Final Project - COMP798 Research

Akhil Perimbeti

May 4, 2023

Deep Neural Networks for Classification of Alzheimer's Disease

Alzheimer's disease (AD) is a progressive and debilitating neurodegenerative disorder that affects millions of people worldwide. It is caused by structural changes in the brain and leads to the deterioration of cognitive functions. According to the World Health Organization (WHO), there are currently over 50 million people living with dementia, and AD is the most common cause of dementia, accounting for 60-70% of cases. The detection, stage classification and accurate diagnosis of Alzheimer's disease is essential for timely medical interventions for slowing disease progression, and can have a significant impact on improving quality of life for those affected. However, accurate detection classification of the stage of the disease is challenging due to the complex nature of the disease and its progression. Traditional methods for diagnosis and classification rely on cognitive and neuropsychological testing, which can be costly, time-consuming and subjective.

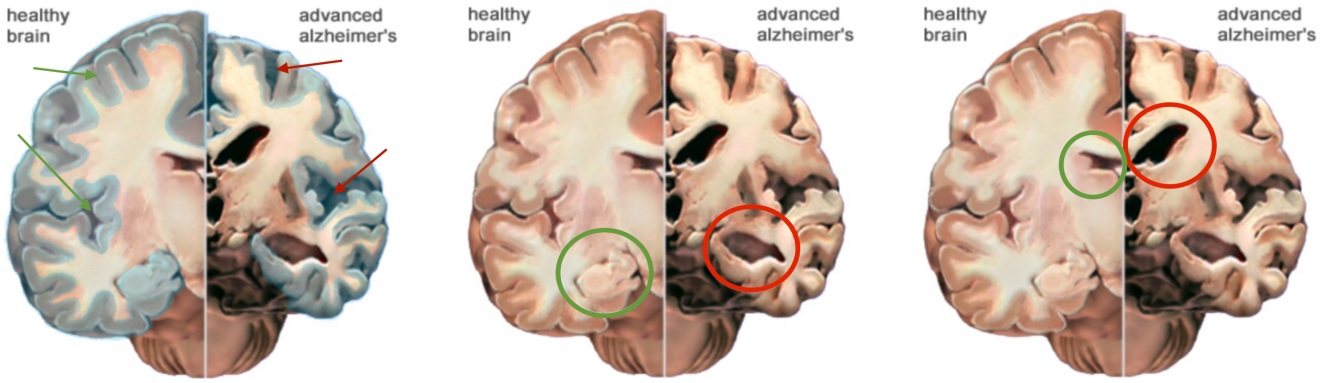
The use of machine learning techniques can provide a cost-effective and reliable method for the identification and detection of the disease. The identification and classification of Alzheimer's Disease using deep learning architectures can significantly improve the accuracy and reliability of the diagnostic techniques used, which can have a significant impact on improving the quality of life for those affected by the disease. Recent advances in medical imaging and machine learning have shown promise in identifying the disease at an earlier stage and with higher accuracy. One of the most promising areas of research is the use of deep learning algorithms for medical image analysis. Deep learning models have shown great success in a variety of tasks, including image classification, object detection, and segmentation, and have been applied to medical imaging for a range of conditions, including cancer, neurodegenerative diseases, and cardiovascular disease. In particular, convolutional neural networks (CNNs) have been used for the classification of Alzheimer's disease from brain MRI scans with promising results.

This research project proposes to investigate the use of deep learning CNN architectures to accurately identify Alzheimer's disease at its various stages. The feasibility of the proposed approach will be explored using various state-of-the-art models, including VGG16, VGG19, DenseNet121, DenseNet201, ResNet50 and ResNet152. The 3D MRI brain scans were pulled from the OASIS (Open Access Series of Imaging Studies) database, with each image being of size $[256 \times 256 \times 160 \times 1]$. The variations of these specific models include: A series of regression models, a series of binary classification models and a series of multiclass classification models. Each of these model series will have further comparison of those loaded with pre-trained model weights from the *'imagenet'* database (converted from 2D to 3D) vs. those that were trained from scratch (TFS). The comparisons are made using accuracy, sensitivity, specificity, and F1 score. Finally, model interpretability will be explored by using saliency maps and GRAD-CAM heatmap outputs to display the transparency of the model's decision-making process

---

Problem Introduction

The disease is characterized by the accumulation of amyloid plaques and tau protein tangles in the brain, leading to the loss of neurons and synapses, and ultimately resulting in cognitive decline, memory loss, and behavioral changes. It leads to a gradual slow progression of impaired memory and cognition, leading to a loss of independence in activities of daily lives. Early detection and classification of AD is crucial for providing timely treatment and improving patient outcomes. The diagnosis of Alzheimer's diseases is currently based on a combination of clinical evaluation, cognitive tests, and neuroimaging techniques, such as magnetic resonance imaging (MRI). Structural MRI, in particular, can provide detailed information about the brain's structural changes associated with AD, such as ventricular enlargement, hippocampal atrophy and cortical thinning.



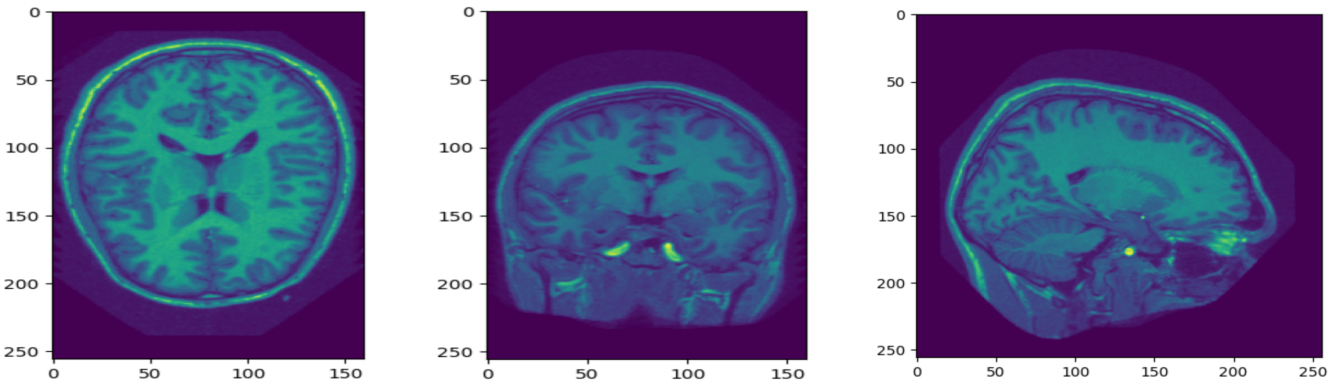
(a) Cerebral Cortex Volume Loss/Shriveling

(b) Atrophy of the Hippocampus and Amygdala

(c) Lateral Ventricular Enlargement

Figure 1: Neurological indicators of advanced Alzheimer's Disease (AD) - Coronal Slice

Alzheimer's disease primarily affects specific regions of the brain that are involved in memory and learning. As Figure 1(a) above shows, one of the earliest indicators of the onset of AD is the loss of volume in the cerebral cortex. The cortex, the outermost layer of the brain, is responsible for higher-order thinking and complex cognitive processes. The folding of the cerebral cortex creates gyri and sulci which separate brain regions and increase the brain's surface area and cognitive ability. Another key indicator of the onset of AD, as shown in Figure 1 (b), is hippocampal atrophy and shrinkage of the amygdala. The hippocampus is a key structure in the brain for memory consolidation, spatial navigation, and learning. In Alzheimer's disease, the hippocampus is one of the first regions to be affected, resulting in a decline in memory function. The amygdala, which is involved in emotional processing, is also affected in Alzheimer's disease, resulting in changes in mood and behavior. Figure 1(c) shows another important structural change that occurs in the brain of individuals with Alzheimer's disease, although it is only observed in the later stages of the disease. This structural change is the enlargement of the lateral ventricles, which are spaces in the brain filled with cerebrospinal fluid (CSF) that help to cushion and protect the brain from injury. In Alzheimer's disease, the shrinkage of brain tissue leads to an increase in the size of the lateral ventricles, which can be easily seen on brain imaging scans.



