Volumetric Medical Classification using Deep Learning

A comparative study on classifying Alzheimer's disease using Convolutional Neural Networks

Presented by:
Richard Masson

Supervisors:
Frederick Nicolls & Jarryd Son
Dept. of Electrical and Electronics Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in fulfilment of the academic requirements for a Master of Science degree in Electrical and Computer Engineering

July 2, 2023

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.
5. I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own. This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.

Signature:..........................
R.D.A Masson

Date:.............................
July 2, 2023

Acknowledgments

This dissertation would not have been possible without the help of all those that have supported me over its lifespan. Immense gratitude goes out to my supervisors, Fred Nicolls and Jarryd Son. It was with their guidance and support that I was able to turn an idea into a true academic work. I'll always be grateful for our meetings where you helped me to make sense of all the thoughts in my head.

I'd also like to thank the University of Cape Town for facilitating the dissertation, and for providing me with the space and utilities to go about working on it. Special thanks must be given to the UCT High Performance Computing Facility, without which I would not have been able to run most of the resource-heavy computations.

Finally, I want to thank the loved ones, family, and friends who supported me every step of the way. Whether it was reading through early drafts, or motivating me to keep going, every contribution helped to make this work a reality.

Abstract

This work sets about designing and implementing a number of deep-learning models capable of identifying Alzheimer's disease from MRI brain scans. A common problem with detecting the disease is the difficulty in doing so before outward mental symptoms have begun to show. Therefore, the models attempt to classify both mild and severe cases. The experimental process proves that a problem involving volumetric medical images benefits from the usage of 3D model architecture over traditional 2D architecture. In doing so, however, it is revealed that the 2D models do ultimately perform only slightly below the 3D model. Thus, the 2D approaches hold merit for potential usage, should a 2D planar approach be desired. The paper presents a total of three models. The first is a 3D CNN model, which performs the best in all regards, with a mean accuracy of 81.3%. It is treated as the optimal means of detecting Alzheimer's. The second is a 2D CNN model which uses separate 2D convolution layers to independently train and combine 2D slices across the depth axis. This approach produces a model that only slightly under-performs compared to the 3D model (80% accuracy). The third and final model is a novel design in which a set of models are each trained on a single unique 2D slice of the volume, across a carefully chosen range of slices deemed to contain the most favourable feature data. The model set is then used in unison to make predictions which are then aggregated using a weighted ensemble-voter to produce a final prediction score. This final design scored between the prior two models (80.6%), and establishes itself as a promising model capable of operating on a fraction of the data. Analysis of the models' activation gradients was conducted to confirm that 2D models are able to train well on isolated 2D slices, but struggle to process the space between these slices. Additionally, the work examines and rates the effectiveness of several optional variables in the overall CNN model design, specifically in the context of training on brain scans. A variety of pixel rescaling functions were found to have a noticeable positive impact on overall model performance. Regularization, as well as augmentation in the form of rotation / elastic deformation, also yielded similar improvements on such models, and are thus universally recommended as considerations for any works attempting to solve a similar classification problem. With all this in mind, a final conclusion is made that machine learning and deep learning are promising tools in the medical field for assessing and diagnosing using raw brain scans.

For additional reference, the code repository for generating and processing the models is available for viewing. An alternate branch, containing the code used to produce the gradient activation maps, has also been included.

Contents

Chapter 1: Introduction

This chapter provides a more detailed introduction to the study. It gives the background and origin of the research question, the core objectives of the work, and the tasks conducted to complete these objectives.

1.1 Background to the study

The rising prevalence and use of machine learning in the modern world has led to its usage in various professional fields. In medicine specifically, machine learning architectures such as deep learning have opened up opportunities to automate the classification of medical imagery. In radiology, radiologists are expected to manually sift through every scanned image in order to make a particular diagnosis. However, this usually requires them to know what they're looking for, and is a costly endeavour. Deep learning models, if properly configured, could be used to automatically detect target symptoms within a digital image, while requiring only minimal feature pre-processing. The question then becomes: in a field where image format, lighting and target diagnosis vary from scan to scan, what is the best design to use for your learning model? Specifically, this work asks this question in the context of classification problems, and investigates a model best suited to assess a raw input image and determine which class of diagnosis the patient would fall under.

The problem presents two paths for accepting input images: 3D volumetric imaging (such as an MRI scan of a brain) versus 2D flat images (such as X-rays). Traditionally, machine learning application was done using 2D data-transformation techniques, regardless of the original shape of the input image. Examples of these older algorithms are the support vector machine and linear regression model. However, newer neural network-based designs have made it significantly easier to train on 3D inputs, thanks to processing options such as 3D convolution layers. Therefore, the question arises of whether 3D volumetric medical images should now exclusively be handled using contemporary 3D architecture, or if there is still value to be found in 2D-based models augmented to process 3D images. In this context, “3D architecture” refers to models designed to accept 3D inputs, and which use processing steps that take advantage of the interconnectivity between all three axes. An example would be a 3-dimensional convolution kernel. “2D architecture” refers to approaches that instead attempt to transform the input data into a 2D representation so that 2-dimensional algorithms can be applied — noting that a portion of the cross-dimensional feature data is ignored in this process. However, if this cross-dimensional data could be proven to be mostly redundant to the learning process, its removal would have the benefit of significantly reducing the computation complexity of the image processing.

One problem that can benefit greatly from such an investigation is that of identifying dementia — specifically Alzheimer's disease — in brain scans. Alzheimer's disease is a severe form of dementia and is a degenerative condition that impairs an individual's cognitive capacities and eventually leads to death. It has been cited to be afflicting over 44 million individuals globally, and in the US is listed as the sixth-leading cause of death above a certain age. Alzheimer's primarily only afflicts individuals above the age of 65, and presents noticeable behavioural symptoms in the form of memory loss and decreased mental acuity. However, the issue lies in the fact that once these behaviours begin to show, the disease is often already in its late stages. As there is no definitive "cure" for Alzheimer's, it is an affliction where detecting it in its earliest stages is absolutely critical. If caught early, patients could be given time to take preventative measures. While the results still require further research, papers such as the one by Silva et al.[20] have demonstrated that basic lifestyle adjustments, such as changes in exercise, diet, or mental stimulation, can mitigate or slow the rate of Alzheimer's development in some individuals. Beyond this, there is value in simply giving patients time to emotionally prepare for the possibility of the disease worsening. For this reason, alongside its high affliction rate, it is a medical issue that stands to benefit a great deal from machine learning. While the diagnosis can be made via manual assessments from an expert, an automated model would allow for such assessments to be conducted far more often. Automated tests could be done regularly after a certain age, and possibly even in the background when brain scans are taken for completely unrelated reasons, in an attempt to detect the disease sooner.

Brain scans are imaged in 3D in order to capture the full volume of the brain. This serves as a perfect problem space to compare 3D and 2D model performance, where the specific goal in mind for the models is identifying and classifying whether a given scan comes from a patient with Alzheimer's. Silva et al.'s work mentions how Alzheimer's is understood to be closely linked with the development of a protein plaque in the patient's brain — with this in mind, this study holds the belief that the presence of this plaque means that trainable feature data can be found within the tissue of these brain scans.

1.2 Objectives of this study

The main objective of the study deals with a comparative evaluation of 2D versus 3D based training models, in the context of identifying and classifying Alzheimer's in brain scans. Observations are made on whether the models are able to accurately predict negative cases (cognitively normal patients), versus positive (some stage of Alzheimer's). The primary metric for testing is the accuracy of the model, evaluated on a held-out set of unseen data, alongside the loss score, which serves to further inform the assessment. Doing so shall prove the superiority of 3D model architecture when working with volumetric data. It will also highlight methods that achieve comparable results with 2D architecture. It is important to note is that producing a near-perfect model is not the main objective of this study. Rather, the model's effectiveness serves to act as a proof of concept, as well as aid in making comparisons between multiple factors that may improve or hinder performance.

The work will also highlight and test several other variables involved in designing such a classification model, such as pre-processing options, augmentation and regularization. These variables, while not specific to a 3D versus 2D discussion, are still important pieces to consider when developing a solution to the Alzheimer's detection problem. The discussion surrounding them, specifically in the context of dealing with brain scans as input data, also offers interesting discoveries. The four main findings that will be presented are:

1. The importance of selecting the right function for intensity scaling and normalization of the image pixel values.
2. The benefits of applying two types of augmentation to the data: rotation and elastic deformation.
3. A discouragement in the usage of skull stripping, due to its negative impact on model performance as well as to its heavy implementation cost.
4. The positive effect that regularization has on the models in this work.

Finally, all prior findings are considered to make a summary evaluation of machine learning's role in the medical domain, in terms of offering automated classification models. This evaluation is made partly in the context of the Alzheimer's problem, and partly in a generalised sense, and will assert that these models are more than capable of producing results and accuracies that warrant practical usage. However, the discussion will also admit that there is some margin for error, and therefore practical application would need to be done in conjunction with medical experts in order to produce the most reliable solution.

1.3 Problems to be investigated

This section provides a quick overview of the core topics to be addressed via the chosen experimental procedure:

- Comparing the performance of 3D-based models to similar 2D ones. The 3D Convolutional Neural Network (CNN) architecture is found to perform the best, though two approaches to a 2D model are presented, with performances only marginally below that of the 3D model.
- Observing the layer gradient activations of the various models, and observing the changes in the values based on the model, as well as the class of image predicted on. From this, it would be discovered that the 2D models do face difficulty in processing the space between axial slices. The visual differences between Alzheimer's-positive and negative cases are found to be too subtle for the untrained eye to differentiate.
- Exploring the pre-processing steps, and assessing their impact on model performance. In some cases, comparisons are tested for several preprocessing variants. This applies to the rescaling and normalization, skull stripping, and augmentation of the input data.
- Exploring the performance of all of the above when dealing with greatly limited data. Medical imagery data is often extremely limited, and as a result processing/ training designs that mitigate this are explored. This includes regularization steps taken to reduce over-fitting, and utilisation of a second dataset for full-scale testing on unseen data.

Chapter 2: Literature Review

This chapter highlights the literature reviewed during the development of the core design of this work. This is broadly split into three topics. The first is the design of the model – both the type used, and as its overall architecture. The second is the impact that certain pre-processing of the input data has on model performance. The final topic is the datasets available, and which set is best suited to this work.

2.1 Model Design

To begin with, the paper by Bratic et al. [2] was assessed. This paper presents summaries of several other academic works, all attempting to develop a training model for cognitive brain conditions. This acts as an overview of the possible approaches one could take for this problem. As the papers are all listed alongside the performance of their models, this made for an efficient way of determining which approaches yield the best results. There are four individual papers present in the summaries that cite accuracies at 90% or higher. The top listed paper by Patil [14] claims to have over 95% accuracy using an Artificial Neural Network, but the paper does not detail its implementation steps, nor its experimental process. This unfortunately makes it difficult to draw useful information from. The second paper, by Payan [15], offers a robust 3D-CNN showcase, with favourable results (87+% across three different classification problems) — this paper will be discussed in detail below. The other two papers, by Liu [12] and Lopez [13], offer novel designs worth mentioning: Liu's approach achieves 89% accuracy using 2D slices as inputs to the learning model, with the aid of ensemble-voting. The Lopez paper reports a much higher accuracy than any other SVM-based approach (90%), and shows that such an approach can produce favourable results in the right conditions. However, the model uses the SPECT dataset, which deviates from most other literature on this topic, and is not dealt with in this work. Because of this, Lopez's strategy was not pursued further. Reviewing these works, as well as the other models listed, the first conclusion made was that approaches utilising older learning techniques, such as Bayesian Gaussian models, logistic regression, or support vector machines, seem to generally perform worse than many examples utilising artificial neural network and CNN-based approaches. Several other papers utilising such models were found to also produce favourable results [11][22][23][24]. The second observation was that the two most commonly used datasets were the Alzheimer's

Disease Neuroimaging Initiative (ADNI) set, and the Open Access Series of Imaging Studies (OASIS) set. This observation became more relevant later when selecting a dataset.

The Payan paper [15], mentioned previously, offers valuable information regarding the implementation of a CNN model, specifically for the ADNI dataset. It also discusses the advantages and disadvantages of using a 3D convolution model versus a 2D one, which is especially relevant to this dissertation. The paper sets up the experiment by constructing two different models, one taking in 3D patches of the brain scans as input data and using 3D convolutional layers, and the other taking an assortment of random 2D scan patches. The layer dimensions are also fine-tuned to ensure an equal number of output nodes for each case. The results of this experiment find the 3D input/layer approach to be superior (89% accuracy in a 3-class problem versus 85% for the 2D approach). This paper also presents a novel approach of evaluating using three different class setups: CN (cognitive normal) versus AD (Alzheimer's disease), CN versus MCI (mild cognitive impairment), and all 3 classes. This approach immediately stands out as more informative, as it demonstrates the potential unequal difficulty the model can experience in identifying some classes versus others. Payan reports the classification of CN versus AD to have an accuracy almost 10% higher than CN versus MCI, which makes sense under the assumption that more severely diagnosed dementia would display more visible patterns of disease than milder cases. Finally, this paper employs a sparse auto-encoder model for feature extraction. However, this strategy does not come up in any other examined literature, and the paper does not offer evidence or discussion on the benefit that the autoencoder has on overall performance.

A paper that makes similar conclusions on a 3D CNN implementation, while also going into significantly more detail on the different possible design options, was published by Wen et al. [23]. Experiments conducted in the paper show several different approaches to training an Alzheimer's classification model, for both the ADNI and OASIS sets. Four models are assessed: a 3D subject-level model (using the entire volumetric image at once as input), a 3D ROI-based model (where a patch containing only the hippocampus area of the brain was used), a 3D patch-level model (using many random 3D patches of the model), and a 2D slice-based model (which attempts to utilise 2D model architectures). All three 3D approaches outperform the 2D one, with similarly equal scores all in the 80 percentiles, versus the 2D model in the 70s. This falls in line with Payan's findings on the effectiveness of 3D model designs. The paper details the architectures of each model in a replicable manner. Furthermore, a novel implementation of ensemble-voting is covered in conjunction with the 3D patch-level and 2D slice-level models. Effectively, a hard-voting classifier is used to aggregate the predictions of multiple trained models (where each model is trained on a different patch/slice for a given input scan). The voting algorithm uses the validation accuracy scores as weights in order to place priority on the models trained on the most valuable data. The results of this paper also show that the ADNI dataset yields better results than OASIS, which lines up with findings in other papers [24].

The papers covered so far have either based their CNN architectures on well-known models such as VGG-net [24], or used a large number of stacked convolution and pooling layers [23]. In contrast, the experiments conducted by Khagi et al. [11] attempt to assess the impact that the complexity of a model can have on performance. The paper attempts to classify Alzheimer's within brain scans using 3D CNNs with an increasing number of convolution layers — from 2 layers up to 6. While more complex models show a higher training accuracy (admittedly only from 99% to 100%), the testing accuracy instead peaks at 4 convolution layers (approximately 96% compared to 92% at 6 layers). The paper further discusses how over-fitting is a major problem faced in a deep learning task like this. It stipulates that one must consider the complexity of the model — that over-complicating the model can increase the tendency towards over-fitting. This is exacerbated in a medical classification problem where there is minimal data available and models are already prone to over-fitting.

2.2 Pre-processing

For pre-processing of the image data, there appear to be a few steps commonly taken by a majority of research papers. In the case of papers using CNN models, the overall processing of the images is still minimal, as deep learning models typically do not require extensive pre-processing, but there are still a few considerations. Several papers crop down the image data, as there is empty space present around the actual brain in most MRI scans. In the previously-mentioned Wen [23] paper, the ADNI images are uniformly cropped to a dimension of 169×208×179, whereas the Yagis paper [24] lists a dimension of 182×218×182 as viable. Skull stripping — the process of mapping the areas of an MRI scan containing only brain-matter in order to produce a mask free of skull data — is also worth considering. It is employed by several papers [19][23][24] as a pre-processing step. However, the Wen paper performs a comparative evaluation on the impact of skull stripping on the training performance and finds it to have only negligible benefit (1% accuracy increase). While sometimes a non-zero performance increase can be worth it, one must consider the cost involved in implementing a skull stripping algorithm.

The Wen paper also includes several other pre-processing experiments. Alongside skull stripping, bias-field correction and non-linear registration are deemed to have negligible effects on model performance. On the other hand, intensity rescaling (rescaling all voxel values to lie between 0 and 1 based on minimum and maximum values) is described as having a massive effect on performance. Without the rescaling function, the results show none of the models able to train beyond a 50% accuracy. When included, the accuracy is shown to increase by 30%. The prevalence of intensity rescaling in most of the other papers explored further validated these results. For this reason, special attention must be given to the intensity rescaling step of pre-processing when dealing with such a classification problem.

The type of volumetric image used as training data also varies between works, and is relevant to the discussion. While both the ADNI and OASIS datasets offer a wide variety of image types, the most commonly chosen type to use for demenetia classification are T1-weighted MRI scans [14][19][23][24]. Even outside the context of brain scans it seems that this type of scan is preferred for volumetric classification, as shown in one paper on pelvic tomography classification [7]. In the Khagi paper [11], an experiment is conducted comparing the performance of models trained on MRI data versus PET scan data. The MRI-trained model is shown to yield accuracies 30% higher than that of a PET-trained model.

2.3 Dataset

From the papers observed by this point, it was clear that the two most commonly accepted datasets were found to be the ADNI set, and the OASIS set. The ADNI set sees the most usage [11][12][15], though some papers do still use the OASIS set [14], or a combination of both [23]. Both datasets are viable options in that they are freely available and well-organised collections of labelled MRI data. Though, across the observed papers, those using ADNI were found to perform better. This is backed up in detail within the Wen paper, as well as in the paper by Bansal et al. [1]. The Bansal paper comes to this conclusion after assessing several papers using either the ADNI or OASIS sets.

Chapter 3: Design

This chapter covers the core design decisions relating to the model development for this dissertation. First, it presents a general overview of the Alzheimer's classification problem, and what requirements a model would need to meet in order to properly solve it. Then, the data-processing pipeline is presented, which covers the steps taken to collect and prepare the data for training and testing. This will showcase the dataset used, how the data is extracted and assigned class labels, as well as the preprocessing steps taken in order to prepare the images for fitting on a model. Following this, the chapter details the design for each of the models created. This covers three model designs: a 3D CNN model, a 2D CNN model trained on a 3D volume, and a design utilising a collection of 2D CNN models trained on individual 2D input slices. Finally, some choices made relating to specific hyperparameters common between all models are outlined, as well as a gradient-activation visualisation experiment. The gradient experiment was used in order to visualise the trained patterns in the models. This chapter presents the design information in a generalised form, such that it could be replicated in any given programming language. The subsequent chapter on implementation will cover the exact means by which certain subsystems were programmed in this work specifically.

