best accuracy: {best_metric:.4f} at epoch
36        ↪ {best_metric_epoch}")
```

Listing 5: ResNet-50 test loop code.