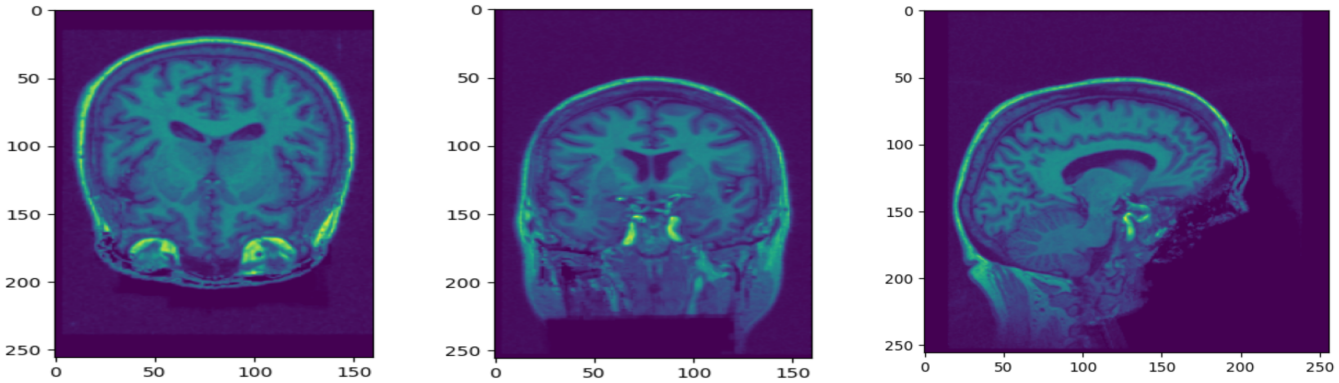
(d) Axial Plane - Slice: 130

(e) Coronal Plane- Slice: 120

(f) Sagittal Plane - Slice: 64

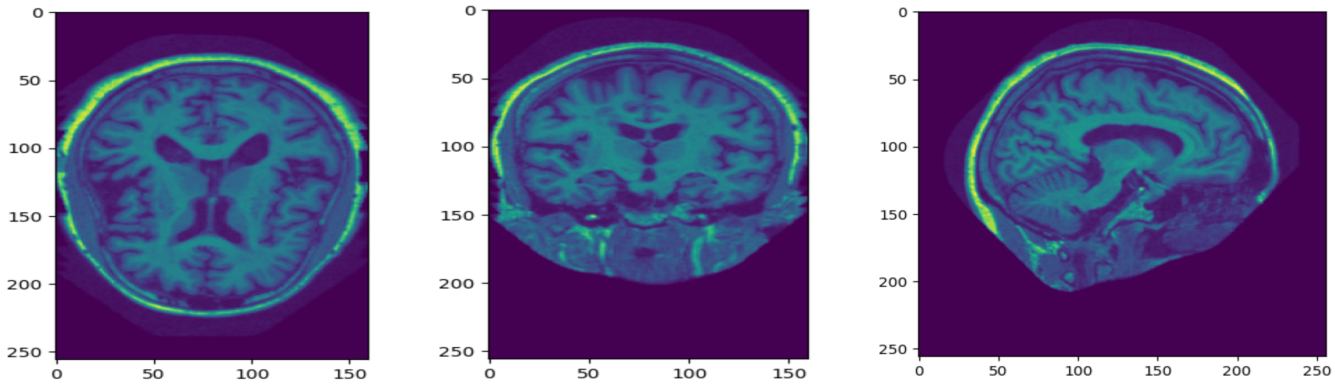
Figure 2: **Patient ID: 0005_MR1**, Given **CDR: 0**, Healthy control, No AD

Figure 2 allows us to see the typical cortical volume of a healthy individual (Subject 5). We can compare this to the severe loss in brain density and volume seen in a person with Advanced Alzheimer's disease in to the subject in Figure 5 (Subject 351) , as indicated by the enlarged ventricles, and loss of volume in the gyri and hippocampus. The characterization of atrophy is still subjective, However, the progression of AD still follows the Braak pathologic staging. The Braak pathologic staging is a method that permits a sufficient differentiton between the intital, intermediate and late stages of Alzheimer's disease. In MCI and early (mild) AD, atrophy is seen primarily in the entorhinal cortex (cerebral cortex in the medial temporal lobe) and the hippocampus, as seen in Figure 3. As AD progresses,



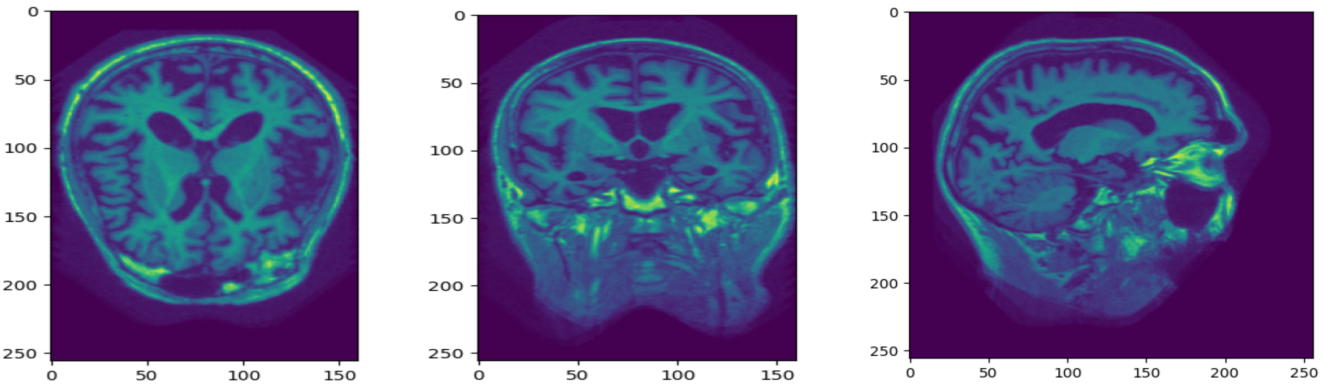
(a) Axial Plane - Slice: 128 (b) Coronal Plane- Slice: 125 (c) Sagittal Plane - Slice: 70

Figure 3: **Patient ID: 0453_MR1**, Diagnosed with **CDR: 0.5**, indicating MCI to very mild AD



(a) Axial Plane - Slice: 155 (b) Coronal Plane- Slice: 128 (c) Sagittal Plane - Slice: 70

Figure 4: **Patient ID: 0052_MR1**, Diagnosed with **CDR: 1**, indicating mild to moderate AD



(a) Axial Plane - Slice: 165 (b) Coronal Plane - Slice: 128 (c) Sagittal Plane - Slice: 64

Figure 5: **Patient ID: 0351_MR1**, Diagnosed with **CDR: 2**, indicating moderate to severe AD

atrophy spreads to the rest of the medial temporal lobe (MTL) including the amygdala, hippocampus, and parahippocampal regions as indicated in Figure 4. Eventually, atrophy progresses to the rest of the cerebral cortex, primarily the temporal, parietal, and frontal lobes, although in late-stage AD, the occipital lobe is also affected. As atrophy progresses, the cerebrospinal fluid (CSF) spaces (eg, ventricles, cerebral sulci, and cerebellar folia) enlarge. Lateral ventricle size also correlates with AD status and stage of progression. Rates of ventricle enlargement are higher in

AD compared with ventricle dilation typical of normal aging. Given this, the most informative slice when it comes to visualizing Alzheimer's disease on brain imaging scans would be the coronal slice. The coronal slice provides a view of the brain from front to back, allowing for visualization of the hippocampus and other medial temporal lobe structures that are important in determining the progression stage of AD. In addition, the coronal slice provides a good view of the lateral ventricles, allowing for detection of enlargement or atrophy of these structures.

It is clear to see that the structural MRI can provide detailed information on the structural changes in the brain that are most closely associated with the onset and progression of Alzheimer's disease. However, the manual interpretation of MRI scans can be time-consuming, subject to inter-observer variability, and prone to human error. This has led to a growing interest in using machine learning (ML) techniques, specifically deep learning (DL), for the automated classification of AD based on structural MRI scans. Deep Learning architectures, particularly the convolutional neural network (CNN) frameworks, have shown great promise in the automated detection and classification of of various medical conditions, including AD. These frameworks have the ability to learn complex features and patterns from large data sets, and can improve the accuracy and efficiency of medical image analysis.

Proposed Solution

The aim of this final project is to compare the performance of various classification and regression based deep learning architectures for AD classification using 3D structural MRI data from the OASIS-1 data set. The proposed method will employ several different classification and regression models to differentiate between healthy patients and those at different stages of Alzheimer's disease progression. Figure 6 below provides an overview of the different frameworks implemented, and their respective variations. The project will involve preprocessing the MRI data, training and evaluating the deep learning models on a high-performance graphical processing unit, and comparing their performance using various evaluation metrics. The project will also investigate the interpretability of the models, using saliency maps and GRAD-CAM to analyze the most important features and regions of the brain for AD classification. The proposed approach will demonstrate the effectiveness of deep learning in medical image analysis and will provide a highly discriminative feature representation for the accurate classification of AD.