3.1 Overview

This work attempts to find means of correctly identifying the presence of Alzheimer's disease within a brain scan. A primary consideration in dementia detection is detecting it before a patient becomes symptomatic, and as such only labelled image data was utilised for training purposes. As dementia can be detected using purely visual patterns within a scan, this paper utilises a pure machine-learning approach in an attempt to train a model that can determine these patterns without additional instruction. Specifically, the model would adopt a deep-learning CNN architecture. Deep-learning models typically require less feature data when training, making them the best fit based on the decision to work with only raw labelled image inputs. This dissertation does not deal with the actual logic and reasoning behind the visual features, as this is better left to literature in the medical field.

Research has shown that Alzheimer's is traditionally classified as one of three states:

cognitively normal (negative case), mild impairment (positive case — early stage of the disease where cognitive performance is impaired, but symptoms may still go unnoticed), and Alzheimer's disease (positive case — late stage diagnosis). Because of this, the problem can be tackled as a classification problem, in which the model will need to properly predict which class a given brain scan falls under. This will be assessed in 2 forms, cognitive normal versus mild impairment, and cognitive normal versus Alzheimer's disease. Cases involving mild impairment versus Alzheimer's disease were not considered as the delineation between two stages of a positive case was deemed not of critical value. Cognitive normal versus Alzheimer's was expected to yield the best results as the two classes have the largest difference, and this was proven to be correct. Attention was still given to cognitive normal versus mild impairment, as it represents the most common practical application of detecting the disease early on while cognitive symptoms are still minor.

3.2 Data Pipeline

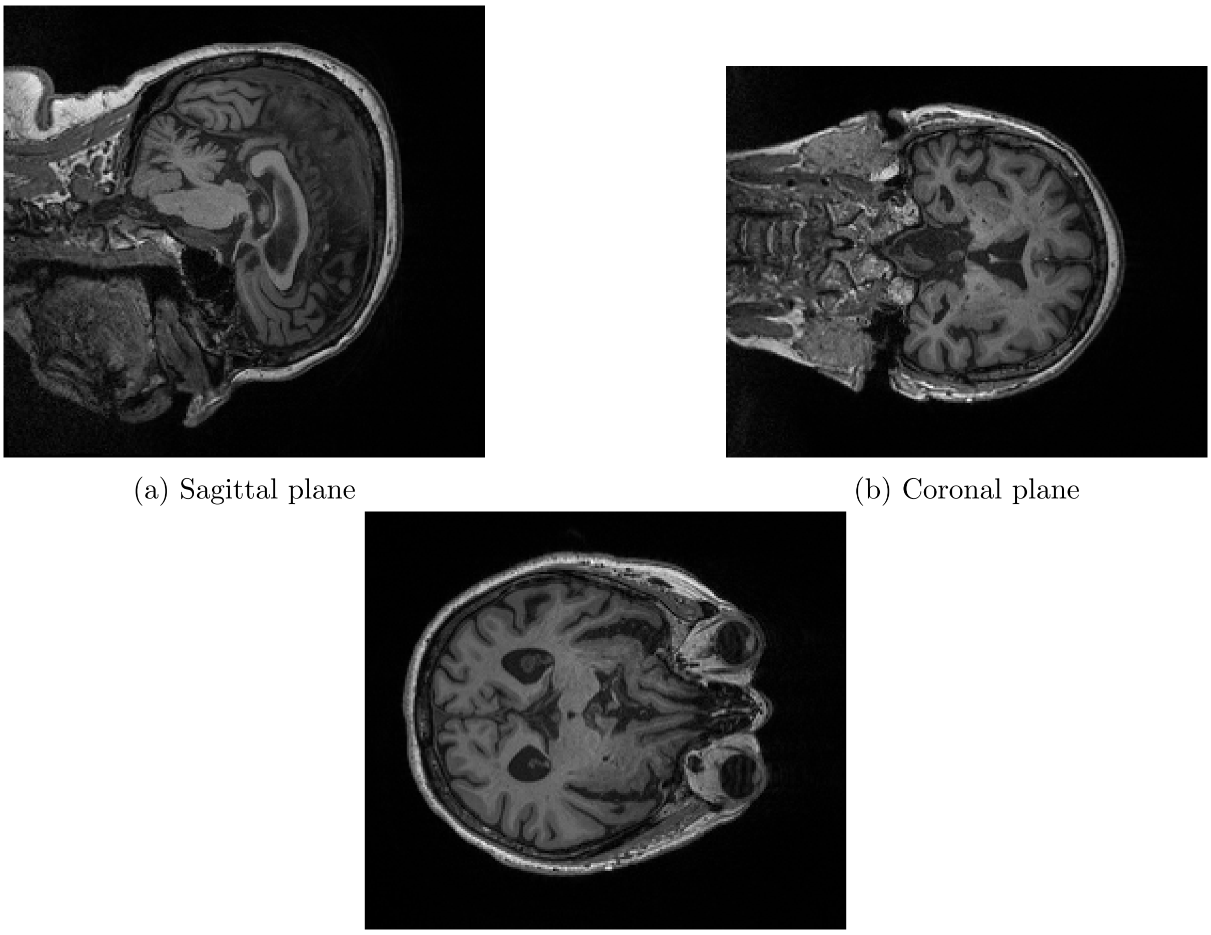
The first step involves collecting and preparing the data so that it is ready to be fit into the desired model for training. An outline is provided on the pipeline that the data takes from its initial collection until that point of fitting. This provides a replicable process, should one decide to attempt to expand upon this work, and also highlights the areas of variable design. Variable design refers to sections of the pipeline that have no predetermined optimal setup, and warrant experimentation in order to report their impact on model performance. This pipeline remains the same for both 3D and 2D architectures, as the 2D models possess the transformation functions necessary to accept the 3D data. These functions will be detailed in a later section.

To make the pipeline easier to follow, it has been split into three subsystems: data collection, data partitioning, and image processing.

3.2.1 Data Collection

The literature review showed the Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset to yield the best training results, and thus it serves as the central dataset for this work. The dataset has been verified by previous works, and thus it can be safely assumed that all samples are accurately labelled.

MRI data was chosen as the universal input data-type. While MRI is not the only type of 3D brain imaging available, it yields the most favourable results in similar literature. The literature showed the highest degree of reliability and favour when using T1-weighted, 3D Sagittal scans. Therefore, this was the type of image extracted from the chosen datasets for this work. An example of this type of scan, from the ADNI set, is presented in fig. 3.1. Some experiments were also conducted in order to confirm the discouragement of using skull stripping that can be found in other works [23]. For these experiments, skull stripping was performed on the dataset using available modules, and the resultant images were saved as a separate set for future testing.



(a) Sagittal plane

(b) Coronal plane

(c) Axial plane

Figure 3.1: Multi-plot showing the three plane views of a sample bearing the "Alzheimer's" classification

Finally, every image intended for use needs a label. As mentioned earlier, Alzheimer's class labelling is typically comprised of Cognitive Normal (henceforth CN), Mild Cognitive Impairment (MCI), and Alzheimer's Disease (AD). In the medical field these are assigned values of 0, 0.5 and 1 respectively, however for application in training this can be simplified to assign labels of 0 and 1. The 0 label is used exclusively to refer to the CN class, and

1 is used for any positive-case class — MCI or AD. All images within the dataset are pre-labelled, so the algorithm need only assign the correct label value pairing to each input image.

Table 3.1 shows the summarised count of each class of image in the ADNI dataset.

Table 3.1: Class count for the ADNI dataset.

Class	**Count**
CN	541
MCI	513
AD	154

3.2.2 Data Partitioning

At runtime, data needs to be stored in data structures such as arrays in order to facilitate the partitioning of the data between training, validation and testing. Labels are stored as simple integers, but the images take up significantly more memory. While the total image count is comparatively low compared to other machine-learning problems, the detailed 3D scans take up a massive amount of memory. During early implementation, it was impossible to load all the images into memory simultaneously without encountering out-of-memory issues. As a result of this, the storage design was significantly altered.

Firstly, text files were generated before runtime, containing the file locations of all the relevant images for any given experiment, alongside the label for that sample. During runtime, the file containing these location strings could simply be loaded in to act as proxies for the actual images. It would not be necessary to extract the actual image out of the sample until training time.

Once the location strings and labels are properly stored within lists, acting as our x and y variable sets, they could be split into training, validation and testing sets. Table 3.2 outlines the partition ratios chosen for this work. The majority of the data was assigned to training, as is common procedure in machine learning. The validation and test sets were not made too small, as the total images available were limited, and it was important that each subset possessed enough data for proper variety in the evaluation.

Table 3.2: Dataset split ratios using the ADNI set.

Partition	**Split**
Training	0.75
Validation	0.15
Testing	0.1

3.2.3 Image Pre-processing

Once the training, validation and testing sets are prepared, each subset is assigned to its own data loader. Each data loader pairs the image directory data with its label, and applies the relevant processing. This processing includes batching and shuffling, as well as mapping a function to the data to extract and process the image only when the data is indexed for the fitting function. By mapping the processing to the directory data, it ensures that each image is only loaded into memory when called during training, and is unloaded from memory directly after. Images loaded are also resized to a uniform shape, and have their pixel values rescaled to normalise the range distribution. Additionally, augmentation is applied to the training data. Augmentation not only simulates additional "new" images in an otherwise limited dataset, but also ensures that the trained model learns to adapt to scans with minor variations. The augmentation techniques chosen were rotation and elastic deformation, and the exact parameter values are shown later in the work. Once more, this processing is largely the same for both 3D and 2D models in the paper. These steps ensure that the full volume is available to the model, and it is up to the 2D model to further process the data depending on the design.

The mapped function itself was designed to do the following to each paired data-label element:

1. Use the location string as an input parameter to load in the relevant image data.
2. Convert the image data into an array of 32-bit float pixel values. This is down-scaled from the default 64-bit values in order to reduce data storage requirement,
3. Rescale and resize the image to the desired specifications.
4. Apply any chosen augmentation functions to the training data.
5. Return the augmented pixel array alongside the label, which is also converted to a 32-bit float.

The augmentation used the following procedure:

1. Rotation, up to 3 degrees in either direction, with each rotation using randomised magnitude and angle. The 3D model and the 2D subject-level model apply this across all three planes, whereas the 2D-slice model applies it over only two planes. During training this is applied with a 60% frequency rate.

2. Elastic deformation, with a randomised affine alpha range between 0 and 0.5. Each model has this transformation configured according to its relevant number of planes. During training this augmentation is applied with a 30% frequency rate.

For all sets, the loader shuffles the data and applies batching as well as pre-batching. Only the training data has augmentation applied to it, as standard practice involves leaving validation and testing data in its original state.

With respect to the image resizing, the decided-upon uniform shape for all images was 169×208×179. This is based on the Wen paper [23]. This was verified to successfully trim unnecessary space while preserving all true brain matter. A full visual summary of this pipeline, with respect to training data, is shown in fig. 3.2. For validation and testing images, the same pipeline occurs except for the augmentation step.

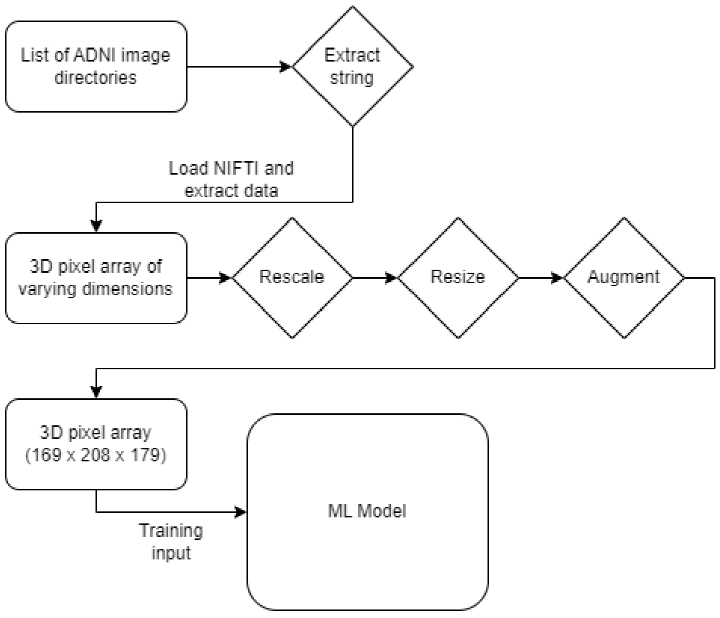

List of ADNI image directories
Extract string
Load NIFTI and extract data
3D pixel array of varying dimensions
Rescale
Resize
Augment
3D pixel array (169 x 208 x 179)
Training input
ML Model

Figure 3.2: Visual representation of the pipeline that a single image will take during model training.

3.3 Model Design

The discussion in this paper predominantly focuses on the development and experimental procedure surrounding a 3D convolutional classifier model. However, the comparison between it and possible 2D CNN implementations — ones that still use volumetric images as an input — is another focal point to the discussion. In the end, two 2D models were developed alongside the 3D model, to show different approaches that can be taken in order to try to solve the existing problem, as well as the pros and cons each offer alongside the original 3D approach. Therefore, this design section can be broken into three subsections: the core 3D CNN model, a 2D CNN model that attempts to put the full subject-level 3D image through a 2D architecture design, and a 2D CNN model that instead performs training on multiple 2D "slices" extracted from the original image. The 3D model's outline will contain most of the content that applies to all three of the models, whereas the 2D model design outlines will only cover their unique design differences.

3.3.1 3D Subject-level Model

The first design is a 3D CNN learning model, which aims to use a combination of layers, particularly convolution layers, in an attempt to create a network that best facilitates the learning of features. The features in this case are visual patterns within a brain scan that the model can leverage to correctly classify the dementia rating of a given subject. As a classifier, when the model trains it takes an image and an associated label as input, and produces an output value that represents the model's estimate of what label class the image would belong to. It then uses the known true value in a loss function to iteratively optimise a set of weights on each layer. This weight optimization is the basis of the supervised learning process.

The 3D model, as well as the other models, was designed with a sequential structure in mind, and thus comprises several sequential layers that activate in order. A visual summary of this design is shown in fig. 3.3. However, the architecture is best described by the following enumeration, which outlines these sequential layers in detail:

1. The input layer accepts an x-y pair of a 3D image and its associated label. The 3D image is treated as having a channel capacity of 1, as it is a grey scale image. Including the batch dimension, this makes the input image 5-dimensional. The labels are converted into binary matrices, which are better suited to multi-class

problems.

2. The input is fed through four convolution blocks. A convolution block comprises the following components:

   (a) 3D Convolution layer: This forms the core of the network, and is set to use a $5 \times 5 \times 5$ convolution kernel, as this has been shown to produce good results in similar networks. Padding is not utilised, as we are fine with the down-sampling this causes. The kernel is given a number of output nodes, which are set to increasing multiples of 8 per convolution block, as is common practice. Finally, the layer is given a ReLU activation function, as ReLU is the most commonly used activation function and has been shown to yield consistently beneficial results.

   (b) Ł2-type regularizer: Only used for the final two convolution blocks. This is applied to the weights of the prior convolution kernel, with lambda set to 0.01. This adds additional regularization to the model to combat over-fitting. L2-type regularization is preferred as it is better than L1-type at learning complex data patterns.

   (c) 3D max pooling layer: Pooling layers are implemented in order to further down-sample the data and reduce the number of trainable parameters in the model. Max pooling is specifically used here as it is best suited for images with dark backgrounds, such as medical scans.

3. The data is then flattened into the single dimension required for the subsequent layers.

4. The flattened data then passes through an intermediate dense layer using ReLU activation. The dense layers further process the output of the convolution kernels into the desired class predictions. The layer is given 128 output nodes.

5. Finally, the data arrives at the output layer. The output layer is another dense layer with a softmax activation function and output nodes equal to the number of classes present, which is extracted from the label set. Softmax is used in order to produce separate scores for each class. While the experiments conducted dealt only with two-class predictions, the design was built to accommodate multi-class predictions, such as for a hypothetical CN versus MCI versus AD case.

This design takes inspiration from Wen et al. [23], except it uses a somewhat simplified approach in which only four convolution blocks are implemented. While Wen at al.'s model yields very high accuracies (>85%), early tests utilising an exact replica of that

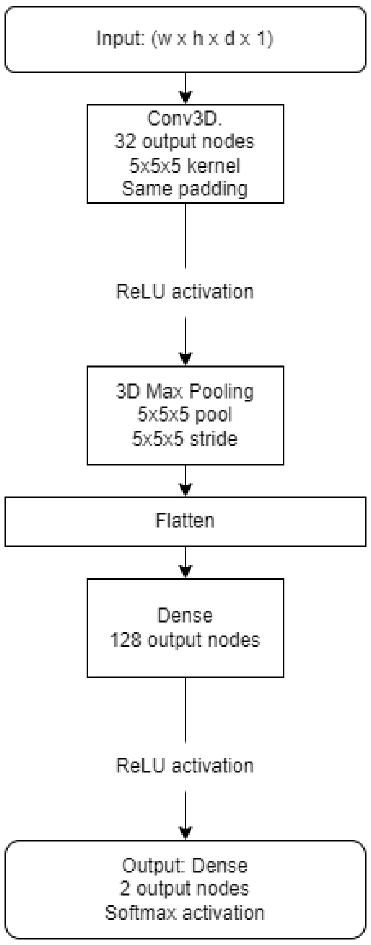


Figure 3.3: Chosen 3D model architecture design. Under each layer, the various parameters as well as output nodes are specified.

design evaluated significantly worse (55% accuracy). This implies that unless very carefully tuned, the complexity of the design can actually hinder the model's performance. In Khagi and Kwon [11], it is noted that a model with high complexity but low data size can produce poorer results unless fine-tuned with extreme precision. Ultimately, the design presented above was found to produce far better results using the chosen ADNI dataset, and thus it was used instead.

3.3.2 2D Subject-level Model

In this work, the first 2D model is referred to as a subject-level design, the reasoning being that the full 3D subject image is still loaded in as an input. However, the model utilises 2D convolution layers. The architecture of this model, which is summarised in fig. 3.4, is the same as the prior 3D model, except for the following changes:

1. 2D Separable convolution layer: Normally, this would not be possible as the 3D input image would have too many dimensions. However the design circumvents this by treating the third dimension as the channel layer. Whereas the previous model would take an input of shape $169 \times 208 \times 179 \times 1$, where the last dimension represents a grey-scale channel layer, this model treats the input as a 169×208 2D image with 179 channel layers. By doing so, one can ensure that the convolutions are conducted in complete isolation from each other, thus effectively differentiating the working of this design from the 3D model. A visual depiction of how these convolutions occur separately is covered in fig. 3.5. The figure shows how treating the volume as a multi-channel 2D plane still results in the full image being processed by the model. Furthermore, separable convolution is used. This is a specific convolution intended for separate depth-wise convolutions that can then be combined at the end. To achieve this, a 1 times convolution is set as the final combining kernel. A 1 times convolution has the effect of purely combining the individual 2D convolutions without performing any further calculation.