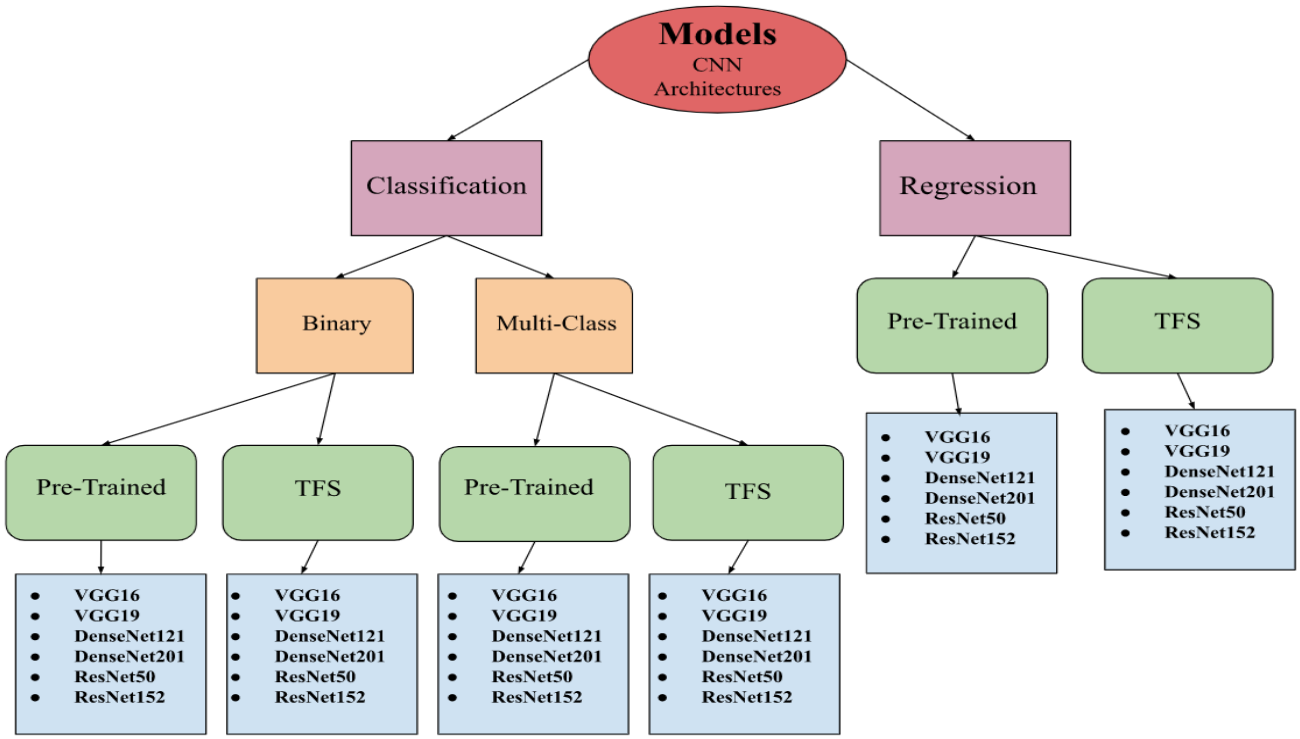
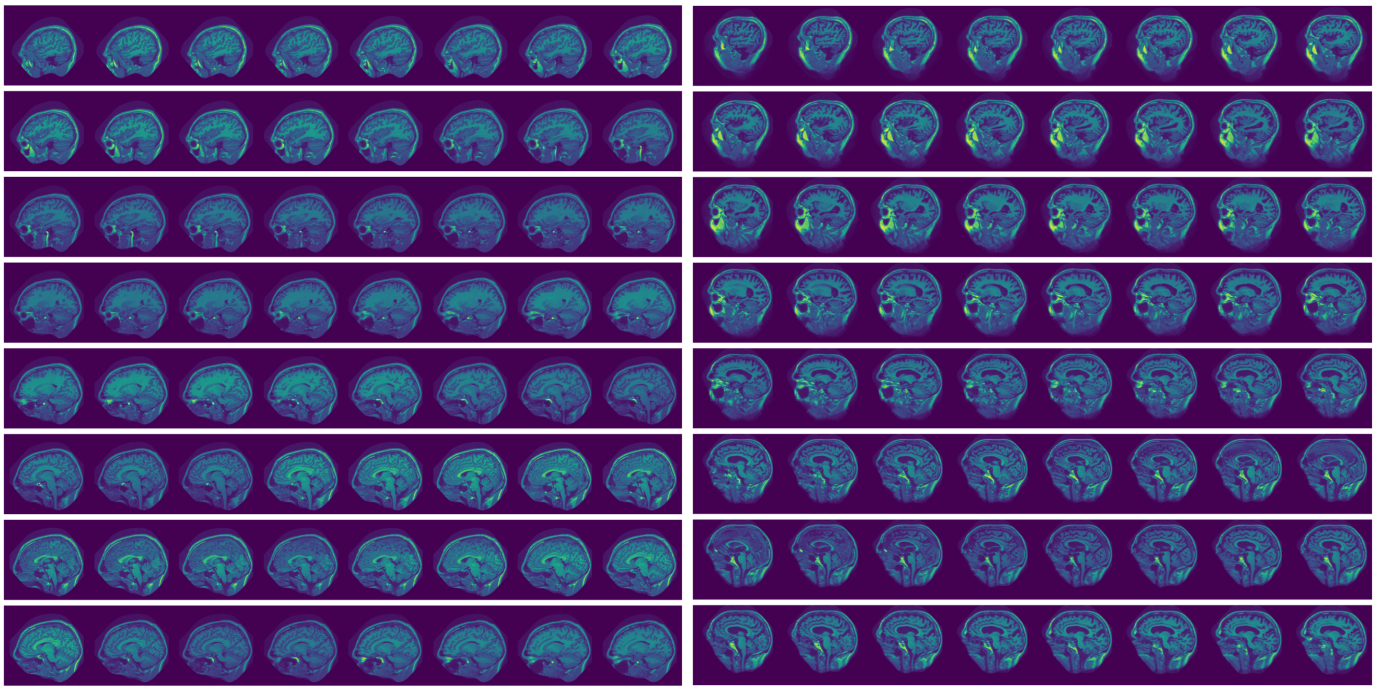


Figure 6: Overview of the CNN frameworks

The OASIS-1 Dataset

The OASIS (Open Access Series of Imaging Studies) database is a series of magnetic resonance imaging data sets that is publicly available for distribution, study and analysis. The aim of the OASIS project was to make brain imaging data more accessible to researchers and clinicians. This project will be utilizing the OASIS-1 data set from this database. The initial data set consists of a cross-sectional collection of 416 subjects aged 18 to 96 years. 100 of the included subjects older than 60 years have been clinically diagnosed with mild to moderate Alzheimer's disease. The subjects are all right-handed and include both men and women. For each subject, three or four individual T1-weighted magnetic resonance imaging scans obtained in single imaging sessions are included. Additionally, for 20 of the non-demented subjects, images from a subsequent scan session after a short delay (within 90 days) are also included as a means of assessing acquisition reliability. Automated calculation of whole-brain volume and estimated total intracranial volume are presented to demonstrate use of the data for measuring differences associated with normal aging and Alzheimer's disease. The images for every patient were pulled from the processed SUB_111 folder which contained the averaged co-registered image of the individual the individual scan images in the native acquisition space in re-sampled to 1 mm isotropic voxels. All the data was anonymized to accommodate public distribution the images were acquired on a 1.5 Tesla MRI scanner using a T1-weighted sequence, and were pre-processed to correct for intensity non-uniformity and to perform skull stripping using the Brain Extraction Tool. Each MRI brain image was of size $[256 \times 256 \times 160 \times 1]$ and the full set is 15.8 GB compressed and 50 GB uncompressed.

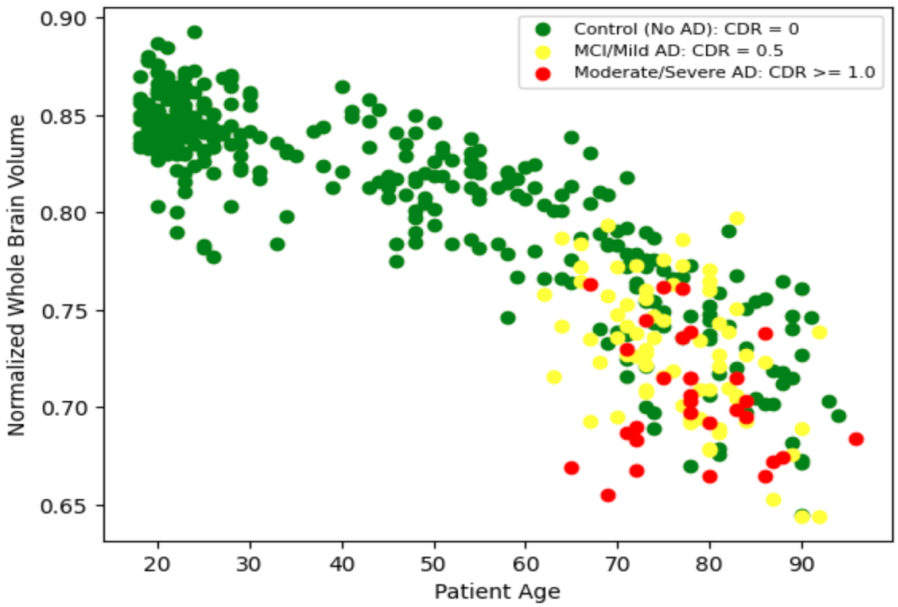
Figure 7A (a) and 7B (b) below allow us to visualize the scope of the 3D MRI image data for two subjects. The figures plot a montage of 2D slices along the sagittal plane of each subject. When plotting 3D MRI data of form $[a, b, c, channel] \rightarrow a$ refers to the axial plane, b refers to the coronal plane, c refers to the sagittal plane, and *channel* is the color display. The montage includes only 64 of the slices along the sagittal plane (Total: 160). For plotting : Subset the image $[:, :, 32 : 96, 0]$ to choose a set of slices allowing us to render the 3D image into 2D for visualization.



(a) Figure 7A: Montage of 64 2D sagittal plane slices; **PatientID: 0005_MR1**

(b) Figure 7B: Montage of 64 2D sagittal plane slices; **PatientID: 0351_MR1**

Figure 8A (a) below shows the distribution of each individual subject by normalized whole brain volume vs. age. It is clear to see that there is a decreasing trend in brain volume as the subjects increase in age, however (given the color groupings for presence of AD), there does not seem to be any actionable correlation between normalized whole brain volume and the presence of AD. The figure is presented the demonstrate the use of the clinical data for measuring the differences associated with normal aging and Alzheimer's disease, especially prevalent between subjects diagnosed with a CDR of 0 and those diagnosed with a CDR of 0.5 (mild dementia).



(c) Figure 8A: Distribution of nWBV vs. Age (color grouped by CDR score class)

		Non-Demented				Demented				
Age Group	N	n	mean	male	female	n	mean	male	female	CDR 0.5/1/2
<20	19	19	18.53	10	9	0		0	0	0/0/0
20s	119	119	22.82	51	68	0		0	0	0/0/0
30s	16	16	33.38	11	5	0		0	0	0/0/0
40s	31	31	45.58	10	21	0		0	0	0/0/0
50s	33	33	54.36	11	22	0		0	0	0/0/0
60s	40	25	64.88	7	18	15	66.13	6	9	12/3/0
70	83	35	73.37	10	25	48	74.42	20	28	32/15/1
80s	62	30	84.07	8	22	32	82.88	13	19	22/9/1
≥90	13	8	91.00	1	7	5	92.00	2	3	4/1/0
Total	416	316		119	197	100		41	59	70/28/2

(d) Figure 8B: OASIS-1 Full summary

Class Imbalance (CI) :

Figure 8B (b) above displays a full summary table of the OASIS-1 dataset. In total, there are 30 individuals diagnosed with moderate AD, 70 individuals diagnosed with mild AD, and 334 individuals (including the 20 reliability data scans) who were not diagnosed with AD. The subjects diagnosed with a CDR of 2 were grouped along with the subjects diagnosed with a CDR of 1, since there were only two individuals diagnosed with a CDR score of 2. It is clear there is a significant class imbalance in the dataset, with a majority of the images in the class belonging to Class 0 (those diagnosed with a CDR of 0). Figure 10A (a) below displays the following class breakdown →

- *Class 0*: Patients with a CDR score of 0 and all healthy controls
- *Class 1*: Patients with a CDR score of 0.5, indicating mild dementia
- *Class 2*: Patients with a CDR score of 1 or 2, indicating moderate to severe dementia

Figure 9 below displays a problem that occurred when initially implemented the transfer learning models using the naturally imbalanced training, validation and testing set. When running performance diagnostics the models to be improving over time but were actually just memorizing the minority class samples in the training set and learning to always predict the majority class for everything else. The model accuracy was relatively good at 77%, but simply

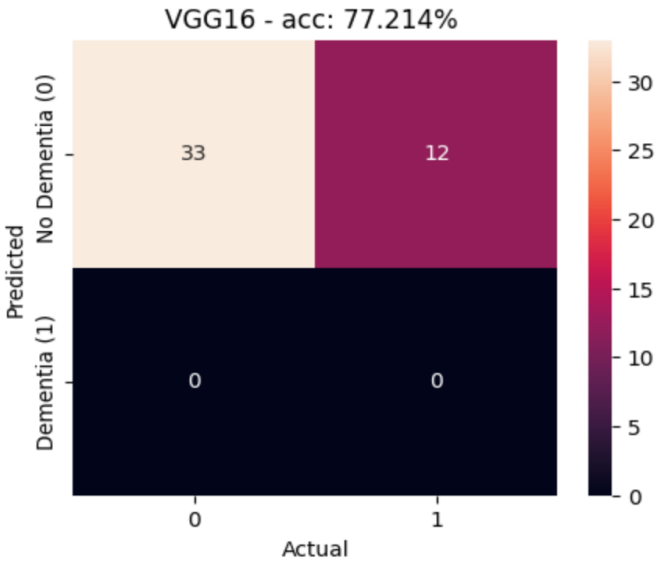
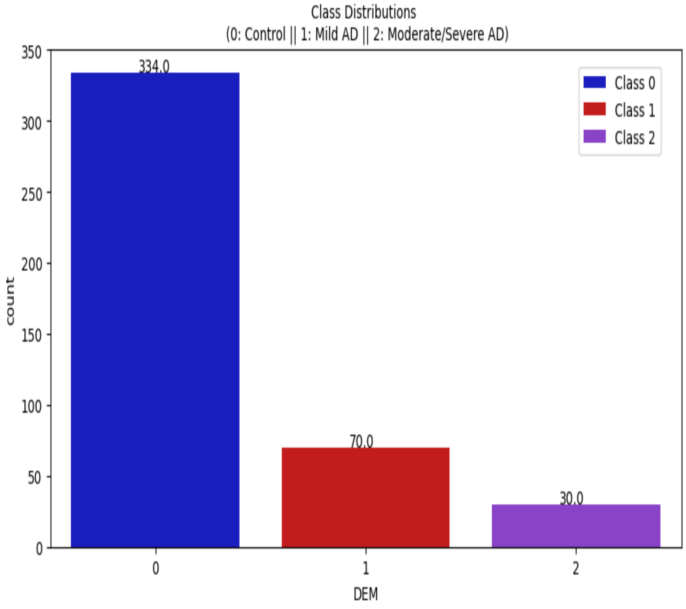
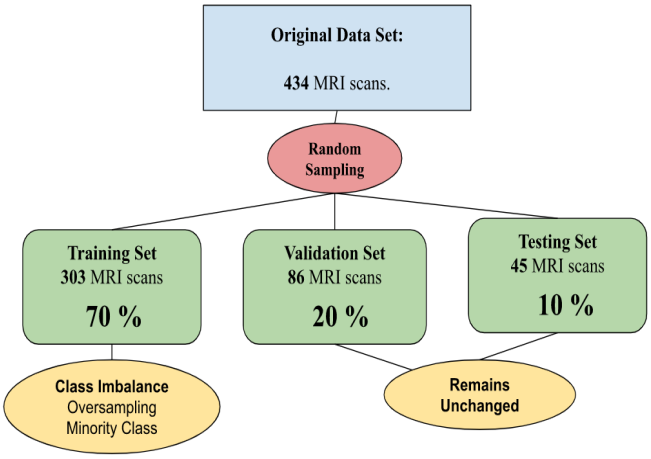


Figure 9: VGG16 Confusion Matrix, trained on naturally imbalanced data set

classified all samples in the test set as negative (No dementia - Class 0). Failing to adjust for the class imbalance in the training set will not allow the model to properly distinguish and classify the minority class. From a clinical standpoint, the focus should be on improving the accuracy of the minority classes, since it is predicting whether or not a subject has AD and its corresponding stage. **NOTE:** The model above was trained for binary classification.



(a) Figure 10A: Overall Class Distributions



(b) Figure 10B: Train-Val-Test Split and CI strategy

As shown in Figure 10B (b) above, the original data was randomly sampled and was split as follows: **70%** of the values to the training set, **20 %** of the values to the validation set, and the final **10 %** of values to the testing set. The minority classes were oversampled (with replacement), **only for balancing the training set**, so that the count distributions were the same across all classes. It is important to note balancing the test set could lead to a bias estimation from the performance of the model because the test set should reflect the original data imbalance. The performance metrics on the validation set should be a good approximation of the performance metrics on the test set. Therefore, since balancing the test set is not allowed, the validation set can not be balanced either. Ultimately, the goal of the project is to apply the models in Figure 6 to the real distribution. Therefore the hyper-parameters should be chosen and adjusted accordingly to maximize performance for this imbalanced distribution.

CNN Model Architectures

Convolutional Neural were shown to excel in a wide range of computer vision tasks, particularly image classification tasks with their ability to extract relevant features from image data. A typical CNN consists of two parts (true in general for all convolutional neural network architectures):

1. **Convolutional Base**. This is the part of the network composed by a stack of convolutional and pooling layers. Its purpose is to generate relevant features from the given input image (feature extraction).
2. **Classifier**. This is the part of the network composed by fully connected layers, which are layers whose neurons have full connections to all activation in the previous layers. Its purpose is to classify the input image based on the detected features (image classification).

Figure 11 below helps us visualize this architecture of a model, based on a convolutional neural network. This is a simplified version that fits the purpose of explaining how we will re-purpose a particular pre-trained model. As seen below, the CNN architectures that are explored are much more complex that Figure 11 suggests.

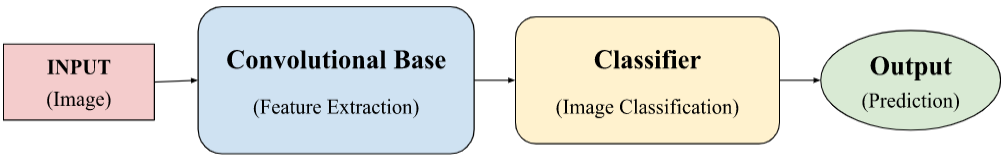


Figure 11: A general model architecture, based on CNN

An important aspect of these deep learning models is that they can automatically learn hierarchical feature representation, which consists of performing feature selection by exploiting dependency relationships among hierarchically structured features. Therefore, hierarchical features correspond to the dependency structure between features.

So, the features computed by the first layer are general, and can be reused for different problem domains, while features computed by the last layer are specific and depend on the chosen data set and classification task. As a result, the convolutional base of our CNN (the layers that are closer to the inputs) refer to general features, while the classifier (and some of the higher layers of the convolutional base) refer to specialized features. This concept will be applied when fine-tuning our models that were loaded with pre-trained from the ImageNet database, as we strive to freeze (non-trainable) as many general feature layers as possible and leave the specialized feature layers as trainable.

This will be expanded on much further in the hyper-parameter optimization and model fine-tuning section below. First, the different CNN model architectures must be introduced. The contribution of this project is to apply these deep learning models to 3D MRI inputs for the classification of Alzheimer's Disease. This research project proposes the use of the following CNN architectures: VGG16, VGG19. DenseNet121, DenseNet201, ResNet50 and ResNet152, to accurately identify Alzheimer's disease at its various stages.