2. 2D Max pooling layer: A 2D pooling layer is necessary as the input images are now treated as 2D slices. This results in the down-sampling only occurring across two dimensions instead of three.

3. 4-dimensional input shape: As the depth is treated as the channel dimension, there is no need for a fifth channel of size 1.

3.3.3 2D Slice-level Model

Unlike the previous design, this approach attempts to solve the dimensionality problem by developing a model that captures the 3D volume by training an individual model for each axial slice of the image. In this work, axial is seen as the top-down axis of a brain scan, viewing the brain from above. The process is as follows:

1. A number of models are instantiated. Each model is identical to that of the prior 2D model, except that the channels are set to 1 and the 2D convolution layer is not separable. The full design per model is shown in fig. 3.6. The number of models instantiated is discussed below.

2. Each model is assigned a specific slice number.

3. During training, each 3D input image is split depth-wise into a user-specified number (discussed later) of 169×208 2D slices. For the given set of models, each

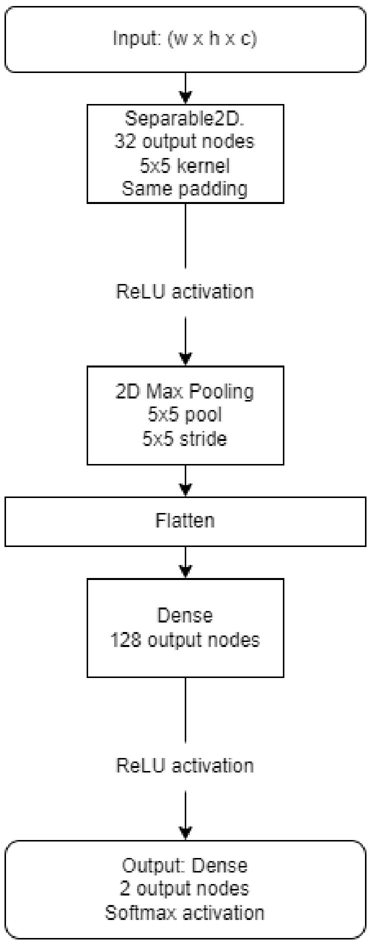


Figure 3.4: Chosen 2D subject-level model architecture design. Convolutional layers apply a kernel of size 3×3 and a ReLu activation function, each pooling layer has a kernel of size 2×2, and the output layer has as many output nodes as the number of classes.

model is trained on the slice of the number associated with it, alongside the label for the original 3D image. This results in an ensemble of models, each trained to deal with one specific slice depth. This ensures every slice is convolved and weighted in complete isolation from the others.

4. When a prediction is made, the image is split into slices and the relevant slices are fed into their associated models.

5. The predictions from each slice model are then combined using an ensemble voting algorithm, which aggregates the predictions. This effectively combines the slices together while allowing for a specific weighting per slice.

This design is similar to the previous one in that the axial slices are convolved in isolation. However, the major difference in this approach lies in the ability to use only a subset of

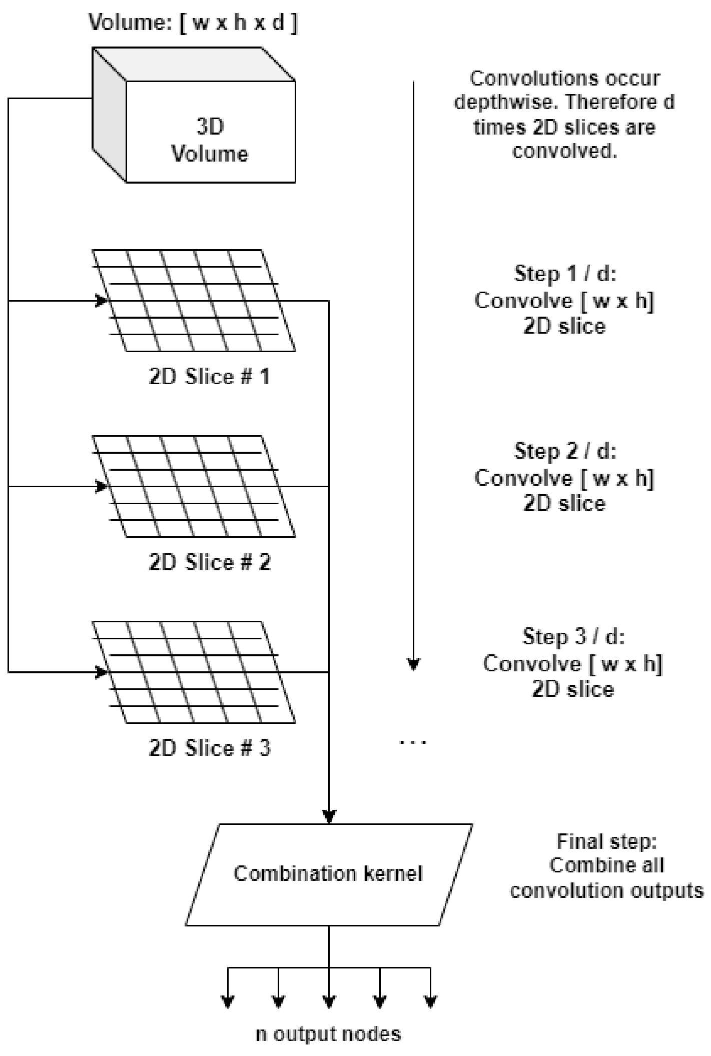

Volume: [w x h x d]
3D Volume
Convolutions occur depthwise. Therefore d times 2D slices are convolved.
Step 1 / d: Convolve [w x h] 2D slice
2D Slice # 1
Step 2 / d: Convolve [w x h] 2D slice
2D Slice # 2
Step 3 / d: Convolve [w x h] 2D slice
2D Slice # 3
...
Combination kernel
Final step: Combine all convolution outputs
n output nodes

Figure 3.5: Illustration of how 2D convolution is applied across channel slices, and how these individual convolutions can still be used to represent the full volume.

slices rather than the full volume, as well as the ability to weight slices different through the voting algorithm. A visualisation of this process is shown in fig. 3.7.

Attempting to train a full model, across numerous epochs, for every single slice in the range proved to be immensely time-consuming. Based on the previously-noted correlation between Alzheimer's and protein plaques, it was hypothesised that this protein-based feature data could lie in localised clusters within the most tissue-rich sections of the scan. This opened the possibility that most of the feature data could be present in only a subset of the slices. Thus, a novel approach was designed to further narrow the training down to only a subset of feature-rich slices. To find these slices, an initial trial training run with fewer epochs was conducted on the ADNI dataset, for all the slices between 50 and 100. Then, slice models that scored validation accuracy above 60% were recorded as "priority slices". As a result of this, the model could be implemented by only training on

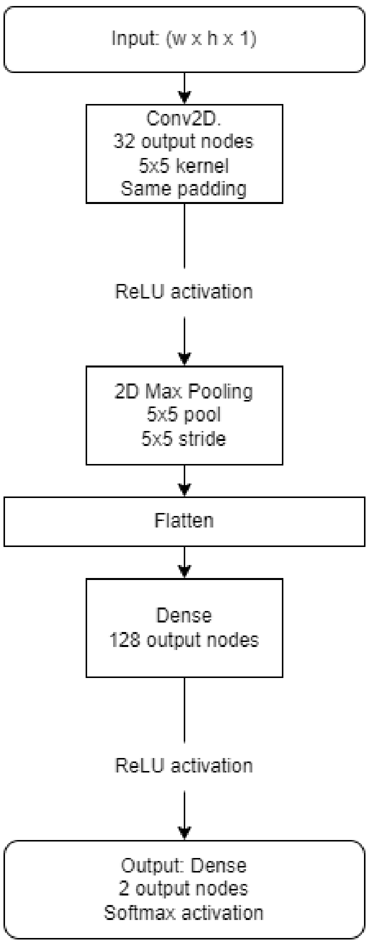


Figure 3.6: Figure showing the chosen 2D slice-level model architecture design. Each convolutional layer applies a kernel of size 3×3 and a ReLu activation function, each pooling layer has a kernel of size 2×2, and the output layer has as many output nodes as the number of classes.

these priority slices, while discarding the less useful ones. While this does not account for slight variations in scan placement across samples, this approach was still able to produce favourable results.

The experiments in this work used the following slices: [56, 57, 58, 64, 75, 85, 88, 89, 96]. Some images showing examples of priority slices can be seen in fig. 3.8.

For the ensemble voting, the two options available are hard voting and soft voting algorithms. Hard voting takes the modal prediction from the model array. Soft voting produces a weighted average of the predictions. Some slices can be safely assumed to contain more feature data than others, such as comparing slices in the centre of the brain tissue to those on the periphery. Therefore, soft voting is the better approach, as it allows for more informative slices to be weighted higher than others. To do this, the validation accuracy recorded during training of each model is used as its own weight — this means

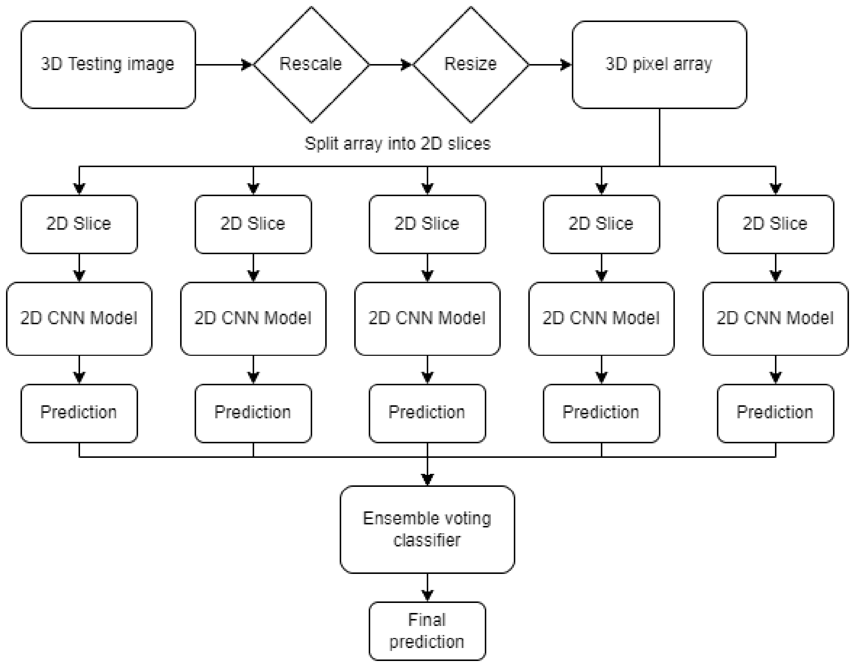


Figure 3.7: Figure showing the 2D slice pipeline that a sample 3D scan goes through in order to produce a prediction.

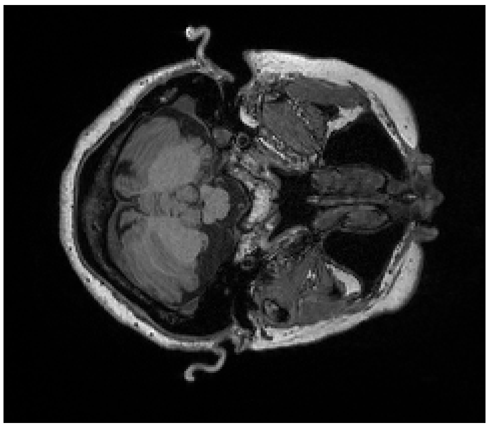
that slices / models that yielded more accurate initial validation readings will be weighted higher than models with worse validation readings.

3.4 Additional Specifications

The following section covers concepts not necessarily tied to each of the individual models. This includes the hyperparameters that were kept the same across all models, as well as the gradient-activation module which was used on several of the models.

3.4.1 Hyperparameters

Hyperparameters are external variables that are not directly associated with the layer design one has used to construct a model. As such, these parameters could be kept identical across all three models, in order to ensure that all differences in performance can be linked to the architecture alone. As each model had the same classification goal, it meant that the loss metrics could be kept the same across all tests. Below is a list of these hyperparameters, and their set values for all models:


(a) Slice 88.

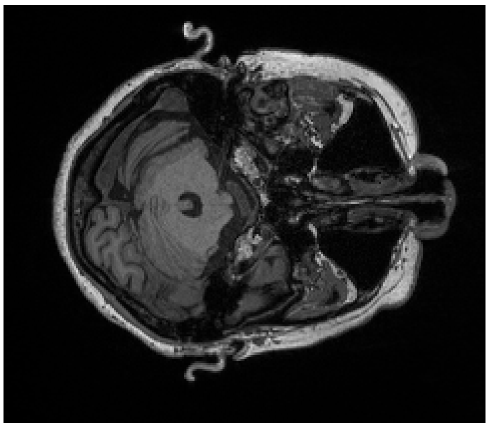

(b) Slice 96.

Figure 3.8: Example images of slices that on average produced higher validation accuracy and were thus selected as priority slices for the associated 2D model.

- Batch size: 3. The batch size was only set to 3, as any higher values resulted in memory issues within the code.
- Max epochs: 25. While the epoch upper limit was set to 25, an early-stopping algorithm was set up when training the models. This algorithm would prematurely end any training run that went 5 epochs without improvement in validation loss.
- Learning rate: 0.0001. This was set lower than the standard rate of 0.01 in order to further minimise the chances of the model over-fitting.
- Loss metric: Cross-entropy loss. Cross-entropy was chosen as the loss function due to its common application in classification tasks. Additionally, as the labels are one-hot encoded, categorical cross-entropy loss was chosen.
- Optimizer model: Adam. Adam is the most commonly used optimizer in most CNN designs, and has been proven to adapt well to almost any design.
- Early-stopping: An algorithm which would prematurely stop the training if it went 10 epochs without an increase in validation accuracy.
- Checkpointing: An algorithm that checks the validation accuracy at the end of each epoch, and after the training completes, it attempts to roll the model weights back to the epoch with the highest score.

3.4.2 Activation Gradient Visualizaton

The work sought to implement a means of visualizing the loss gradient between a given prediction's activations and the target class. Specifically, this dissertation employed an approach inspired by Grad-CAM [18], in which the gradients of the final convolution layer of a model are extracted and transformed into a heatmap that can be superimposed over the original image. By visualizing the values produced from predicting on an Alzheimer's-positive image, the resulting image highlights regions of the scan that yield the greatest positive-case activations. These highlighted regions can be used to infer which parts of a brain scan offers the most valuable data for detecting positive cases, allowing a conclusion to made on whether an accurate prediction requires the entire brain scan, or only a portion of it. Because negative cases are treated as the default zero-state, a map for negative activations was not investigated.

Additionally, visualising the convolution activations allows for comparisons to be drawn between the 3D model and one of the 2D models. In this case, the slice-based model was chosen. Each model produces gradient maps of equivalent dimensions. Therefore, 2D maps were generated for slices 50 through 100, and then combined to create a reconstruction of a 3D map.

Chapter 4: Implementation

The strategies presented were designed to be open-ended and reproducible using any number of programming languages and packages. However, this work was done in the Python language, using the TensorFlow package for model construction and Numpy for data processing. Other packages of note will be referenced when relevant. Additionally, the work utilised a high-speed computing server possessing 56 CPU cores and 4 high-performance GPU cards for the purposes of training and testing. This was done with the goal of greatly reducing training time, though it could have been reproduced (at a slower rate) on any machine with sufficient processing power (either CPU or GPU).

4.1 Pre-processing

In this section, it is explained how the data was set up and processed before model training / evaluation. The first part of this is data preparation, which covers how the data was prepared for the training pipeline. This includes how directories were set up, pre-runtime processes like skull stripping, and augmentation. The second part is the formatting of data — how each image was formatted directly before being input into the model. This involved resizing and normalization of the images.

4.1.1 Data Preparation

Pre-processing steps were set up according to the design specifications. The following is a list of noteworthy steps taken:

- The locational strings and labels were stored in Numpy arrays, then bound together in a Tensorflow data loader object.
- Skull stripping, when used, was done using the Python implementation of the ROBEX package [9], known as Pyrobex. A separate script was used to generate a stripped version of every image in a given directory, and these versions were saved to disk. This way, the process would not need to be repeated every time.

- A script was implemented to generate the list of file locations for each desired image, and then save them alongside their corresponding label. This allowed for different text files to store different dataset configurations. For example, one text file may contain the information for all AD and CN images, while another may have all MCI and CN skull stripped images.
- The data loader for any given subset was assigned a mapped function that could convert the loaded location and label into a Numpy pixel array with an associated encoded label. The image is loaded in using the Nibabel library [3].
- Data augmentation was also applied as part of the mapped function. For the 2D slices, this augmentation was done using the inbuilt Numpy augmentation functions. 3D augmentation was done using the Volumentations package [21], whereas 2D augmentation (for the 2D slices) used the Imgaug package [10]. The augmentation transformations are those outlined in the design chapter (rotation and elastic deformation) and were applied to all training samples.

4.1.2 Data Formatting

This subsection relates to the data resizing and intensity rescaling applied to each image. For the resizing, a centering algorithm was needed. In other words, the images need to be cropped around the edges of every axis, rather than from a single side. For the 3D model, the algorithm crops around each volumetric side of the image until it is an array of shape $169 \times 208 \times 179$, then adds another axis to the end of the array to make the object compatible with batching. This is the same for the 2D subject-level model. For the slice-based one, once the image has been resized, an individual slice is extracted based on the associated model.

The main goal of the rescaling process was to rescale all the pixel values to a common scale. Most often this is set as 0 to 1, as a smaller normalised range has been shown to improve model training performance. However, multiple approaches exist for implementing this, and the literature did not present a clear best option. Many papers had placed great importance on the rescaling algorithm [23], therefore a decision was made to explore four promising implementation strategies. The four algorithms are as follows:

1. Normalization with trimming: Takes a maximum pixel threshold value as input, then rescales all pixel values to a range between 0 and 1. Pixels above the threshold

are trimmed down. For the images in this work, the minimum pixel value was always 0.

2. Normalization without trimming: Same as above, only without trimming down values greater than the threshold. This produces some values greater than 1 in the rescaled range.

3. Local normalization per image: This approach eschews using a chosen threshold value for the entire set. Instead, each individual image is rescaled using that given image's minimum and maximum pixel values to set the bounds for the new scale, from 0 to 1.

4. Standardizing: Using a "standardize" algorithm, where pixel values are instead rescaled using a formula that uses both the mean and standard deviation of the set to generate a standard range.

Each approach requires several additional values to be known. These were the average value across all sets, the values of the 2nd and 98th percentile (representing non-outlier min and max values), as well as a maximum cut-off value that would support the greatest share of the data. Those values, obtained during experimentation, are shown in the next chapter. For now, the function formulae are shown below:

Normalize: $y = \frac{x}{m}$, where x is either the trimmed or untrimmed pixel value, and m is either the input maximum threshold value, or the local maximum in the case of the 3rd approach. As the minimum is assumed to be 0, it does not factor into this formula.