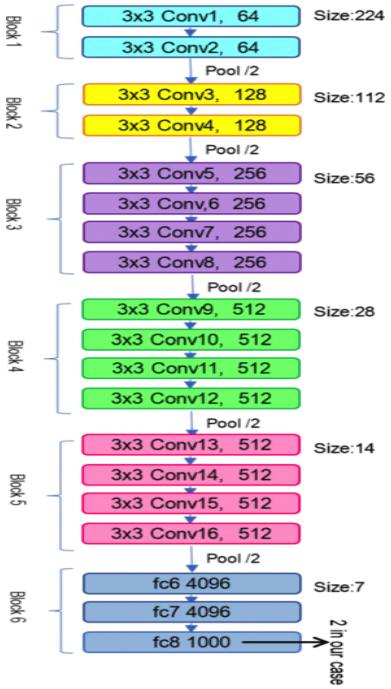
VGG Architecture (VGG16 & VGG19)

The VGG (Visual Geometry Group) architecture was developed and first introduced by Simonyan and Zisserman in their paper "Very Deep Convolutional Networks for Large-Scale Image Recognition." (2014). The VGG architecture is based on a simple and uniform design philosophy. It consists of a series of convolutional layers followed by max-pooling layers. The convolutional layers use small 3x3 filters with a stride of 1 and a padding of 1. These layers are designed to learn a variety of low-level and high-level features from the input image. The architecture also includes max-pooling layers after every two convolutional layers to reduce the spatial size of the feature maps. The max-pooling layers use 2x2 filters with a stride of 2. The max-pooling layers are used to reduce the spatial size of the feature maps, while retaining the most important information. Each convolutional layer is followed by a rectified linear unit (ReLU) activation function, which introduces non-linearity into the model. The ReLU function sets all

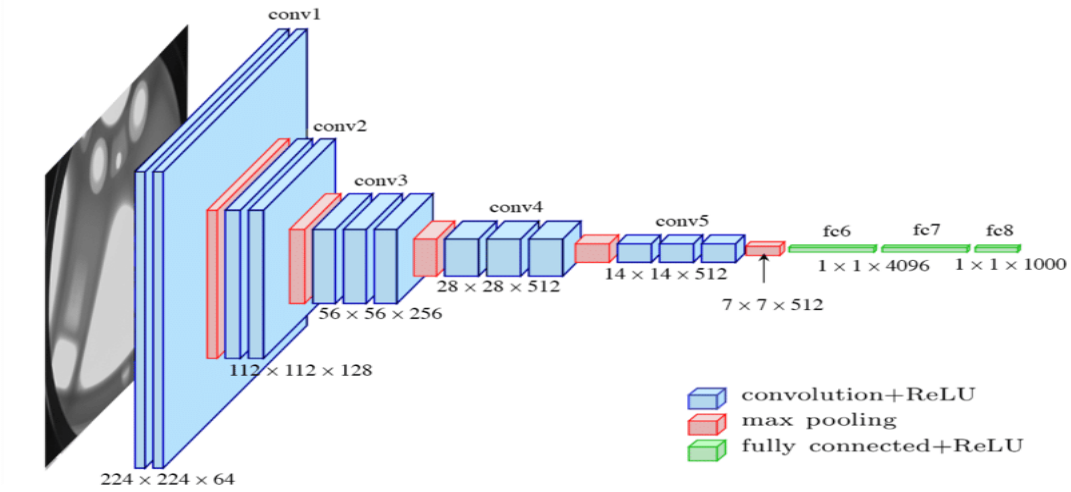
negative values to zero, while keeping positive values the same, helping the model learn non-linear relationships between the input and output. Figure 12 (a-c) illuminate the overall structure of the VGG network.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

(a) Figure 12A: VGG Architecture



(b) Figure 12B: One-way information forward network



(c) Figure 12C: VGG19

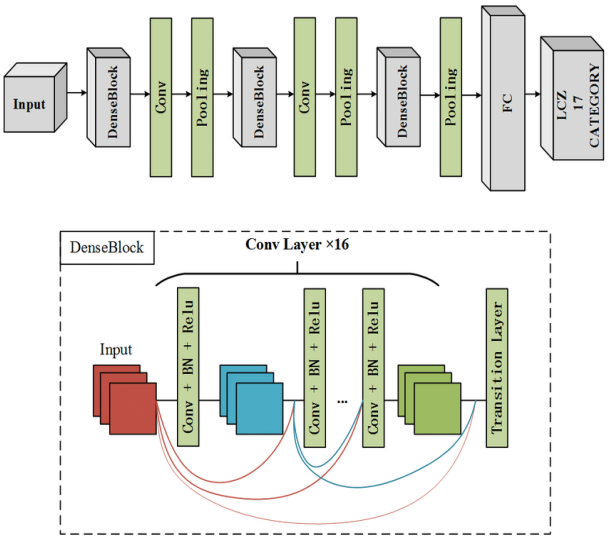
The VGG architecture includes 16 or 19 layers, depending on the version. The VGG16 architecture includes 13 convolutional layers, followed by 3 fully connected layers. The VGG19 architecture includes 16 convolutional layers, followed by 3 fully connected layers. The convolutional layers are grouped into 5 blocks, each with a fixed number of filters. The number of filters increases as we go deeper into the network, allowing the model to learn more complex features. The first two blocks have 2 convolutional layers, while the last three blocks have 3 convolutional layers. The architecture is a traditional CNN architectures that feeds one-way information forward. Both VGG16 and VGG19 have been widely used in computer vision applications, particularly in image classification and object detection. The VGG models are known for their simplicity and effectiveness, and have achieved state-of-the-art results on the ImageNet dataset. These models have become a benchmark for evaluating the performance of other CNN architectures and are an important reference point in the development of new CNN architectures.

DenseNet Architecture (DenseNet121 & DenseNet201)

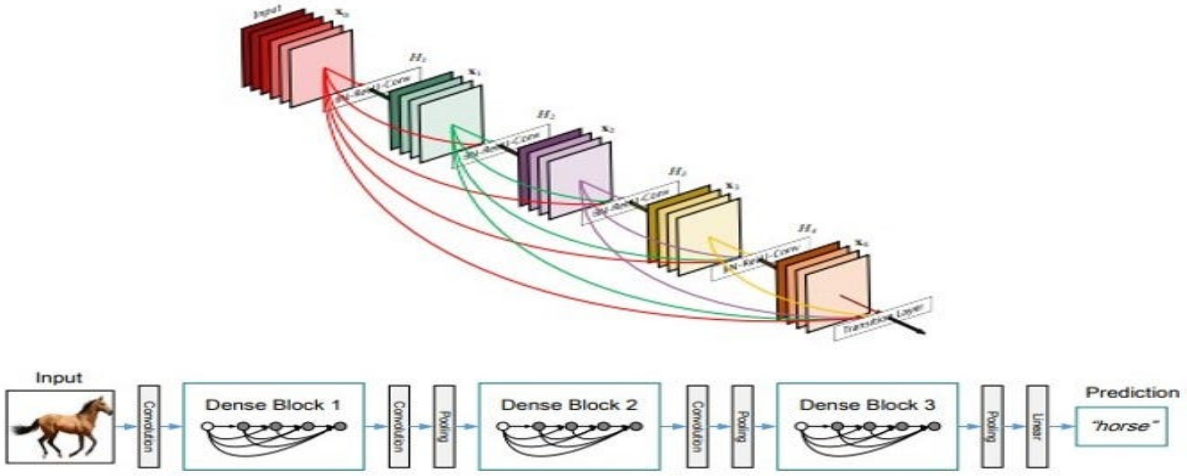
The DenseNet CNN architecture is a deep learning model that was introduced by Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger in their paper "Densely Connected Convolutional Networks." (2017). The DenseNet architecture is based on the idea of concatenating the outputs of each layer instead of adding them. This is done by connecting each layer to every other layer that comes after it, in a feed-forward fashion. This creates dense connections between the layers, which allows the network to reuse features learned at different layers throughout the architecture. This not only reduces the number of parameters in the network, but also improves the flow of gradients during training, leading to faster convergence and better accuracy. This enables the network to learn more robust and discriminative features compared to more traditional CNN architectures (such as the VGG group). Figure 13 (a-c) illuminate the overall structure of the DenseNet network, including dense blocks and layer to layer communication.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201
Convolution	112 × 112		7 × 7 conv, stride 2	
Pooling	56 × 56		3 × 3 max pool, stride 2	
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56		1 × 1 conv	
	28 × 28		2 × 2 average pool, stride 2	
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28		1 × 1 conv	
	14 × 14		2 × 2 average pool, stride 2	
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Transition Layer (3)	14 × 14		1 × 1 conv	
	7 × 7		2 × 2 average pool, stride 2	
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32
Classification Layer	1 × 1		7 × 7 global average pool	
			1000D fully-connected, softmax	

(a) Figure 13A: DenseNet Architecture



(b) Figure 13B: Dense Block



(c) Figure 13C: DenseNet121

The DenseNet architecture is characterized by dense blocks (groups of interconnected layers), which are groups of layers that are connected to each other. Each dense block consists of multiple convolutional layers, followed by a transition layer that combines the output feature maps of all the previous layers. The transition layer is used to reduce the spatial size of the feature maps, while retaining the most important information. This transition layer serves as a bridge between the current dense block and the next one, allowing for a smooth transition between layers. The DenseNet architecture ends with a global average pooling layer, which is used to convert the 4D tensor output

of the final convolutional layer into a 2D tensor. This is followed by a fully connected layer with a softmax/sigmoid activation function at the end of the network to produce the final classification output.