Standardize: $y = \frac{x-\mu}{\sigma}$, where x is the pixel value, μ is the global mean of the dataset, and σ is the standard deviation of the set.

During experimentation, the functions were then compared in order to determine the best option for these models. The approach of local normalization per image was found to yield the most reliable results, and was subsequently utilised as the standard for the model. This will be elaborated on within the results chapter.

4.2 Model Training

Here, the dissertation briefly covers the data partitioning for model training, as well as the actual implementation of the models. As these topics were covered in detail in the prior chapter, this section only covers the technicality of the partitions, as well as the additional steps that had to be taken for the 2D slice-based model ensemble's implementation.

4.2.1 Data Partitioning

Data splits were done using the SKLearn stratified-split function, to ensure subsets had a proper distribution. Additionally, all splits utilised seeded shuffling. While a random arrangement of data was desired, a seed allowed for that same distribution to be kept consistent across all experiments.

Previously, it was shown that the AD class has a significantly lower count than the other two classes — approximately 4 times smaller than the next class. Therefore, in order to avoid performance loss due to imbalanced datasets, training attempts using the AD class would use a trimming function. This decision was motivated by the fact that many of the papers reviewed utilised subsets of the ADNI set for training. While this does significantly reduce the amount of data available, tests using unequal class sizes resulted in the model quickly over-generalising to predict only a single class. The trimming function implemented trims the larger class by taking a seeded random subset of elements equal in size to the smaller set. This minimises class imbalance with a random, unbiased selection, however the seed ensures that the same distribution would be used for every test, so as to not introduce variance to the tests.

4.2.2 Model Implementation

Each model was set up using the Keras package for Python. The process was the same for each model, except for the chosen layer architecture. The layer specifications can be found in the previous design section.

For the 2D slice-based design, each model used the same approach, but additional steps were needed. The implementation of fitting models for each unique slice utilises a loop to instantiate and train a model, where each new loop changes a global variable for the slice index. The variable controls which 2D slice of the input volume is used for each input sample during training. The data loader is configured to fetch slices based on this variable. For ensemble voting, there are many packages available in Python for automating ensemble learning, however they are often limited to scenarios where the dataset remains constant. This experiment requires each model to view a different slice level of the dataset, and thus it proved easier to design the voting ensemble from first principles. Using the test set of data, predictions are generated by splitting a given test image into slices, feeding each relevant slice into the corresponding model, and having the model produce a prediction. Each set of predictions is then appended to a single 3D array, where each prediction is a set of probabilities for the various classes. The soft voting algorithm then takes the weighted average of this set, using recorded validation accuracies as the weighting variable. The hard voting algorithm translates the probabilities into a final class prediction for every slice, then finds the model class.

4.3 Evaluation Step

The evaluation of the models, though not mentioned until now, is a vital step to be taken. The way in which the models are evaluated, and how these results are stored, will form the basis of the next chapter. This section will highlight by what metrics the models were evaluated, and how testing was run in order to obtain a diverse set of results. Additionally, this section will cover how the gradient visualisation was conducted in practice, and how its outputs were recorded.

4.3.1 Recording Results

Models were assessed across three avenues: the training metrics, the evaluated scores, and the performance average across repeated testing.

The training metrics refer to the training/validation scores available directly from the fitting step. While they do not reflect how the model performs in the presence of unseen data, this data permits analysis of the actual training process. Possible observations include whether the model is under or over-fitting, and how different permutations in some variables might affect the training rates. The metrics assessed here, applying to both training and validation, are:

- Loss: Categorical cross-entropy. The same loss as was used for the training function. While accuracy serves as an easier measure of effectiveness at a glance, the tested loss of the model is still important as well. It conveys how much the label prediction output deviate from the true values.

- Accuracy: Binary accuracy. This was chosen as the accuracy metric, as all the models only predict on two classes. While loss is based on the measurable different between the actual and expected outputs, accuracy reflects how often the model will be able to make the correct diagnosis. As humans more naturally understand and contextualise accuracy readings, this will be used as the main means of comparing the performance of different model configurations.

Actual testing was conducted using evaluated scores. These involved running an evaluation function on the trained model, which simply returns the loss and accuracy scores obtained using the test set. As this is based on a held-out set of unseen data, it is the most valuable data and is the primary objective of the evaluation step.

In order to properly validate a model's average performance and variance, a method of repeated testing and data collection was needed. A k-fold training approach was utilised for this, and acts the primary source of data for the work going forward. The k-fold algorithm operates as follows:

1. The model is generated, and its initial weights are saved so that they can be accessed later.
2. The training and validation sets are automatically partitioned based on the value of k, in order to ensure a different distribution on each new training run. This is handled by the SKLearn k-fold module, which stratifies and splits the data based on the value of k.
3. The model is then trained on the partitions as usual, and a snapshot of the model is saved for future reference.
4. Standard testing and evaluation is conducted on the model. These values are all stored in memory, and the final loss and accuracy values for the fold are appended to arrays.
5. This process then repeats, up to a total of k times. In our case, k was always set to 5. Lower than 5 would be counter-intuitive to the idea of getting a wider array of tests. Higher than 5 causes the validation partition to become too small, as the ratios are dictated by the value of k.
6. Once all folds are complete, the averages and the standard deviation of the model's loss and accuracy are computed and returned.

With averages and standard deviations in hand, the data produced by this procedure can then be used to generate box-plots for easy comparison between multiple models. Box-plots offer a more visual means of assessing the variance in each model, compared to its overall efficiency.

4.3.2 Activation Gradient Visualisation

As specified in the Design chapter, a GradCAM-type algorithm was implemented in order to map out and visualise the activations and gradients of the 3D model and the 2D slice-based model. The vast majority of information regarding GradCAM implementations is limited to 2D images. Inspiration was taken from the MedCam module [8][17][4], which outlines specific extraction and visualisation of gradients relating to volumetric medical images. Unfortunately, the module only works with PyTorch models, but it still offered valuable insight.

The actual code implementation was conducted using a modified version of the Keract module [16], which is designed to extract activations from a given layer in a model. This was used to extract the activations from the convolution layer in each model (which also includes the ReLU activation function). The Keract module combines the channels of a given map together and superimposes it across the original image as a heatmap. The activations are then saved as either a NIFTI file (3D) or a PNG image (2D). For the gradients, a custom script was instituted to produce a gradient array using the Tensorflow GradientTape function, which records gradients in the desired layer during a prediction call. The channels of the maps were summed based on a weighted mean function across all channels. Using this algorithm, the 3D model produced a 3D array, which is saved as a NIFTI file. For the 2D model, a setup was desired in which multiple concurrent 2D gradient maps could be combined into a 3D volume, for the purpose of comparing to the 3D model map. Therefore, a slightly altered version of the slice-based model was used here. Instead of training on only the priority slices, the models were instead trained over the range of slices 50 to 100. These maps could then be combined into a coherent volumetric subsection of the brain, and saved as a NIFTI.

In all of the above cases, the visuals (NIFTI/PNG) were produced for an example CN image, as well as an AD one. The results of this, as well as the discussion surrounding it, are documented later in the dissertation.

Chapter 5: Results

All the experimental results of this work are presented within this chapter, alongside observations on the outcomes. The first section deals with the three models and their evaluated scores. Brief discussion is made on the comparisons between the performance of each model. The training times of each model are also noted, as this contributes to the discussion on model efficiency. The following section covers the evaluation of the impact that several training parameters have on model performance. Finally, the outputs of the gradient activation visualisation are presented in the final section.

5.1 Model Assessment

Evaluation of models used for the classification problem can be divided into the following sub-categories: the 3D subject-based CNN model, the 2D subject-based CNN model, and the 2D slice-based collection of CNN models. The following observations deal with overall performance of the model architecture itself, focusing on the readings given by each of the full systems. The individual effects of variables such as pre-processing are to be observed in a later section.

5.1.1 3D CNN Classification

The 3D CNN model, using the final design presented in the design section, on a single training run yielded the results shown in table 5.1. Each training run ran for up to a maximum of 25 epochs, not counting premature ends due to early stopping. Results are split for both classification modes. The training accuracy has been included to emphasize that while a model may have little trouble learning the training set perfectly, it is not necessarily able to perform equally well during validation and testing. While each fold's training data will be large enough to allow to the model to fit onto the available data, the validation set verifies whether the model can adapt to a smaller, varying batch of data. Furthermore, the held-out testing set serves as the most reliable assessment of whether the model can perform well on entirely unseen data.

Table 5.1: Evaluation metrics for the 3D model for both classification cases, showing the final tested accuracy and the associated training and validation accuracies. The evaluation was conducted using a standard accuracy metric and a categorical cross-entropy loss metric.

Classification	**Accuracy**	**Loss**	**Validation Acc**	**Training Acc**
CN versus AD	84	0.97	85	98
CN versus MCI	65	1.13	66	91

Some training curves (accuracy and loss) for the CN versus AD run are shown in figs. 5.1 and 5.2. These highlight a sample case of the progression of the training and validation metrics over the course of multiple epochs. As the figures show, training performance shows quick improvement until it reaches nearly 100% accuracy. However, the validation metrics show a slightly worse, more realistic pattern based on unseen data. Though the model validation loss fluctuates over the first few epochs, it quickly corrects, and both metrics improve over the course of the remaining epochs. That both the training and validation curves follow the same general shape shows that the model is not over-fitting to the training data. Curiously, the curves show that the model could have continued to improve with more epochs, though the rate of improvement is already beginning to fall off by epoch 25. It is also worth noting that despite the positive trend in the validation accuracy curve, there are still some epochs where it briefly drops. This is likely due to the model over-correcting for some data elements, and reinforces the need to track the checkpoints of each epoch so that the model can be rolled back if the final epoch weights are not desirable.

Following this, for the CN versus AD case, the model was assessed more extensively via a *k*-fold validation strategy in order to validate the above results. Identical versions of the model were trained across 5 folds, where each fold used a unique set of shuffled and stratified training/validation data splits. Each fold was evaluated using the same held-out test set. These results were then compiled into box-plots better suited to summarising the performance. Based on the box plots, the mean scores act as the primary point of comparison, while the interquartile range (IQR) of the box acts as an estimate of the model's expected variance. The results of these experiments can be seen below in figs. 5.3 and 5.4. While the prior tables show the difference in scores between the two classification cases, these box-plots highlight the extent of variance in the model.

Observing these results, we see a mean accuracy of 81.3%, an accuracy IQR of around 4%, and a loss IQR of around 0.13. On its own, this shows the model's variance is not a cause for concern, as the earlier quoted accuracy and loss both fall near the mean value of both box plots. The loss variance is larger than that of the accuracy, though

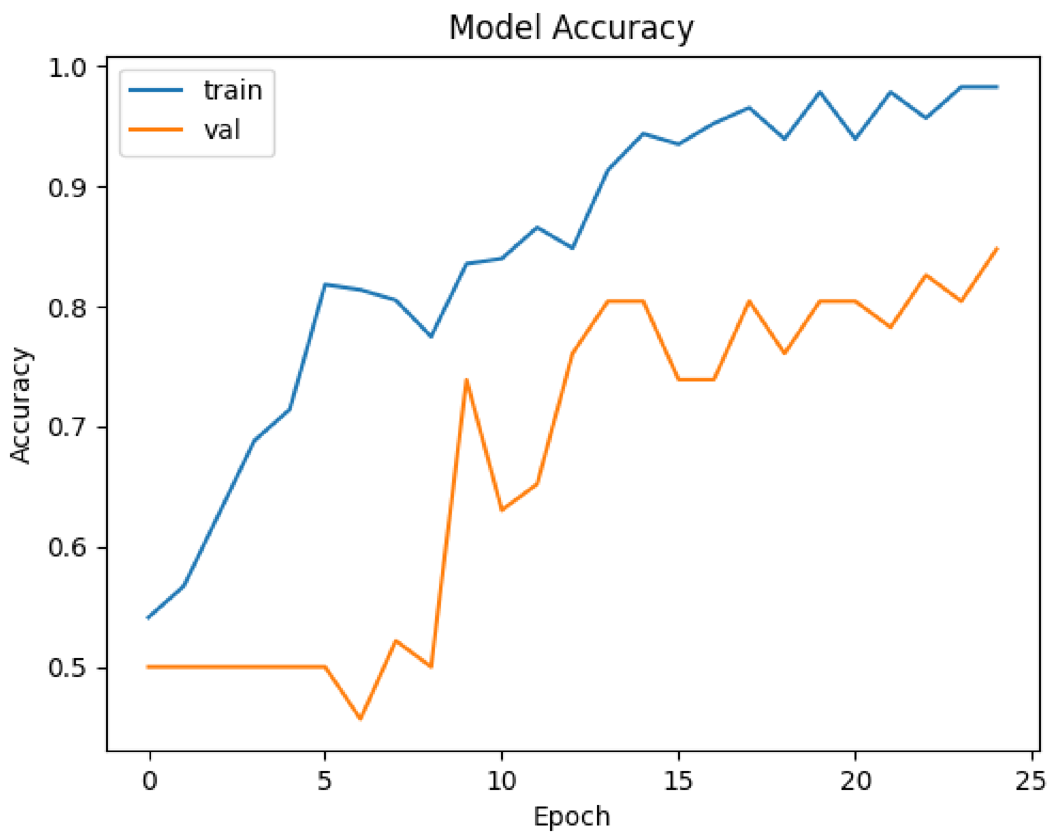


Figure 5.1: Plot showing the training and validation accuracy curves of the 3D CNN model, trained over 25 epochs, with respect to classification of CN versus AD subjects.

this is unsurprising given that the checkpointing system optimises for weights with the best accuracy above all else. It also means that while the model exhibits accurate final predictions, it is perhaps more sensitive to outlier data. This outlier data could result in exaggerated loss penalties, despite the model performing well otherwise. Furthermore, while there is variance in the final accuracy of the model, a quoted accuracy of 81.3 ± 4% is still a very well-performing classifier. Additionally, the earlier quoted score of 84% accuracy is confirmed to fall within this range. Even the lowest outlier, at 74%, falls within 10% of the mean and would be an acceptable target.

The existing variance likely comes from variations in data partitions. While every run of the model possesses identical starting weights, hyperparameters, and testing samples, each variation of the training possesses slightly different training/validation sets. This is due to the inherent shuffling of the training/validation data partitions, which influence the optimisation of the weights differently each time. While for the most part the variation is slight (<4%), there are still outliers, such as the run at 74%. To explain this, it could have been the result of suboptimal image samples within the training data that, if fetched

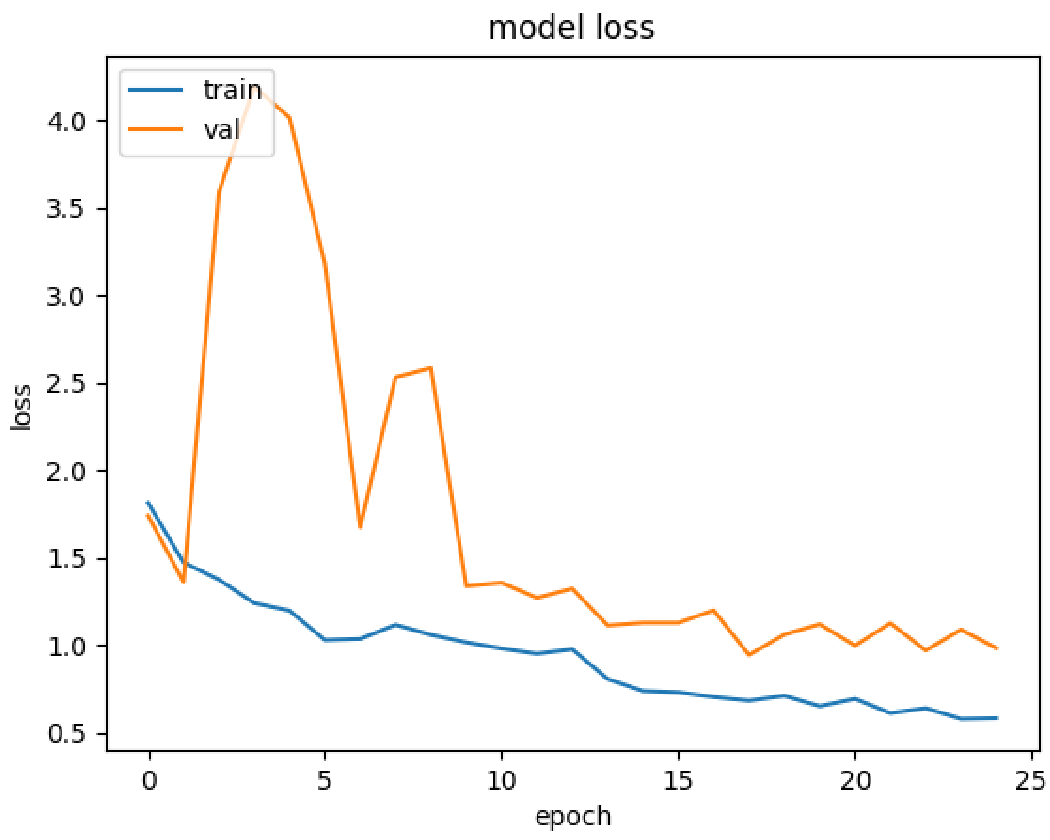


Figure 5.2: Plot showing the training and validation loss curves of the 3D CNN model, trained over 25 epochs, with respect to classification of CN versus AD subjects.

early during training, could negatively skew the training process. When presented with difficult-to-classify images in the testing data, these small negative skews would result in some models failing to correctly classify an image that other variations succeeded with. In an attempt to investigate this, a tally was kept of test images that a model would fail to correctly classify. These difficult images were then sorted and ranked. A sample that was consistently misclassified in all five folds is presented in fig 5.5.

Fig 5.6 shows a comparison between this sample and an example "easy" sample, which was used as a data example earlier in the dissertation. Unfortunately, the comparison does not reveal any obvious differences (at least to one not professionally trained to assess MRIs), besides some greater pixel intensity ranges in the difficult image.

Comparing the 3D model's performance to other similar works, particularly those summarised by Bratic et al. [2], places these results firmly in the middle of the rankings. The two closest comparisons are by Ewers et al. [6], who got 86% accuracy for CN versus AD and 62% for CN versus MCI, and Chu et al. [5], who got 85% and 65%, respectively.

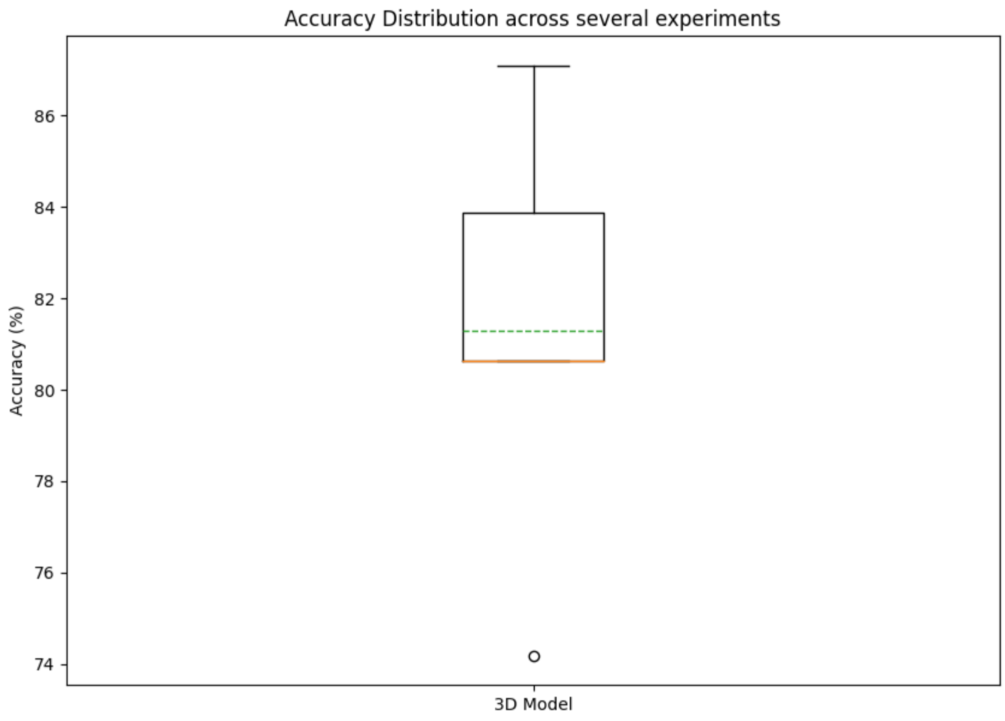


Figure 5.3: Box-plot showing the accuracy summary of the final 3D model for the case of the CN versus AD classification task. The orange line marks the median, whereas the green line marks the mean.

These are both very close to our model's single-run scores of 84 and 65%, and shows that it is common for models to struggle significantly more with the MCI case than the AD case. The performance of the 3D model was thus deemed satisfactory, especially when considering that it is only one of three approaches being discussed in this work. Of course, there is still some room for further improvement if desired, as evidenced by works such that of Wen et al. discussed earlier.

5.1.2 2D Subject-level CNN Classification

Table 5.2 shows the evaluated performance of the 2D model used to fit the data at a subject-level. Similarly to the 3D model, this was also trained over 25 epochs. Comparing the two classification cases, there is a smaller differential compared to the 3D model's results. The loss scores are fairly close though, which suggests that while the model struggles to reach the MCI prediction, the error present in the individual class confidence scores is better than the accuracy score suggests.

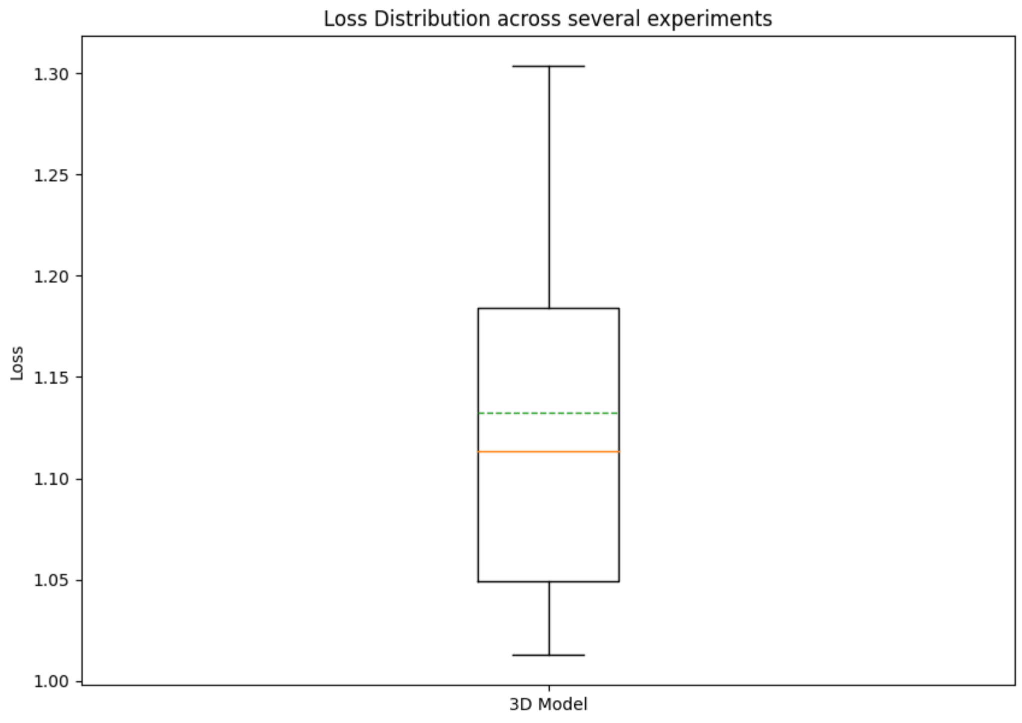
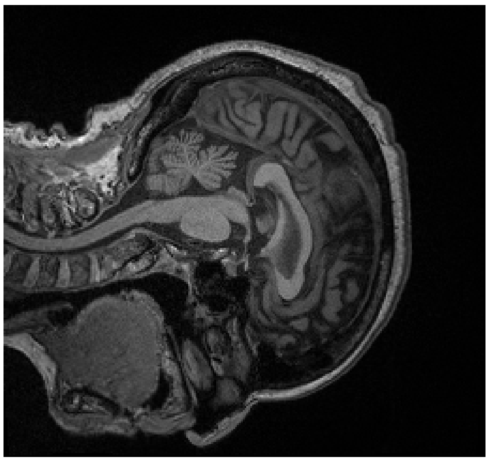


Figure 5.4: Box-plot showing the loss summary of the final 3D model for the case of the CN versus AD classification task. The orange line marks the median, whereas the green line marks the mean.

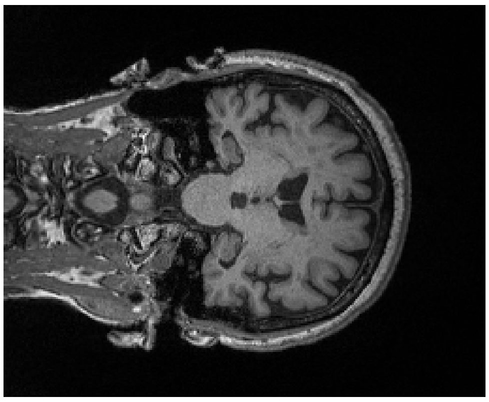
Similarly to the 3D approach, the model was also subjected to repeated k-fold training runs, which serves as the main way of comparing the two models. The comparisons of accuracy and loss, for the CN versus AD case, are shown in figs 5.7 and 5.8. Continuing forward, results used to compare models will be exclusive based on the CN versus AD case. This is for two reasons. First because the dataset for this case is smaller and allows for faster experimentation. Second, this case universally yields the highest performance for each model, allowing comparisons to be made in a best-case scenario. One can observe by comparing these box-plots that the 2D model has a marginally lower mean accuracy (2% lower), but better loss compared to the 3D implementation. However, it possesses a significantly larger IQR than the 3D model.

Table 5.2: Evaluation metrics for the 2D model for both classification modes. Similarly to the 3D model, evaluation was conducted using a standard accuracy metric, and a categorical cross-entropy loss metric.

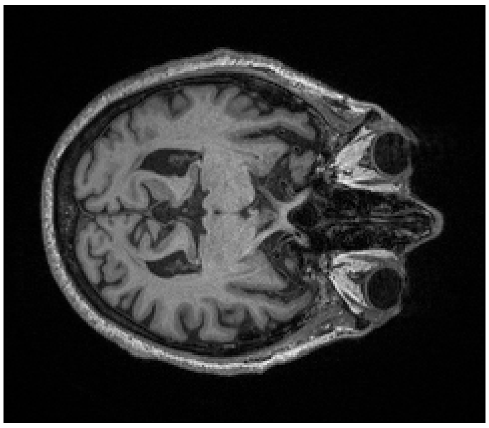
Classification	**Accuracy**	**Loss**	**Validation Acc**	**Training Acc**
CN versus AD	74	0.72	74	90
CN versus MCI	66	0.74	58	74



(a) Sagittal plane.



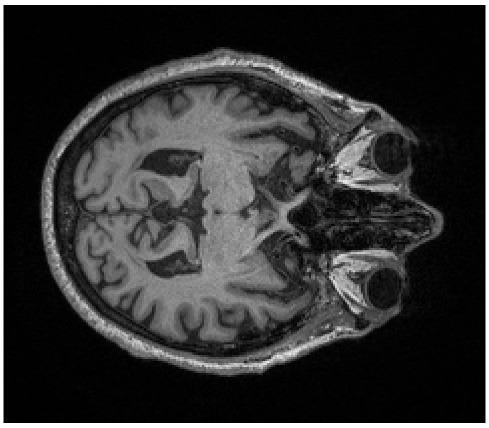
(b) Coronal plane.



(c) Axial plane.

Figure 5.5: Multi-plot showing the three plane views of an AD sample that proved extremely difficult for the model to predict correctly.

While the 2D model's IQR does extend above the 3D model's, there is too much variance in the values and thus the 3D model is considered to be performing better. This suggests that performing the depth-wise convolutions in isolation produces less reliable results, due to the loss in some level of feature data. Aside from the one outlier point, the 3D model yields more consistent and better performance. On the other hand, the loss comparison shows the 3D model to perform noticeably worse. As the 2D model does not yield better accuracy despite this large difference in loss, it can be assumed that the 2D model is less sensitive to outlier data, and thus incurs a lower overall loss, though its overall prediction accuracy is still lower than that of the 3D model. As we are evaluating based on accuracy, the optimised 3D model is deemed to perform better. However, the fact that the 2D model performs nearly as well, using only a 2D convolution window, shows that it is still a valid alternate strategy.


(a) Difficult image (axial plane).

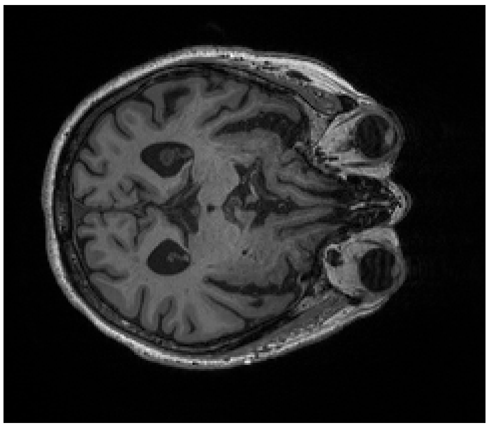

(b) Easy image (axial plane).

Figure 5.6: Paired figures contrasting an image that proved difficult for the model to classify, versus one with which it had no difficulties.

5.1.3 2D Slice-level CNN Classification

The 2D slice-level implementation, using the model ensemble trained on a set of high-priority slices, was tested in the same way as the previous two architectures. Firstly, table 5.3 shows an example of the individual scores for each slice when trained on CN versus AD. This helps show the validation accuracies, which were used as weightings for each slice in the soft ensemble voter, as well as the accuracy of the individual predictions that are later combined in the ensemble voter. These results showcase how some slices will yield more accurate predictions, such as slice 64, and thus should be given more weight in the final selection. Also note that, on their own, each model exhibits significant validation loss.

Table 5.3: Validation accuracy and loss per slice individual slice model trained for the 2D slice-level design.

Slice Model No.	**Val Accuracy (%)**	**Val Loss**	**Accuracy (%)**
56	75	1.45	75
57	56	1.81	75
58	62	1.81	63
64	66	1.58	88
75	59	1.75	57
85	69	1.52	69
88	72	1.86	69
89	69	1.51	75
96	73	1.45	56

Table 5.4 shows the ensemble evaluated scores, this time for both classification cases, after running the slices through either hard or soft voting. As the voting algorithm only

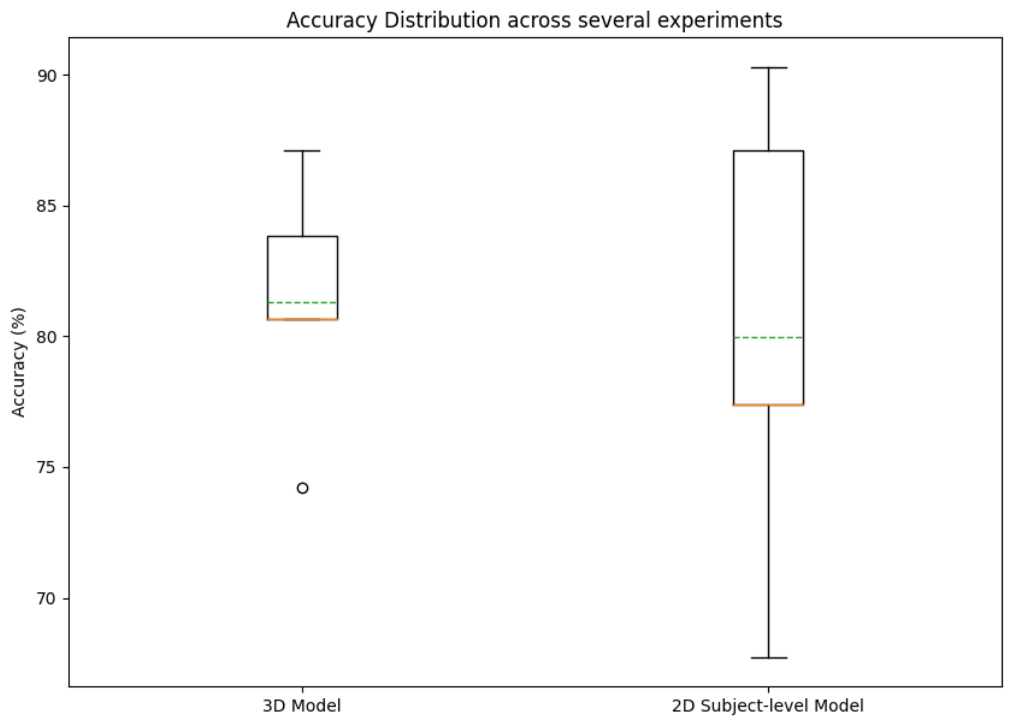


Figure 5.7: Box-plots comparing the accuracy performance of the 3D model to those of the 2D subject-level model, in the case of a CN versus AD classification problem. The orange line marks the median, whereas the green line marks the mean.

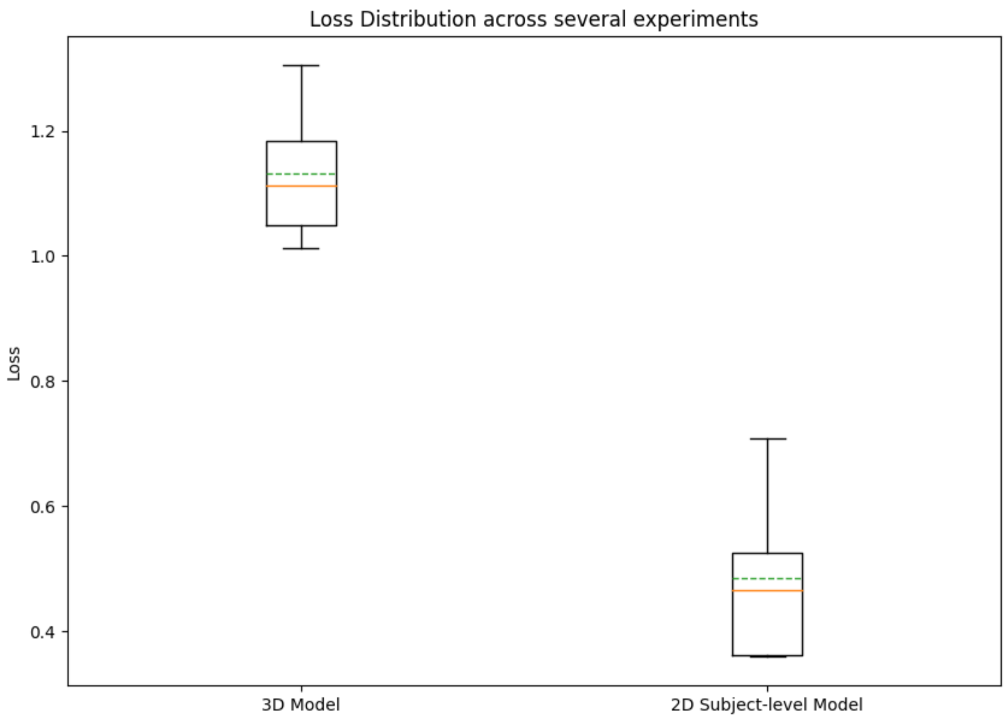


Figure 5.8: Box-plots comparing the loss performance of the 3D model to those of the 2D subject-level model, in the case of a CN versus AD classification problem. The orange line marks the median, whereas the green line marks the mean.

Table 5.4: Ensemble voted prediction accuracy for the slice-level 2D model for both classification cases.

Classification	**Voting Method**	**Evaluated Accuracy (%)**
CN versus AD	Hard	81
CN versus AD	Soft	81
CN versus MCI	Hard	62
CN versus MCI	Soft	69

aggregates the accuracy scores, loss is not covered here. Once again we can see that the model performs significantly better with the AD data compared to MCI. Furthermore, while both voting methods yield the same accuracy for the AD case, soft voting performs noticeably better for the MCI case. In general, soft voting was found to consistently perform either equal to or better than hard voting. Therefore soft voting, or some manner of weighting the individual slice predictions differently, is recommended for this kind of design. Furthermore, the accuracy using the ensemble prediction is noted to be significantly higher than the average of the individual slice accuracies. In the CN versus AD case, for example, the average of the individual accuracies would be 70%, whereas the accuracy of the final combined design reached 81%.

In figs 5.9 and 5.10, the slice-based model ensemble is compared to the 3D model, as well as the previous 2D model. This model's accuracy is found to be even closer to the 3D model (only 1% difference), though the IQR falls lower. It exhibits similar variance to the 3D model — again less than the first 2D model. For the same reasons that the 3D model was rated higher than the 2D subject-level model, this slice-based design is rated higher as well. However, while the mean loss is similar to the 3D model, there is practically zero variance in the loss. This likely comes from the small selection of slices, which may not include the outlier data that produces spikes in loss. Similarly to before, while the 3D still performed slightly better, this 2D model is practically just as good for making predictions. Due to the potential for optimisation in the design (in terms of slice selection / voting algorithm), as well as its novelty, this slice-based model is regarded as more valuable than the previous 2D design. The model's ability to produce promising results while using only a fraction of the data gives it a notable advantage over the 3D model.