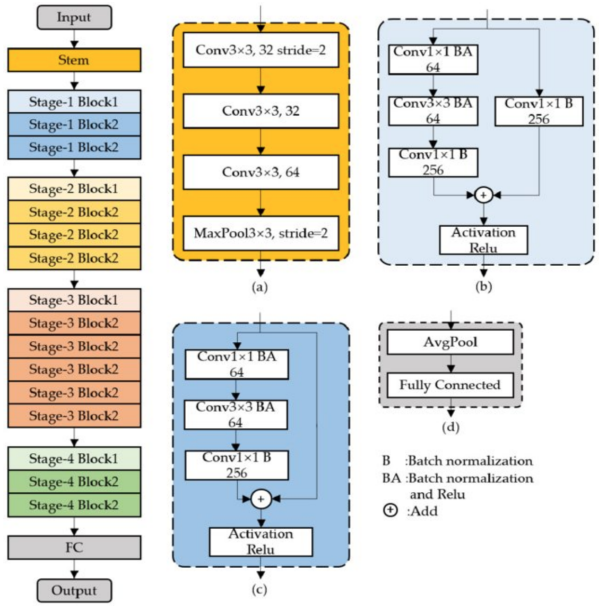
The DenseNet121 and DenseNet201 models are very similar, with the main difference being the number of convolutional layers. DenseNet121 has 121 convolutional layers, and DenseNet 201 has 201 convolutional layers. Both models use the same basic building blocks, consisting of convolutional layers, transition layers, with a pooling layer and fully connected output layer at the end of each network. The main difference between these models is their depth, with DenseNet201 being a deeper architecture than DenseNet121. The DenseNet architecture has been widely used for a variety of computer vision tasks, including image classification, object detection, and segmentation, and has been used as a pre-trained model for transfer learning. Its dense connectivity allows the network to learn more robust and discriminative features compared to traditional CNN architectures.

ResNet Architecture (ResNet50 & ResNet152)

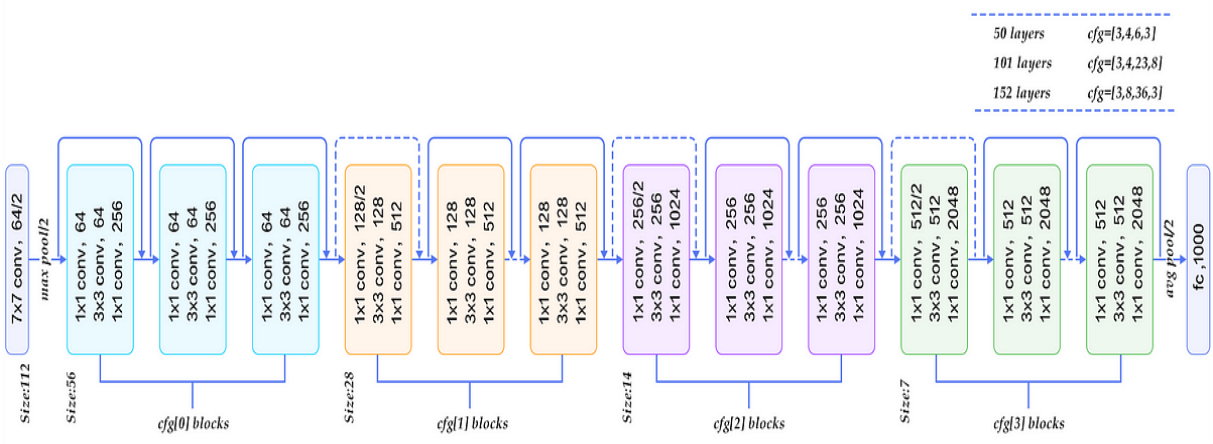
The ResNet (Residual Network) CNN Architecture was first introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. in their paper "Deep residual learning for image recognition." (2016). The ResNet architecture is based on the idea of residual learning, where each layer is connected to the previous layer through a residual block, which allows the network to learn the residual mapping between the input and output. Figure 14 (a-c) illuminate the overall structure of the ResNet network, including residual blocks and layer to layer communication..

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

(a) Figure 14A: ResNet Architecture



(b) Figure 14B: Residual Blocks (skip connections)



(c) Figure 14C: ResNet50

In the ResNet architecture, each layer is connected not only to the previous layer but also to the layer two steps back. The ResNet architecture also uses skip connections, which allow the input to bypass one or more layers and be added directly to the output of a later layer. This approach further improves the flow of information through the network and reduces the vanishing gradient problem. The ResNet50 consists of 50 layers and includes residual blocks with skip connections, while ResNet152 includes bottleneck blocks and multiple residual blocks with skip connections. Both architectures include batch normalization and activation functions after each convolutional layer, as well as global average pooling and fully connected layers for classification. Both models have achieved state-of-the-art results on several benchmark data sets and continue to be widely used in computer vision applications. However, ResNet152 is deeper and more complex than ResNet50, includes multiple residual blocks, with skip connections that bypass several convolutional layers, to create a deeper and more expressive representation of the input image.

It is interesting to note that the VGG models have been used as the basis for popular CNN architectures, such as ResNet and DenseNet. The ResNet and DenseNet networks are similar (both adress the vanishing gradient problem), but have a few differences between them. ResNet adopts summation to connect all preceding feature-maps while DenseNet concatenates all of them. ResNet and DenseNet also differ in how they connect layers in the network. ResNet uses skip connections to directly connect layers, while DenseNet uses dense connections to connect layers. ResNet architectures tend to be deeper and have a more straightforward architecture, while DenseNet architectures tend to be shallower and have a more complex architecture. Refer to Figure 15 below to visualize the comparison.

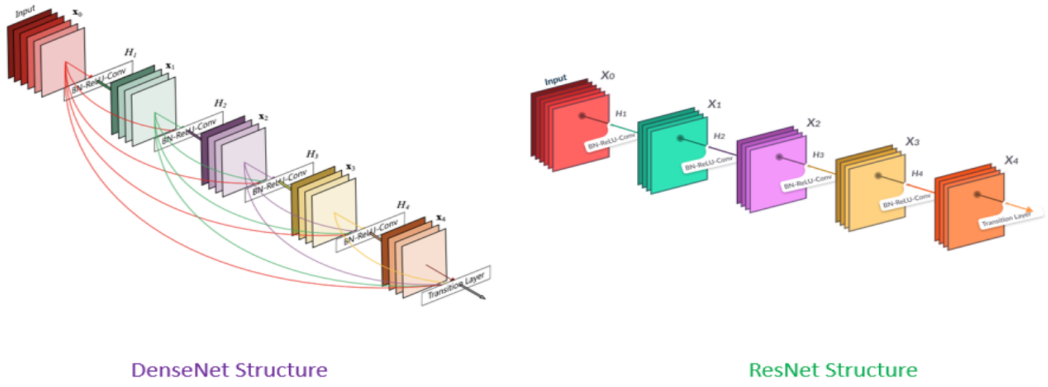


Figure 15: DenseNet vs. ResNet Architectures (information movement)

Transfer Learning (Pre-trained models)

Transfer Learning is a technique where a pre-trained model is used as a starting point for a new model, instead of training that model from scratch. It consists of taking the features learned for one particular problem, and leveraging them onto a new problem, and is usually often done for classification tasks on small data sets. The VGG16 network is used as an example here, but the procedure will be applied to all the architectures described above. By following the steps below, transfer learning can help to accelerate the development of these deep learning models and make them more accessible to a wider range of users.

1. Load the pre-trained VGG16 model by applying the saved pre-trained weights to the model's layers. The model was previously trained on 2D images from the *ImageNet* database. These 2D weights are converted to 3D fit a 3D CNN model and are loaded in for the convolutional base of the VGG16 model. The pre-trained layers will convolve the input 3D MRI image data, and extract features according to *ImageNet* weights.

2. Build and bootstrap a custom fully connected layer (i.e classifier; "top" layer) to the VGG16 model. The pre-trained model was trained on the *ImageNet* image net database, with an output layer with 1000 nodes predicting probabilities for the 1000 classes. Our bootstrapped fully-connected layer will be used to generate predictions for the classification of AD. So, instead the custom classifier layer will have an output layer with 1, 2 or 3 nodes with a linear, sigmoid or softmax activation function respectively. This is because we are training this model to perform either regression, binary classification, a multi-class classification. Refer to Figure 16.

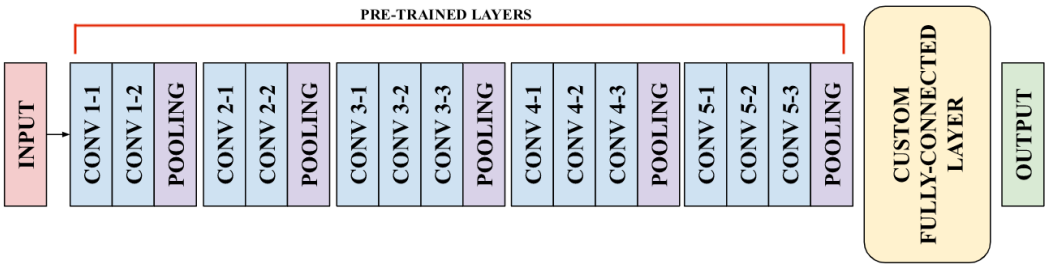
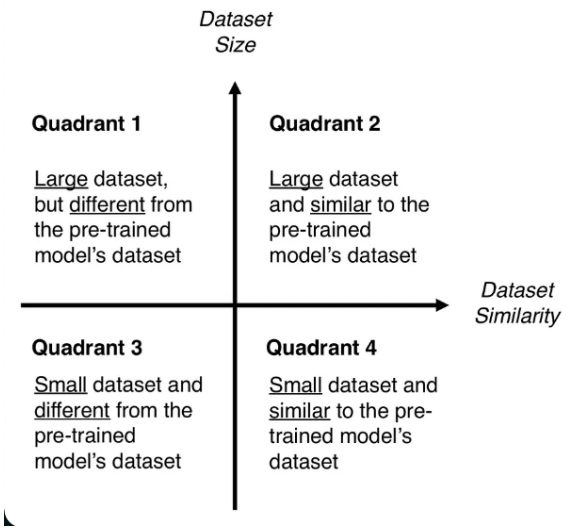


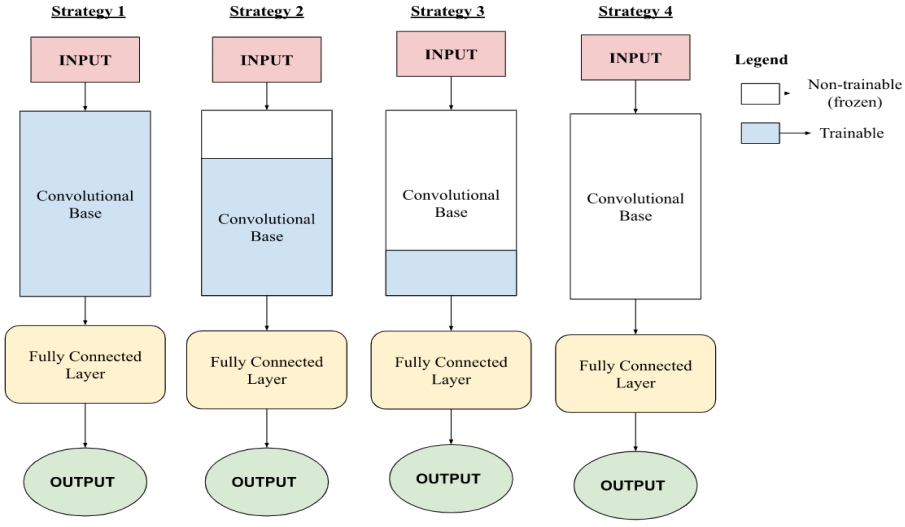
Figure 16: Pre-trained VGG16 architecture with a customized Classifier (Steps 1 and 2)

3. **Fine-tune** the VGG16 model. Fine-tuning can help the model to better adapt to the specific task at hand and can lead to further improvements in performance. This is the last step, and often the most difficult one to perform when dealing with pre-trained models, especially in the context of this project. This is because the data set for which we have the pre-trained weights for (*ImageNet*) is **not similar** at all to the data set we will be utilizing in this project (the OASIS-1 3D MRI scans).**Refer to Figure 17A (a) and 17B (b) below.** We must classify our problem accordingly and using one of the following strategies :

   - Strategy 1: Train the entire model (TFS). None of the pre-trained layers are frozen, and only the architecture of the pre-trained model is used to train on our dataset (OASIS-1). Do this when the size-similarity is in Quadrant 1 (large data set, but different from the pre-trained model's data).
   - **Strategy 2: Freeze a few of the lower layers (near the input) and leave a majority of the layers as trainable. Do this when the size-similarity is in Quadrant 3 (small data set, but different from the pre-trained model's data) It encapsulates the least optimal situation.**
   - Strategy 3: Freeze a majority of the layers, and leave a minorty of the upper layers (near the output) as trainable. Do this when the size-similarity is in Quadrant 2 (large data set and it is similar to the pre-trained model's data).
   - Strategy 4: Freeze the entire convolutional base, setting no trainable layers. Do this when the size-similarity is in Quadrant 4 (small data set and it is similar to the pre-trained model's data).



(a) Figure 17A: Size-Similarity Matrix



(b) Figure 17B: Fine Tuning Strategy

It is important to freeze the layers of the pre-trained model so that they are non-trainable. This is because the weights in these layers have already been learned on a large dataset and are often considered to be "optimal" for a wide range of tasks. By freezing these layers, we can leverage this knowledge and reduce the amount of training data required to achieve good performance on the new task. Lower layers refer to general features (problem independent), while higher layers refer to specific features (problem dependent). Here, we play with that dichotomy by choosing

how much we want to adjust the weights of the network (a frozen layer does not change during training). Since our OASIS-1 MRI data set is relatively small (434 images), and it is not similar to the *ImageNet* database, Strategy 2 will be implemented for fine-tuning the CNN model architectures. With a small data set and a large number of parameters, more layers must be frozen in order to avoid overfitting. However, if the data set is large and the number of parameters is small, you can improve your model by training more layers to the new task since overfitting is not an issue. It will be hard to find a balance between the number of layers to train and freeze. If the model goes too deep (too many trainable layers) and will overfit, but if the model is too shallow (too many frozen layers) it won't be able to learn anything useful. It is also important to note that this is often done using a smaller learning rate than was used in the initial training phase, as the only minor adjustments are needed for weights in pre-trained layers.

Hyperparameter Optimization

Hyperparameters are variables and structures that are set prior to training the model, such as the learning rate, batch size, number of epochs, optimization algorithm, dropout rate, callbacks, activation functions, loss functions, fully connected layers (nodes), and the depth of the network. Refer to Figure 18 below. As previously discussed, fine-tuning (i.e freezing and un-freezing pre-trained layers) changes the overall depth of the network and is a form of hyperparameter optimization. These hyperparameters can have a significant impact on the performance of the model, and finding the optimal values for them can be the a time-consuming process when developing DL models.

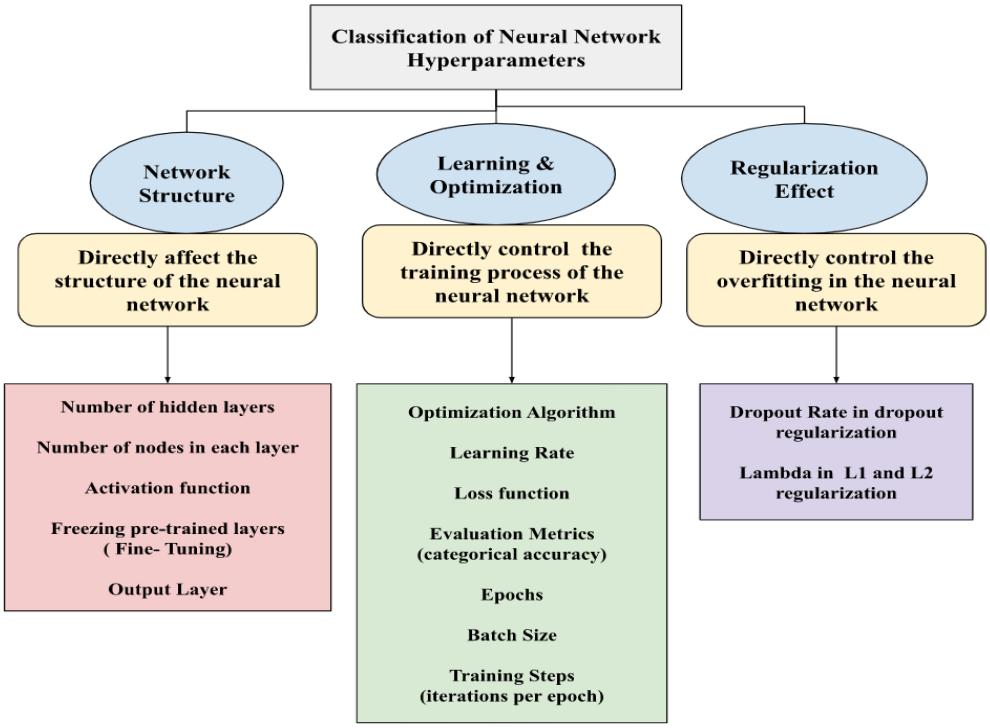


Figure 18: Overview for all types of Hyperparameters

- **Learning Rate (α) :** The learning rate is be one of the most important hyper parameters when configuring the neural network. It controls how much to change the model in response to the estimated error every time the model is updated. Essentially, it determines the step size the optimizer will take when updating the weights of the model during training. A smaller learning rate can help to prevent the weights from diverging, while a larger learning rate can lead to faster convergence but can also lead to unstable training. In order to tune the learning rate while training, this project implements the **ReduceLROnPlateau()** function in the Keras framework, that will adjust the learning rate when a plateau in model performance is detected, e.g. no change for a given number of training epochs. This callback is designed to reduce the learning rate after the model stops improving with the hope of fine-tuning model weights. The function requires you to specify the metric to monitor during training via the *"monitor"* argument (often validation loss), the value that the learning rate will be multiplied by via the *"factor"* argument and the *"patience"* argument that specifies the number of training epochs to wait before changing the learning rate. The initial learning rate was set to: $1e^{-5}$

- **Dropout Rate:** The dropout rate is a key hyperparameter used in deep learning architectures for regularization. It is a technique that randomly "drops out" some of the nodes (sets to 0) in a layer during training, which can help to prevent overfitting and improve generalization performance. The dropout rate determines the probability that a node will be dropped out during training. A high dropout rate can improve generalization performance by making the model more robust to noise, but can also lead to underfitting. A low dropout rate can lead to overfitting and poor generalization performance. The optimal dropout rate depends on the specific task and data set. This project implements a drop out rate of 0.6 (60%) in order to improve the generalization performance of the model. In addition, λ in L1 and L2 regularization is an important hyperparameter that can be used to prevent overfitting in deep learning models (but not implemented in this project).