5.1.4 Training Time Cost

In the interest of noting the cost in terms of time that each model carries (with the slice-based 2D model showing time taken for priority versus full training), table 5.5 below

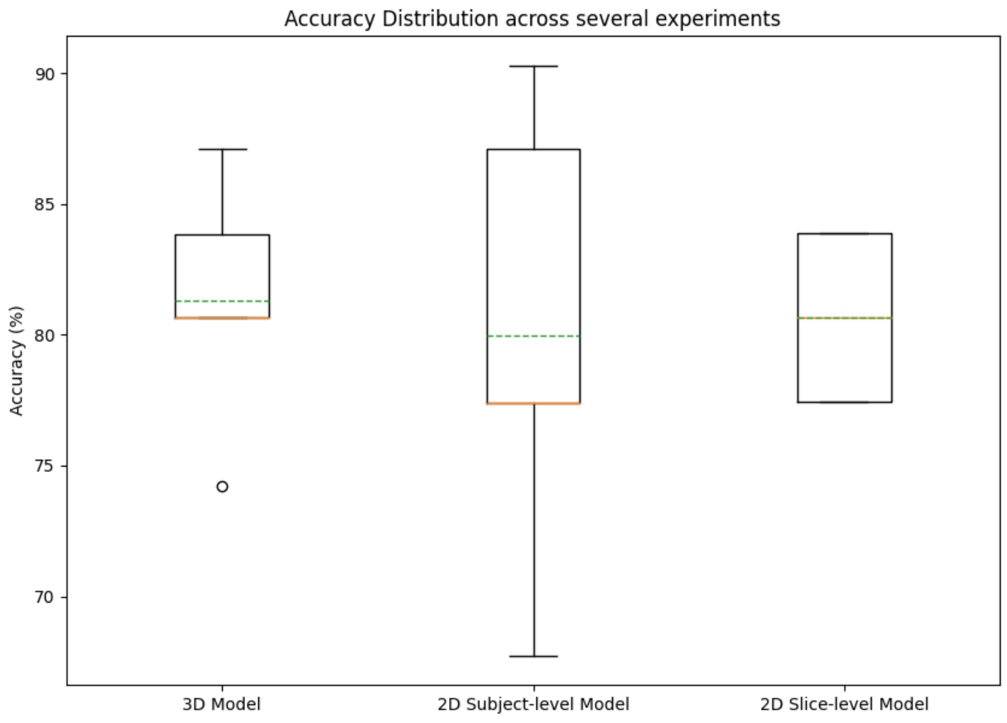


Figure 5.9: Box-plots comparing the accuracy of all three models evaluated on the *k*-fold experiment in the case of a CN versus AD classification problem.

Table 5.5: Time taken to train each of the models, in an identical training setup for classifying CN versus AD.

Model	**Training (hh:mm:ss)**	**Epochs**	**Time/epoch**	**Total %**
3D	5:02:23	25	0:12:05	99
2D Subject-level	5:17:28	25	0:12:42	98
2D Slice-level	8:04:48	23.4 (average)	0:20:46	99

shows the total training time for each of the models. Furthermore, in order to account for differing numbers of epochs trained, the approximate time per epoch is also presented. For the 2D slice model, this was an averaged from the epochs taken by each model. Admittedly, this does not account for overhead time costs, but they are considered small enough to not factor in. Also included are the percentages of the total runtime that training takes up, with the near-100% rates showing that training time is the only real time metric to consider. The experiment was put through additional repeated runs, under the same parameters, to ensure that each run's times did not deviate (+- 5%) from the mean. This was done to verify that there was minimal variance within the presented results.

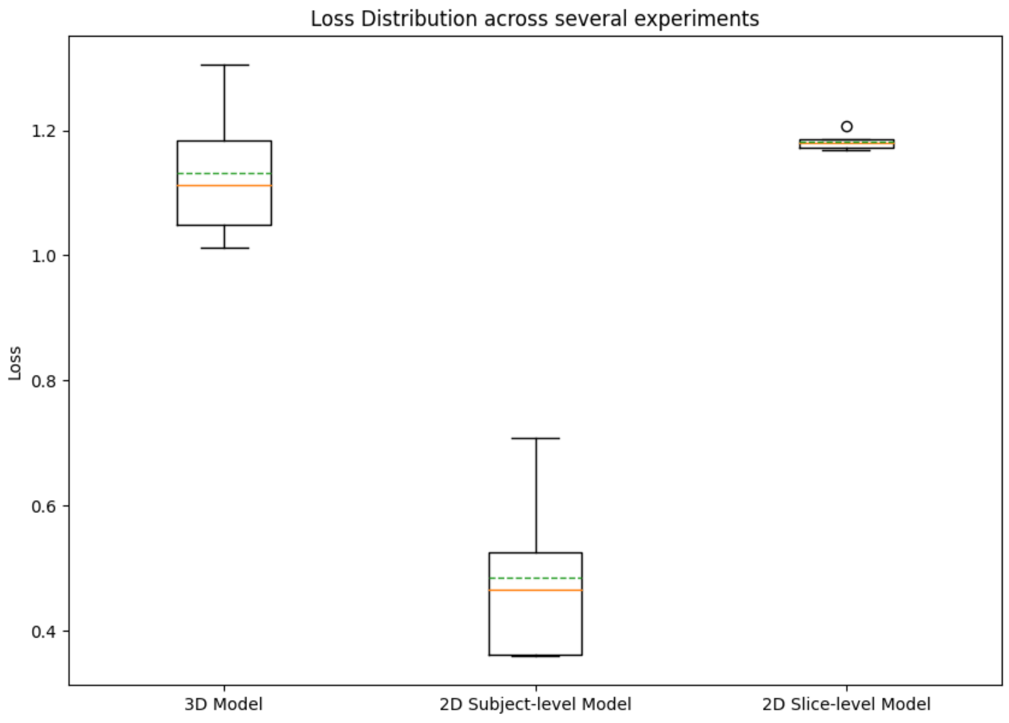


Figure 5.10: Box-plots comparing the loss of all three models evaluated on the k-fold experiment in the case of a CN versus AD classification problem.

The 3D model has the shortest training time of the three. This is not unexpected for the slice-based ensemble, which has the overhead of having to train multiple models, and thus takes nearly twice as long as the other two. The 2D subject-level model, on the other hand, holds much similarity to the 3D model and thus the difference is very slight. A difference of less than a minute per epoch could have several causes, such as delays in the convolution layers, or costs within the module code where one type of layer is simply more efficient than the other. Regardless, the fact that the 3D model trains the fastest while also yielding the best results is another point in its favour.

5.2 Training Parameters

Several variable parameters were present during the training process. While the previously presented results were those of the final selections of each model design, the following subsection highlights the many variable parameters and their effects on the overall performance of the model. This examination shows results only for the 3D model, as it is the primary interest of this work, and any trends observed on the 3D model were largely the same on the other two models.

The parameters to be split into pre-processing steps, applied to the training / testing images, and over-fitting correction, applied to the model.

5.2.1 Pre-processing Steps

Pre-processing experimentation is split into three groups of variables: intensity rescaling, skull stripping, and augmentation. While continuous experimentation was used to find the optimal choices for each, and thus set the parameters used for all future experimentation, the following are a series of tests conducted to observe the isolated effects of each variation in these groups. For each variable, the 3D model, trained for CN versus AD, is treated as the base point of comparison.

The first experiment concerned the intensity rescaling function. The readings in table 5.6 show the evaluated scores of otherwise identical models (using the final 3D model design) utilising different rescaling functions. The noticeable drop when not using a rescaling function is expected, as the literature shows that intensity rescaling has a large impact on CNN training performance. Furthermore, the different functions used have a small but noticeable effect on model accuracy, with trimmed global, as well as localised normalization showing the best results. On repeat tests, the trimmed global normalization showed some variance in performance, and therefore the localised version was used in all future experiments. Further investigation showed a high degree of variance in pixel values across all the images in the dataset. While the average pixel value across the entire set is 1054, 158 images exceed a value of 200 (double the average), with the maximum being 1200. Using localised minimum/maximum values therefore makes sense as it prevents outlier images from skewing the scale.

Table 5.6: Evaluated scores for models utilising one of four different intensity rescaling functions, each trained on the same data splits.

Function	**Accuracy**	**Loss**
Global normalization with trimming	84	0.92
Global normalization without trimming	81	1.04
Local normalization per image	84	0.97
Global standardizing	81	0.97
No rescaling	77	1.07

The second experiment deals with the impact skull stripping input images has on overall performance. A test was conducted using the final iteration of the model, training it both with and without skull stripping. The results of which can be seen in table 5.7. Skull stripping was actually found to cause a drop in performance. This is hypothesized to either be due to the rescaling function not interacting well with the new images, or due to the stripping process removing valuable feature data on the surface of the brain-tissue. In either case, the significant effort required to obtain skull-stripped images (generating masks for each image, followed by either needing to rewrite the entire set or save altered copies) outweighs the possible benefit of the process. This falls in line with what other literature has claimed on the topic, and thus it was not further investigated.

Table 5.7: Performance of the 3D model with and without skull stripping, in the context of a CN versus AD classification task.

Experimental Setup	**Accuracy (%)**	**Loss**
With skull stripping	74	1.14
Without skull stripping	84	0.97

The third experiment observes the effect of augmentation on the training process. The experimental process is the same as the one used in the previous experiment, with the variable now being the augmentation transformations applied during training. The two transformations were rotation and elastic deformation. As outlined by the design, rotation augmentation was performed on 60% of training images and elastic deformation on 30% of them. The results of the 3D model with both techniques, and without any augmentation, can be observed in table 5.8. Removing augmentation yielded an accuracy decrease of 7%. This was a large enough impact (falling outside the 3D model's IQR noted earlier) to warrant the continued usage of augmentation. In contrast to the skull stripping, augmentation is also a much less demanding option to implement, as it only requires adding relevant functions to the data-loaders. Therefore it is recommended to use augmentation when tackling training problems like this.

Table 5.8: Tested scores of the 3D subject-level model with and without augmentation.

Augmentation	**Accuracy (%)**	**Loss**
Augmentation	84	0.97
None	77	1.05

5.2.2 Over-fitting Correction

Unexpectedly, the concern of over-fitting ended up posing quite a significant challenge. Early iterations of the model quite easily over-fit onto the given dataset if no measures were taken to correct this behaviour. As a result, validation and tested scores would very quickly diverge from training scores after only a few epochs. On the other hand, experiments showed that applying too much over-fitting correction negatively affected model performance, though this could possibly be alleviated with a different hyperparameter set-up. The final version of the design found a good balance between the two, and produced favourable results. Still, an experiment was conducted in order to evaluate how the same design would fare with all regularization removed.

In table 5.9, the evaluated accuracy and loss of the model is compared with and without regularization. Without regularization, the accuracy is 80%. While this is a 4% difference from the base score, it still falls within the expected IQR shown in the k-fold test. At first glance, this could imply the drop is simply a result of variance. However, the full picture becomes clear when we also take the training curves into account. In figs. 5.11 and 5.12, the accuracy and loss curves for the no-regularization model — both training and validation — are shown.

Table 5.9: Final tested performance of the 3D model, with and without regularization layers, in the context of a CN versus AD classification.

Experimental Setup	**Accuracy (%)**	**Loss**
Regularization	84	0.97
No regularization	80	0.71

From these curves, the over-fitting of the model immediately becomes evident. Around epoch 10, both validation accuracy and loss begin to diverge from the training curve. Comparing this with the earlier validation curves, which improve with each epoch, shows that regularization creates a significantly more stable model with a higher performance ceiling. While the given experiment did yield a fairly high-accuracy result for the non-regularized model (owing to the checkpointing system rolling back to the best epoch), the curves show that such a result is not guaranteed, and the actual training process is extremely erratic. Furthermore, model performance quickly drops off, guaranteeing that such a design would not see any improvement with an increased number of training epochs, unlike the original design. Thus, the importance of utilising regularization for such a model is reconfirmed.

5.3 Model Activations

Several gradient maps were obtained for the purpose of comparing model learned feature patterns. The first images come from the 3D model, which acts as a baseline for the comparisons. The figures in 5.13 show the activation maps for a CN image as well as an

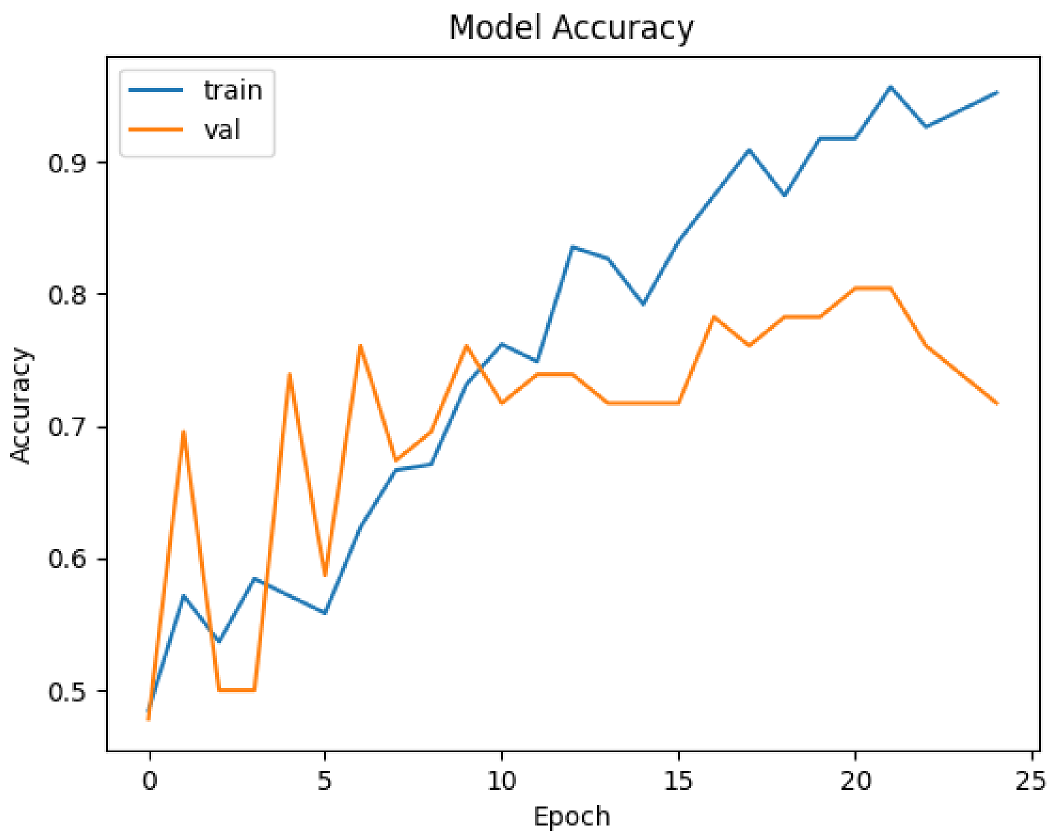


Figure 5.11: Accuracy curve for the no-regularization 3D model, trained for CN versus AD.

AD one (both of which were correctly classified by the model). The figures show the 3D volume, as well as midpoint slices from each axis. The corresponding gradients are shown in figure 5.14. Observing these images, it is difficult to find an obvious visual pattern in the regions with significant activation. Comparing the CN maps to the AD ones shows little difference, and yet the model was still able to correctly classify both images with high confidence. This suggests that the patterns in the feature data are inconspicuous, and there is no clear "region of interest" that the training could be refined to. A basic assessment shows though that the outer folds, as well as the central-most folds, appear to produce the most activations.

Next, we present the gradients generated by the 2D slice-based model ensemble, on an AD-class image. In order to observe their relationship with each other, a range of maps, corresponding to a range of slices, were produced for this 2D model. The maps were then combined into a 3D volume. This was done using the modified 2D design mentioned in the implementation chapter. The model was trained on slices 50-100, and the output is compared to a corresponding 3D model map of slices 50-100. Fig. 5.15 shows the

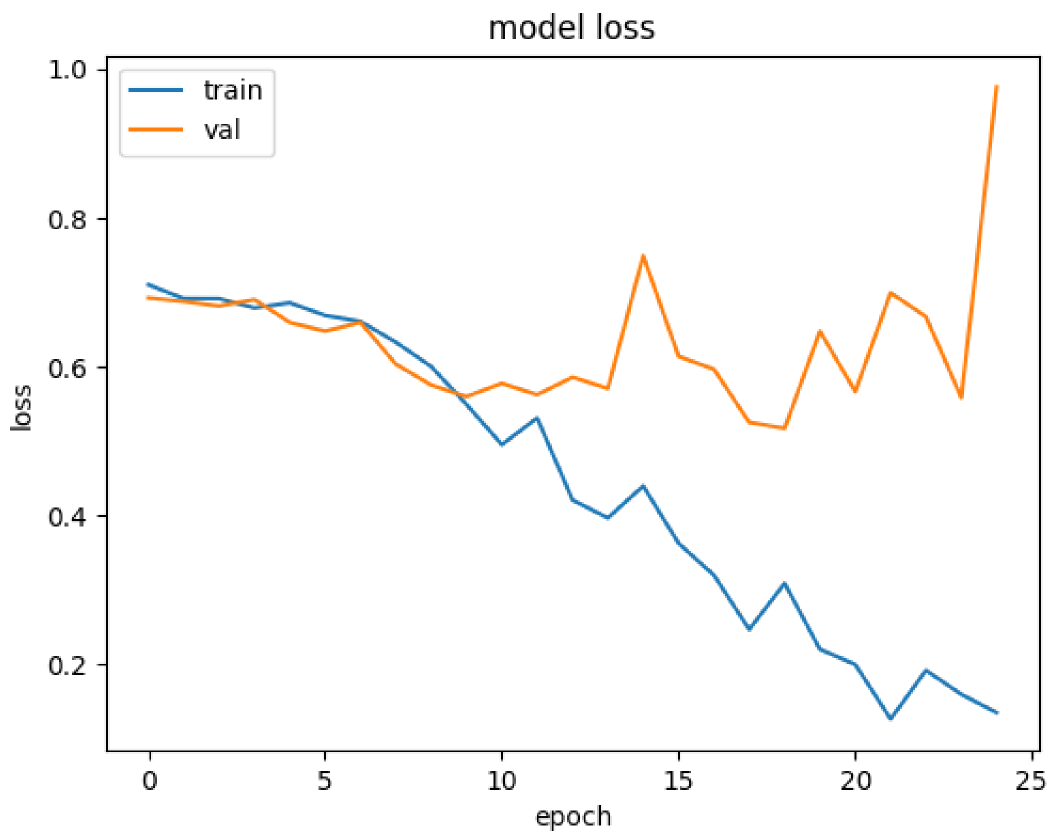
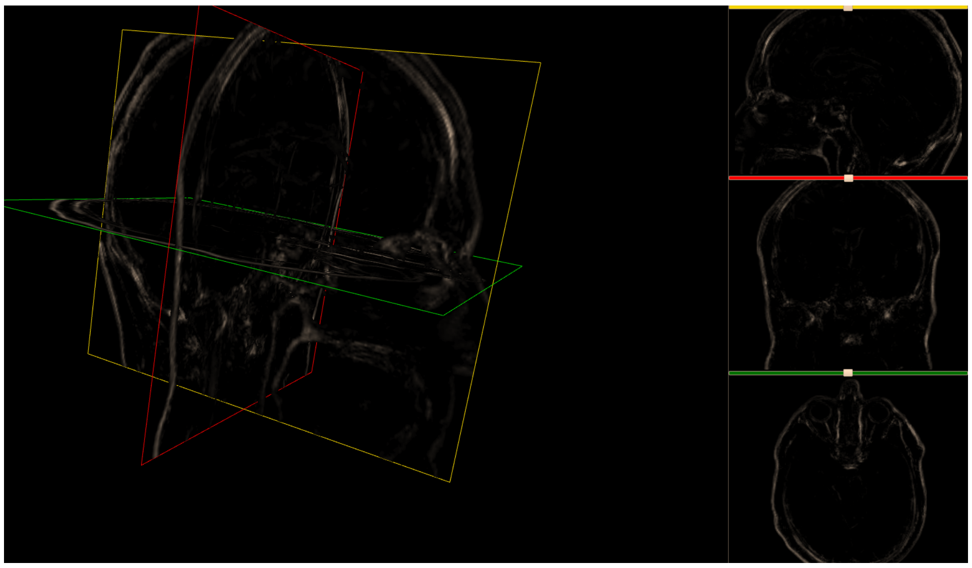


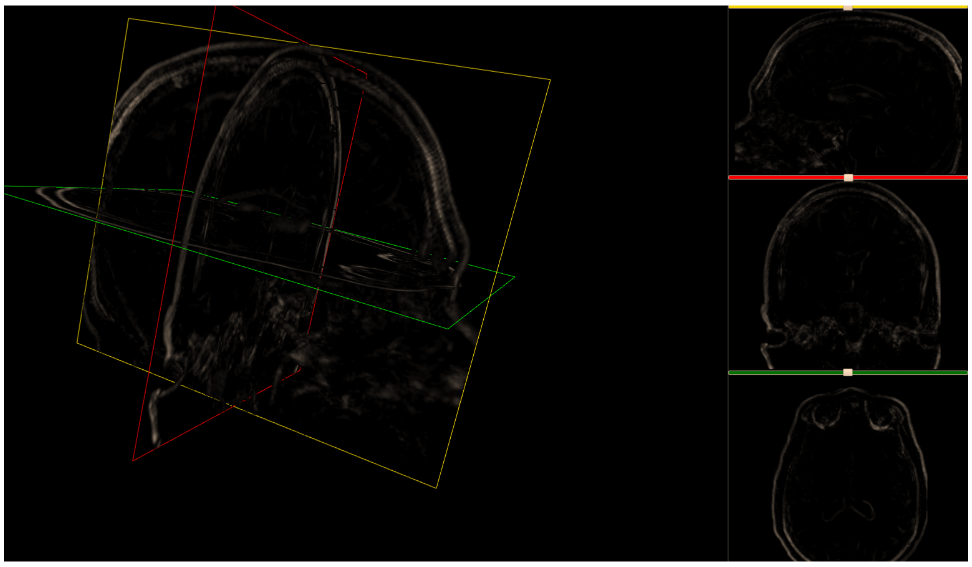
Figure 5.12: Loss curve for the no-regularization 3D model, trained for CN versus AD.

isometric view of the two maps, whereas fig. 5.16 shows a closer look at the axis slices. Upon closer inspection, the fold tissues are shown to be the regions with the highest activity density. Note that both figures were artificially brightened in order to improve visualisation.

Comparing these two maps highlights the issues that 2D implementations face in this problem. While the trained axial slice maps are largely the same between 2D and 3D, the information between the slices does not fully translate even when combining the slices together. The isometric view and the first two axis views show a sort of noise in the maps. This is a direct result of the 2D convolutions occurring in isolation, causing information along the other axes to become lost or misinterpreted. The axial slices are also slightly different, with the 2D heat-maps showing more active / lit up regions. This is likely due to each 2D model needing to draw more information from each individual slice.

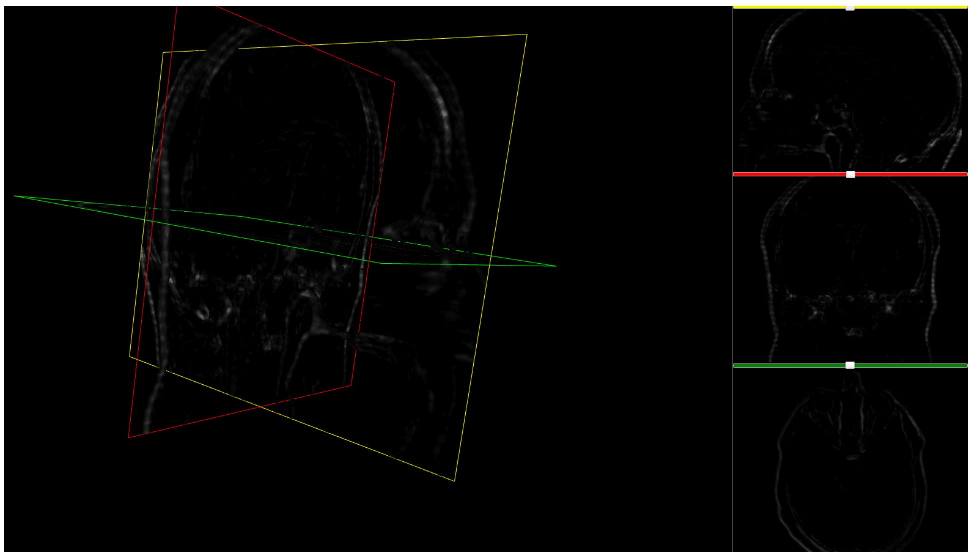


(a) CN Image.



(b) AD Image.

Figure 5.13: Visualisation of the activation maps produced by the 3D model while predicting on two different classes of image.


(a) CN Image.

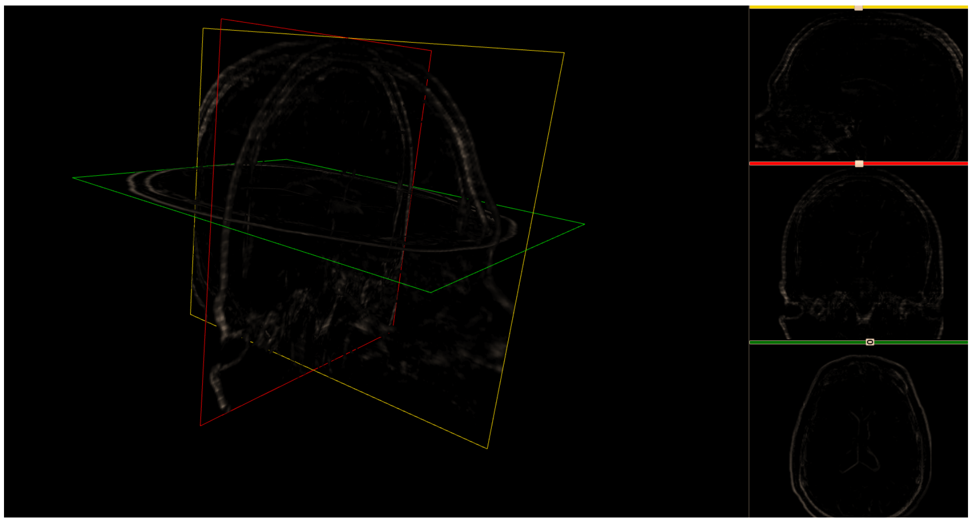
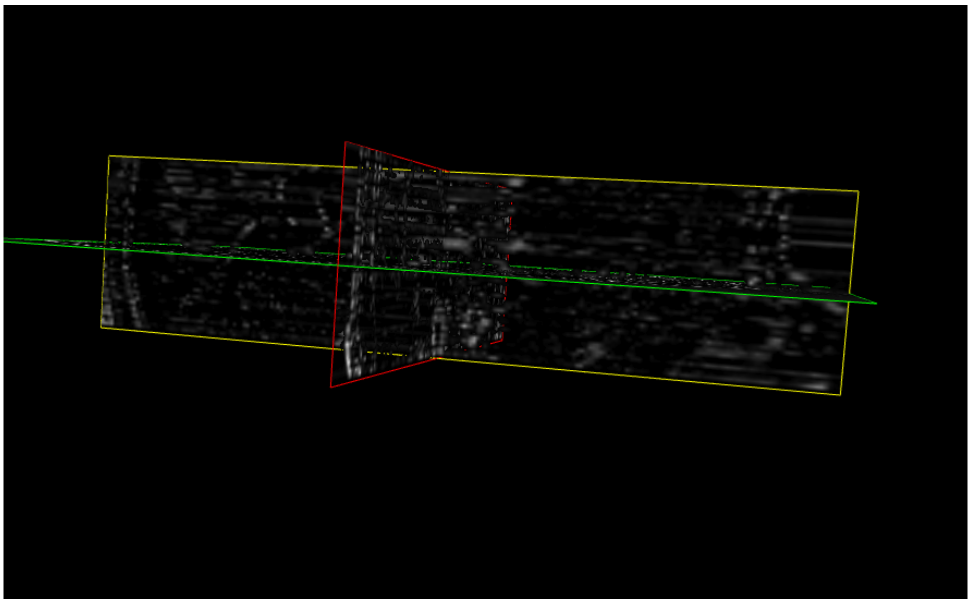
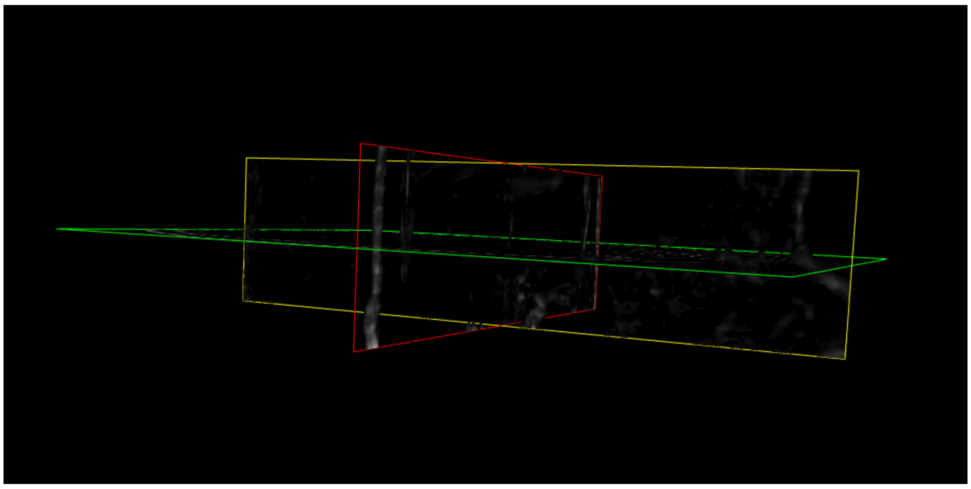

(b) AD Image.

Figure 5.14: Visualisation of the gradient maps produced by the 3D model while predicting on two different classes of image.

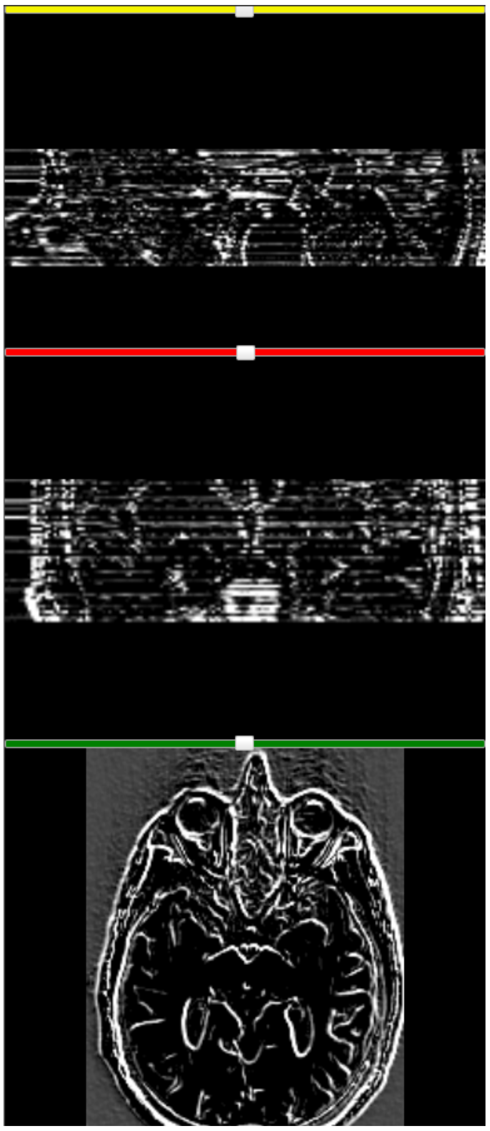


(a) Combined 2D Maps.

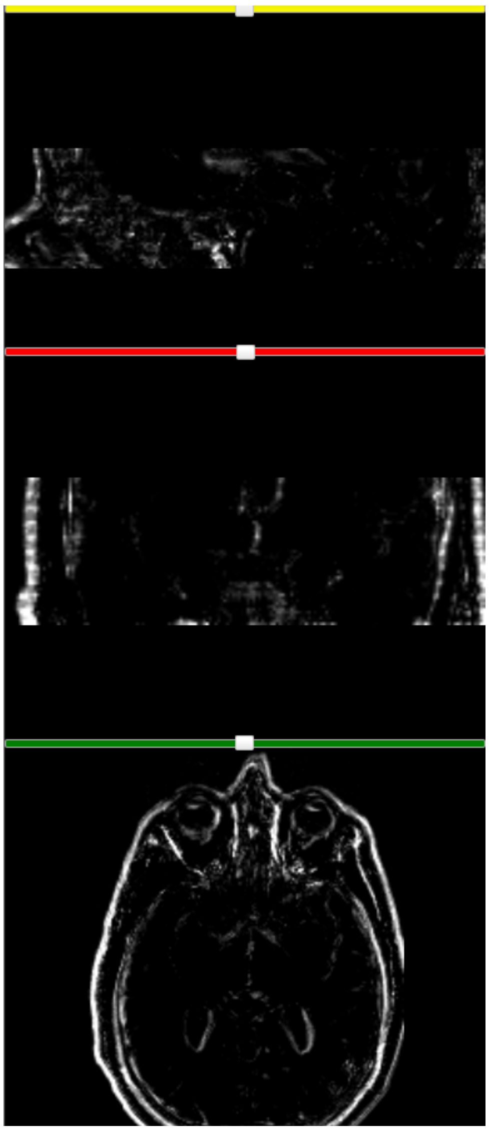


(b) 3D Map.

Figure 5.15: Visualised gradient map volumes for the combined 2D slice models, as well as the 3D model. Both maps used an AD input image.



(a) Combined 2D Maps.



(b) 3D Map.

Figure 5.16: Midpoint axis slices of the gradient maps for the combined 2D slice models, as well as the 3D model. Both maps used an AD input image. Pixel brightness was artificially increased to improve visibility.

Chapter 6: Discussion

While the results chapter contains some discussion relating to the experimental data, this chapter attempts to summarise and discuss the outcomes of the work on a broader level. It considers the combined outcomes of all previous analysis and uses this to make observations on the original question of whether a 3D or 2D approach is best for this problem. Assessment is made on the superiority of using a 3D model, as well as the overall validity in using learning models for identifying Alzheimer's disease in brain scans. Finally, this chapter takes a look at the practicality of implementing such a model in a real-life medical scenario.

6.1 3D versus 2D Classification

The initial hypothesis of this dissertation stated that, given volumetric image data as an input, utilising a 3D-based classification model would prove to be a superior choice over a similar 2D architecture. The work conducted here has proven this fact, however it has also brought up several points of discussion. Firstly, the 3D model design had a mean accuracy of 81.3%, making it the best performing model. However, it must be stressed how minor the difference in performance between the models ended up being. The 2D subject-level model held a mean accuracy of 80%, while the 2D slice ensemble had a mean accuracy of 80.6%. This disparity is actually quite minor. The 3D models as still deemed the best due to having a better spread of results, all three models were deemed successful designs. This presents the notion that, while 3D may be better, and simpler, to implement for an isolated volumetric data problem, 2D-based approaches are also perfectly viable. In other words, such a problem can still be solved with only 2D convolution layers. Additionally, the 2D subject-level model actually yielded better loss scores than the 3D model. While this doesn't change the final verdict, as the mean accuracy was still lower, it shows that the 3D model did not outperform the 2D approaches in every way. Ultimately though, the 3D model required almost no additional effort to implement compared to the 2D models, making cases where one would prefer to utilise a 2D approach rare. One such case could involve a dataset wherein the images are split into a collection of 2D slice images, and there is concern over data lost in between the slices.

The slice-based model's implementation was more complicated, but it did perform slightly better than the other 2D model (in terms of mean performance as well as variance). This was impressive to see, in spite of the design's limitations. The design has to make an assumption that all input images are aligned similarly, so that the slice of one model aligns with that of another. While this is true of orientation, there is slight variation between brain positioning between scans. However it seems this did not hinder the model much, as there must be enough overlap of the regions of interest in images to mitigate the issue. Furthermore, this design exhibited significantly less variance than the other 2D model, with similar mean accuracy. If one were to approach this problem using 2D convolution, the slice-based method may the better option, despite the increased complexity. Other works have made similar attempts to isolate 3D zones of interest in images, however utilising predetermined 2D slices is a novel approach offered only by this work. Even with a rudimentary approach to selecting the slices, this design shows that one does not need the full volume of an image in order to extract features. While it is difficult to make assumptions for all kinds of data, this work proves it in the case of Alzheimer's prediction. Additionally, this design presents an obvious option for further optimisation in the selection of the slices.

Across all three models, the CN versus AD classification scenario performs significantly better than the other case. This common pattern shows that the model faces difficulty in handling MCI-class images, regardless of the type of model design used. This is very likely due to the models struggling to differentiate MCI samples from CN ones. MCI samples represent earlier states of the disease, and thus almost by definition their features are less pronounced. The fact that the CN versus MCI case performs worse despite having significantly more data available to train on only reinforces this notion. This has certain implications that will be discussed in the next section. What matters here is the realisation that regardless of the optimisation done to the model, it can be surmised that there will always be a drop in performance when considering MCI versus AD cases.

The gradient mapping showed how the 2D slice-based model produces similar gradients to the 3D model, at least on a per-slice basis. The problem came in when attempting to look at the information between slices, which was distorted. This behaviour can be extrapolated to apply to 2D approaches in general, as by definition their convolutions are limited to 2D slices. The distortion explains why, despite learning well on individual slices, the 2D models do still produce slightly worse results than the 3D model. The map visualisations also showed that the regions of interest were located across the full volume of the model. Based on this, it is suggested that training inputs use the full brain volume, rather than attempting to concentrate on any sub-regions.

Finally, the topic of the real-time cost of training models was considered. As illustrated in the results section, the models were all trained in less than ten hours, with the 3D approach only taking a bit over five hours. Compared to other large-scale machine learning tasks, this is quite desirable, though these times were based on the CN versus AD classification scenario. Early training attempts using the full dataset took up to three days. Though this is a larger margin, and stands to benefit from efficient design, it is still significantly lower — for all three models — than is average for most natural image learning tasks. This dataset, like most medical datasets, is significantly limited in performance by a lower volume of samples compared to other natural-image cases. As such, it is believed that with the current architectures available, this problem stands to gain more from an increase in available data, than would be lost in the increase in training time. This is especially true when considering the way in which this dissertation handled data loading — one could drastically increase the size of the dataset without evoking memory issues as only one batch of images is ever present in memory, and there is still room for a significant increase in training times before it would become impractically slow.

6.2 Classifying MRI Brain Scans

Several training variables experimented with ended up producing results worth analysis. In this section, they are assessed in greater detail beyond their initial discussion.

For pre-processing steps, the results obtained made a clear impression that pixel rescaling/ normalization yields tangible benefit when attempting to build a model for this sort of problem. For the pixel rescaling, it was expected that its implementation would have a noticeable impact on the model, as many works have cited its importance for developing an image-classifier model. Without any normalization, the model saw a drop in accuracy of approximately 7%, putting it 4% below the expected mean. Local normalization and global normalization, with trimming of values above a certain threshold, performed the best, but they had very similar performances. The best and worst normalization functions had a difference of 4% in accuracy, which does actually just fall within the model's recorded standard deviation. As such it is difficult to conclusively say that either top performer is strictly superior to the other options. However, what it does convey is that so long as some form of image rescaling is employed, an increase in performance can be expected. This will not necessarily be the case for all datasets, but there are some reasons why it is the way it is for our case. The pixel ranges across the dataset were quite varied, thus causing the global normalization to struggle with certain data elements, as no chosen median value would apply well to every image. This was especially so in some cases where entire images would be comprised of pixels with values greatly above the median. By instead computing locally, images with higher pixel values than usual were treated the same as other images, thus keeping training uniform. Despite the low differential between the results, local per-image normalization was chosen as the best option due to being more reliable, as well as having the additional benefit of not requiring pixel median / maximum values to be calculated beforehand (which is needed for the other functions).

For processing the input data, augmentation was also shown to have a positive impact. Without augmentation, the model saw an accuracy drop equivalent to the image rescaling test (4% below the mean, and outside the IQR). This is hardly novel, as augmentation is often recommended for learning models. Augmentation produces more diverse training data, as it produces images with sufficient differences to give the model new information to train with. Therefore, in a scenario like this one with limited data, it helps greatly to reduce over-fitting. Much like with the rescaling, removal of augmentation does not critically compromise the model, however its inclusion is still of value. While only the results for the 3D model were shown, it was straight-forward to implement 3D transformation techniques that would directly translate between 3D and 2D data.

Similarly, regularization was also found to offer a positive impact on model training. While its use is commonly accepted in most learning models, it was valuable to observe the differences in the training curves that removing it produced. Removing the regularization layers still allowed the model to produce acceptable results, as shown by the 80% accuracy. However, as demonstrated by the training curves, it greatly destabilizes model training and limits its potential due to the over-fitting that will occur quite early into the training process.

In contrast, utilising skull stripping was found to have a negative effect on the model. Other similar works have made conclusions that skull stripping offers negligible impact, however the loss in accuracy observed here further dissuades from its use. The recorded accuracy, 74%, is the same as one of the outlier scores in the k-fold test, and could thus possibly be unrelated to the process. However, this would still suggest that the stripping offered no positive impact and this, alongside the difficulty of implementing this process, makes it difficult to recommend such action. Acquiring brain-matter maps of every image, then processing them using a separate module and generating new cropped images, is not worth the minuscule, or in this case negative, impact it offers. Perhaps this would be more valuable in a segmentation task where the boundary between brain matter and skull is more relevant, but for this problem its use is not advised.

6.3 Machine Learning and Medical Prediction

With all the results laid out, we can set about discussing what it means in the greater context of medical predictions and the Alzheimer's problem. The models developed all showed promising results — enough to, from a non-expert perspective, show potential in classifying brain scans. Furthermore, they would likely hold similar merit in terms of classifying other types of volumetric medical scans, assuming the data is at least similarly feature-rich. The most common obstacles faced with those types of datasets is that of low sample sizes and large input images (in terms of memory) — both of which are factors taken into account in this dissertation. The work conducted also showed that a variety of model designs can produce these sorts of results, as each model was able to produce an admirable evaluation score (80% or above).

In terms of the Alzheimer's scenario, the main obstacle appears to be the reduced performance when dealing with MCI images. MCI samples are quite vital as they represent the disease in its earliest stages, when cognitive symptoms may not yet have manifested and detecting via a scan is critical. By the time a patient/sample is classified as "AD", the outward symptoms are very often obvious enough that a brain scan classifier would not be needed to detect it. Nonetheless, the CN versus MCI classifier still produced good results, and with further optimisation this performance could be further improved. Based on the patterns observed across the three models, a design further optimised to better improve the MCI case detection would almost certainly work on the 2D approaches alongside the 3D approach. Therefore, all the findings from this study would still bear the same relevance.

While the gradient-maps highlight a pattern in the brain regions that yielded activations in the models, this work did opt not to do an in-depth investigation into the nature of these features. Rather, focus was placed on the viability of developing a model that would require minimal assistance / medical expertise to construct. To this end, speaking in terms of practical application of the model, the findings are promising, as the models demonstrated were able to pick up on Alzheimer's-positive patterns even with minimal feature extraction. However, when making diagnoses on patients with a real, potentially life-threatening disease such as Alzheimer's, any amount of loss can have dire ramifications. While the performance of the proposed models shows great promise, it is unlikely the model would ever reach a perfect prediction rate. Misclassifying someone with Alzheimer's could incur unnecessary costs in follow-up evaluations, and misclassifying them as negative could result in the disease being ignored until it is too late. For this reason, the proposed practical application is to use such a model in conjunction with a radiologist or similar professional. The implementation would involve setting up the model to automatically run in the background of brain scans — even those taken for unrelated reasons. This allows background tests to be conducted on people without incurring any costs beyond the initial cost to set up the model. Then, should the model produce a positive prediction, a radiologist could investigate to either confirm or refute the prediction. In the case where the scan was taken explicitly for the purpose of investigating dementia, the medical professional could double-check on any conflicting predictions. Even medical professionals can make mistakes, however using a conjoined approach of a model and a human allows them to cover up each other's mistakes. With the current model's performance, compounded with the expert's verdict, would yield a high degree of accuracy. The encoded label output of the model is also valuable here, as it would allow the professional to assess the model's weighting of each class in a given prediction. The only obstacle would be the model's difficulty with MCI cases, for which a practical version would need to be further optimised. In practice, the greater importance is expected to be placed on detecting early cases. Finally, the model design utilised in this situation would be the 3D model. The availability of raw volumetric scans would negate the benefits of the 2D models, and thus the greater efficiency of the 3D approach would be best.

Chapter 7: Conclusions

This chapter covers the closing remarks and conclusions of the dissertation. First, conclusions are given for the actual experiments conducted on the three model designs. Following this is a set of conclusions made for the potential practical application of such models in the real world. Finally, the shortcomings of the work are listed alongside a list of possible additions that could be implemented by future works.

7.1 Conclusions For the Experiment

Three models were proposed by this work: the 3D model, the 2D subject-level model using channels as a means to process the third dimension, and the 2D slice-level combined models. All three of the models were found to produce respectable results. Ultimately the 3D model was deemed the highest performing one, which confirms the hypothesis that a problem dealing with raw 3D image scans should use 3D model architecture. The final version of the 3D model achieved an average accuracy of around 82% for classifying cognitively normal scans versus Alzheimer's scans, in addition to a low loss score with minimal variance. This level of performance acts as a proof of concept that CNN model designs have potential for predicting Alzheimer's in patients using only brain scan input data.

However, the 2D designs were only marginally less accurate (1-2%) than the 3D model, and both performed better than expected and yielded very good results on their own. While the 3D deep learning model will typically be the best approach due to low variance and simplicity in implementation, the 2D models offer viable alternate strategies. The novel concept offered in this work is the design of the slice-based 2D model. The design utilised several independent 2D models, each trained to specialise in a particular slice of the data known to contain significant feature data. The predictions of the slices can then be combined using a weighted soft-voting ensemble. This design performed better than the other 2D model, and further refinement on the slice selection process and the voting algorithm could stand to improve its accuracy further. As the usage of slice subsets greatly reduces the size of input data that needs to be processed, this design has potential in offering a viable low-memory approach to building a model.

Additionally, it was found that a model attempting to classify MCI-class images (mild cognitive impairment) will suffer a sharp decrease in performance (between 10 and 15%) compared to one dealing with AD-class (a more severe class of the disease). This is due to the less obvious visual signs of the disease within the brain, and thus less well-defined feature data. This is a problem that is present within all three models, and is one that rather requires optimisation and additional training data in order to mitigate. Classification of MCI-class patients is quite important, as it makes up the target population of individuals that possess the disease but may not yet know it. Therefore future research should place importance on optimising around this scenario, and perhaps that future data collection projects focus on expanding the training data available for the MCI class.

Investigation was also conducted around several variables within the training process, in an attempt to find strategies that yield optimal performance when training models for this Alzheimer's problem. In the end, the following design suggestions are made based on the results obtained:

1. Utilisation of a pixel rescaling and normalization function is essential during pre-processing, as it drastically affects the performance of a given model. Either use a per-image local normalization algorithm, or one based on a global threshold that trims values over said threshold.
2. Augmentation, using rotation and elastic deformation, offers a small but noticeable increase to model performance, at least when done in moderation.
3. Regularization is a valuable tool to increase model stability and to prevent over-fitting from inhibiting training potential. This applies to most learning models, but especially so in this case where the data is so limited.

7.2 Conclusions in General

Overall, the results discussion did find great potential in the practical application of such models. Models such as the ones presented in this paper possess low enough loss values as to accurate classify and identify Alzheimer's disease using only raw MRI scans. However, this conclusion came with the caveat that there is still some error, and any error can still be disastrous when human lives are involved. Thus, the best-case implementation would involve a combined effort of a model and a professional radiologist. The model would act as an early-warning system, capable of detecting and warning about the presence of the disease in any scan including the brain. This removes the need for a radiologist to check every single scan with the specific goal of looking for signs of Alzheimer's. Instead, the model would be able to perform background assessments of any scan taking place, and raise a flag when it makes a positive prediction. Then, the radiologist can act as a backup system to confirm the results of the model, or at least make the decision on whether follow-up examination is warranted or not.

As such, this dissertation puts forth the belief that the advancement of deep learning models has opened the way for machine learning to aid in detecting Alzheimer's in patients before they ever begin showing symptoms. Further investigation into this problem with more data, and further optimised architectures with more powerful hardware for training, could produce something usable in a real-life application. When it comes to Alzheimer's, an early detection grants more time to act and prepare in the worst case, and saves lives in the best case. In this endeavour, the advancement of specifically 3D CNN models currently seems the most promising option.

7.3 Shortcomings

The paper found success in most of the elements it set out to achieve, however there were certain limitations that had to be circumvented in the process. Said limitations can be found below:

- Batch size limitations: During implementation, it was found that the size of the input images would often result in out-of-memory issues. While this was solved using careful pre-processing steps, the models all still had to use a conservative batch number of three in order to preserve memory. While the project did utilise a high-speed computing server for the training process, it did not have exclusive access to this server and thus hardware resources were limited. Future attempts on a similar project would likely benefit from using greater batch numbers, as well as more powerful hardware in general.
- Classification problem scope: While the paper performs extensive experimentation and analysis surrounding the Alzheimer's classification problem, the conclusions made cannot necessarily be said to extend to other problem cases. The conclusions surrounding the superiority of 3D models may differ in other volumetric medical image classification tasks not involving brains. While it is assumed that the results

would not change drastically, investigation into other parts of the body holds value in the discussion surrounding 3D versus 2D model design.

- Data limits: The experiments performed for this work exclusively used the ADNI dataset. While there are other datasets available, such as OASIS, the large difference in image data made it too costly to attempt to optimise a model around multiple sets.

7.4 Future Improvements

Alongside the limitations presented above, there were also several elements that, while offering unique points of further interest to the discussion, fell outside the scope of the project:

- Different model types: The scope of the work is limited to CNN models, as this model design has been shown to produce the most reliable results. There are still several other types of model that could be investigated in order to assess if the difference between 3D and 2D approaches would behave similarly with other model types.
- Segmentation: Segmentation is a complex enough topic to prevent its inclusion in this dissertation, as it would require significant focus and investment that would detract from the other aspects. Still, it offers another facet of experimentation around Alzheimer's detection in the form of identifying regions of the brain that correspond to dementia features.
- Additional datasets: As mentioned previously, configuring the model to train on more datasets than just ADNI would help alleviate the limitations of the limited data available for training. This presents several obstacles, such as different file structures, labelling systems, and image pixel ranges. However, the potential benefit of doing so, in terms of improving model accuracy and flexibility, would make the endeavour worthwhile.
- Further optimisation: The models developed for this project were optimised to the point of producing sufficient results to make informed comparisons and discussion. However, there is still room for further improvement in terms of architecture of hyperparameters. Certain papers were able to report accuracies in the 90-percentile range. Therefore, should one wish to extend upon the models in this work, further refinement is an obvious place to start.

Bibliography

[1] Deepika Bansal, Kavita Khanna, Rita Chhikara, Rakesh Dua, and Rajeev Malhotra. Classification of magnetic resonance images using bag of features for detecting dementia. *Procedia Computer Science*, 167:131–137, 01 2020.

[2] Brankica Bratić, Vladimir Kurbalija, Mirjana Ivanovic, Iztok Oder, and Zoran Bosnic. Machine learning for predicting cognitive diseases: Methods, data sources and risk factors. *Journal of Medical Systems*, 42, 10 2018.

[3] Matthew Brett, Christopher J. Markiewicz, Michael Hanke, Marc-Alexandre Côté, Ben Cipollini, Paul McCarthy, Christopher Cheng, Yaroslav Halchenko, Satra Ghosh, Eric Larson, Demian Wassermann, Stephan Gerhard, and Ross Markello. nipy/nibabel package, June 2022.

[4] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N. Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 839–847. IEEE, 2018.

[5] Carlton Chu, Ai-Ling Hsu, Kun-Hsien Chou, and Peter Bandettini. Does feature selection improve classification accuracy? impact of sample size and feature selection on classification using anatomical magnetic resonance images. *NeuroImage*, 60:59–70, 12 2011.

[6] Michael Ewers, Cathal Walsh, John Trojanowski, Leslie Shaw, Ronald Petersen, Clifford Jack, Howard Feldman, Arun Bokde, Gene Alexander, Ph Scheltens, Bruno Vellas, Bruno Dubois, Michael Weiner, and Harald Hampel. Prediction of conversion from mild cognitive impairment to Alzheimer's disease dementia based upon biomarkers and neuropsychological test performance. *Neurobiology of aging*, 33:1203–14, 12 2010.

[7] Jie Fu, Yingli Yang, Kamal Singhrao, Dan Ruan, Fang-I Chu, Daniel A. Low, and John H. Lewis. Deep learning approaches using 2D and 3D convolutional neural networks for generating male pelvic synthetic computed tomography from magnetic resonance imaging. *Medical Physics*, 46(9):3788–3798, 2019.

[8] Karol Gotkowski, Camila Gonzalez, Andreas Bucher, and Anirban Mukhopadhyay. M3D-CAM: A PyTorch library to generate 3D data attention maps for medical deep learning, 2020.

[9] Juan Eugenio Iglesias, Cheng-Yi Liu, Paul M. Thompson, and Zhuowen Tu. Robust brain extraction across datasets and comparison with publicly available methods. *IEEE transactions on medical imaging*, 30(9):1617–1634, 2011.

[10] Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, François-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte, et al. imgaug. `https://github.com/aleju/imgaug`, 2020. Online; last accessed 01-Feb-2022.

[11] Bijen Khagi and Goo-Rak Kwon. CNN model performance analysis on MRI images of an OASIS dataset for distinction between healthy and Alzheimer's patients. *IEIE Transactions on Smart Processing & Computing*, 8:272–278, 08 2019.

[12] Shen Liu, Zhang. Ensemble sparse classification of Alzheimer's disease. *NeuroImage*, 2012.

[13] M. López, J. Ramírez, J.M. Górriz, I. Álvarez, D. Salas-Gonzalez, F. Segovia, R. Chaves, P. Padilla, and M. Gómez-Río. Principal component analysis-based techniques and supervised classification schemes for the early detection of Alzheimer's disease. *Neurocomputing*, 74(8):1260–1271, 2011. Selected Papers from the 3rd International Work-Conference on the Interplay between Natural and Artificial Computation (IWINAC 2009).

[14] Dr. Manasi Patil and Anil Yardi. MLP classifier for dementia levels. *International Journal of Modeling and Optimization*, 01 2011.

[15] Adrien Payan and Giovanni Montana. Predicting Alzheimer's disease: a neuroimaging study with 3D convolutional neural networks. *CoRR*, abs/1502.02506, 2015.

[16] Philippe Remy. Keract: A library for visualizing activations and gradients. `https://github.com/philipperemy/keract`, 2020. Online; last accessed 23-Sep-2022.

[17] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[18] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.

[19] Zahraa Sh and Hawraa Abbas. Classification of Alzheimer's disease based on several features extracted from MRI T1-weighted brain images. *Kerbala Journal for Engineering Science*, 01 2022.

[20] M.V.F. Silva, C.d.M.G Loures, L.C.V Avles, L.C.d.S Souza, K.B.G Borges, and M.d.C Carvalho. Alzheimer's disease: risk factors and potentially protective measures. *Journal of Biomedical Science*, 26, 5 2019.

[21] Roman Solovyev, Alexandr A Kalinin, and Tatiana Gabruseva. 3D convolutional neural networks for stalled brain capillary detection. *Computers in Biology and Medicine*, 141:105089, 2022.

[22] Halebeedu Subbaraya Suresha and Srirangapatna Sampathkumaran Parthasarathy. Alzheimer disease detection based on deep neural network with rectified adam optimization technique using MRI analysis. In *2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, pages 1–6, 2020.

[23] Junhao Wen, Elina Thibeau-Sutre, Mauricio Diaz-Melo, Jorge Samper-Gonzalez, Alexandre Routier, Simona Bottani, Didier Dormont, Stanley Durrleman, Ninon Burgos, and Olivier Colliot. Convolutional neural networks for classification of Alzheimer's disease: Overview and reproducible evaluation. *Medical Image Analysis*, 63:101694, 05 2020.

[24] Ekin Yagis, Luca Citi, Stefano Diciotti, Chiara Marzi, Selamawet Workalemahu Atnafu, and Alba G. Seco De Herrera. 3D convolutional neural networks for diagnosis of Alzheimer's disease via structural MRI. In *2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS)*, pages 65–70, 2020.