- **Batch Size and Number of Epochs:** The batch size and epoch parameters are important hyperparameters that are used in deep learning models for training. The batch size refers to the number of samples that are processed in a single forward/backward pass during training, while the epoch is the number of times that the entire training data set is passed through the network during training. A larger batch size can result in faster training times but may lead to poorer generalization performance, while a smaller batch size can lead to better generalization performance but may result in slower training times. Batch size controls the accuracy of the estimate of the error gradient when training neural networks and there is a give and take between batch size and the speed and stability of the learning process. The epoch parameter is certainly not as important as the batch size and other hyperparameters. The total number of epochs can be set very large and the **EarlyStopping()** callback function in the Keras framework can be used during training to prevent overfitting and improve generalization performance. The function requires you to specify the metric to monitor during training via the *"monitor"* argument (validation loss), the behavior that the function will be monitoring via the *"mode"* argument (minimum) and the *"patience"* argument specifies the number of training epochs to wait before stopping training. Total number of training steps was set to $\rightarrow$ (length of training set)/ (batch size).

- **Activation Function:** The activation function is another important hyperparameter in deep learning models that plays a critical role in determining the output of each neuron in the network. The activation function is applied to the output of each layer of the neural network, allowing the model to learn complex non-linear relationships between inputs and outputs. The ReLU function, or Rectified Linear Unit, is a popular choice for activation functions due to its simplicity and effectiveness. The ReLU function maps any input value below zero to zero, and any input value above zero to the input value itself. This allows the network to learn more complex and expressive representations of the data, while also avoiding the vanishing gradient problem. In general, the ReLU function is a good choice for a wide range of deep learning problems. The activation function for the output layer is especially important as it is determined by the type of problem to be solved. A linear activation function is used on the regression models, a sigmoid activation function (one node) on the binary classification models, and finally a softmax activation function (3 nodes) on the multi-class classification problems.

- **Loss Function:** The loss function is another important hyperparameter in deep learning models that measures the difference between the predicted output and the actual output. The goal of the loss function is to minimize this difference, or error, over the entire training data set.There are several popular loss functions available, including mean squared error (MSE), binary cross-entropy, and categorical cross-entropy. The choice of loss function depends on the specific task and dataset at hand. This project implements various loss functions depending on the task. It uses a concordance correlation (ccc) and mean squared error (mse) for the regression models, binary crossentropy for the binary classification models, and categorical crossentropy for the multi-class classification models. The choice of loss function can have a significant impact on the performance of the model, as it is a key part in the learning rate reduction and early stopping monitor callbacks. It is important to note that the data must be one-hot encoded when using binary or categorical cross entropy loss functions.

- **Optimization Algorithm:** The optimization algorithm (Optimizer) is an important hyperparameter in deep learning models that affects the speed and quality of the learning process. The optimizer function controls how the weights of the network are updated during training, by adjusting the learning rate based on the gradients of the loss function. They allows for an adaptive learning rate. ADAM stands for Adaptive Moment Estimation and it uses a combination of gradient descent with momentum and adaptive learning rates, making it well-suited for non-convex optimization problems such as training deep neural networks. RmsPROP, or Root Mean Square Propagation, is another popular optimizer that adapts the learning rate of each weight based on the average of the squares of the past gradients for that weight. However, in general, ADAM is known for its good performance on a wide range of deep learning tasks, and is what this project implements.

Results

During the training of the models for the classification of Alzheimer's disease, the training and validation accuracy and loss were recorded and displayed below. The purpose of displaying these metrics is to monitor the performance of the models during the training process and to identify any potential overfitting issues. Overall, the models showed promising results, with both training and validation accuracy increasing with each epoch. However, some of the models slightly overfitted the data, which means they performed well on the training data but did not generalize well to new data. Despite this, the hyperparameter optimization process seemed to work well in improving the overall performance of the models. Fine-tuning was performed by freezing the first 1/3 of the pre-trained layers (slightly different for different models), which were already trained on the ImageNet dataset, and then training the rest of the layers on the new dataset. By doing this, the model was able to update the weights on the new dataset while retaining the previously learned generalized features from ImageNet. Performance summaries are shown below.

NOTE: A random_state parameter was set to randomly shuffle the data set before the split into the training, validation and testing sets. This was to ensure all the versions of models performing a similar task (regression, binary classification or MC classification) were trained, validated and tested on the same data (for fair comparison).

Performance Metrics

A true positive **(TP)** is an outcome where the model correctly predicts the positive class. A true negative **(TN)** is an outcome where the model correctly predicts the negative class. A false positive **(FP)** is an outcome where the model incorrectly predicts the positive class. A false negative **(FN)** is an outcome where the model incorrectly predicts the negative class. The positive class is Class 1/2 (Diagnosed Alzheimer's) and the negative is Class 0 (Healthy).
The confusion matrices allow us to measure precision (positive predicted value), recall (sensitivity), and specificity which, along with accuracy, F1 score and AUC, are the metrics used to measure the performance of these models. Precision measures the accuracy of positive predictions, while recall measures the completeness of positive predictions. Sensitivity (true positive rate) refers to a test's ability to designate an individual with disease as positive. A highly sensitive test means that there are few false negative results, and thus fewer cases of disease are missed. The specificity (true negative rate) of a test is its ability to designate an individual who does not have a disease as negative.

$$Precision = \frac{TP}{TP + FP} \qquad Recall(Sensitivity) = \frac{TP}{TP + FN} \qquad Specificity = \frac{TN}{TN + FP}$$

Accuracy represents the number of correctly classified data instances over the total number of data instances. However, accuracy may not be a good measure if the data set is not balanced (both negative and positive classes have different number of data instances). F1-score is a harmonic mean (weighted average) of precision and recall, and so it gives a combined idea about these two metrics. It is maximized when precision is equal to recall. In both precision and recall, there is false positive and false negative, so it also considers both. **F1 score is more useful than accuracy, especially when there is an uneven class distribution.** However, the interpretability of the F1-score is poor. This means that we don't know what our classifier is maximizing – precision or recall. So, we use it in combination with other evaluation metrics which gives us a complete picture of the result.

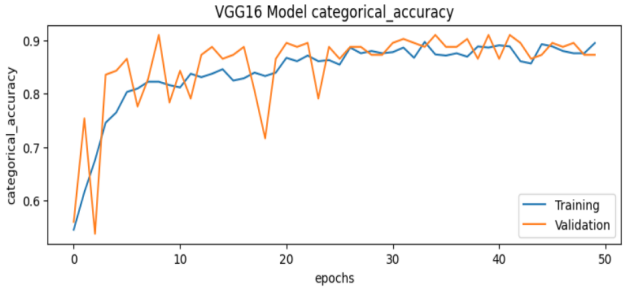
$$AUC = \frac{1}{2} - \frac{FPR}{2} + \frac{TPR}{2} \qquad Accuracy = \frac{TN + TP}{TN + FP + TP + FN} \qquad F1\_score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

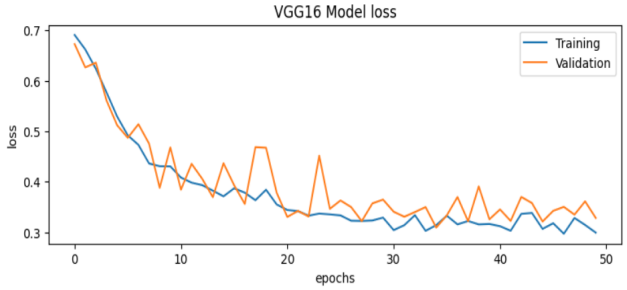
AUC (Area Under the Curve) is also preferred over Accuracy as it is a much better indicator of model performance. This is because AUC uses the True Positive Rate and False Positive Rate of the model across different cut-off thresholds, and if you are using the Accuracy metric, it is advised to use other metrics as well. Recall that a model with an AUC score of 0.5 is no better than a model that performs random guessing.

This method is slightly different for the multi-class classification, since there are no inherent (binary) positive or negative classes. So instead, the TP, TN, FP and FN are calculated for each individual class and summed to find the overall TP, TN, FP, and FN and precision, recall and F1 score are calculated using the methods above, along with micro or macro formulations. Micro scoring would be done by calculating the metrics globally (sum of TPs, sum of FPs, etc..). Macro scoring would done by calculating the metrics for each class and averaging across them. This project calculates the performance metrics using the macro scoring technique for the multi-class models.

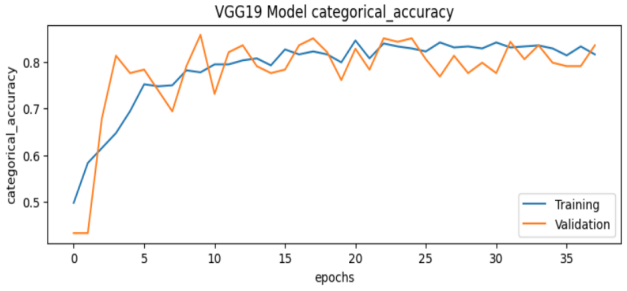
Binary Classification

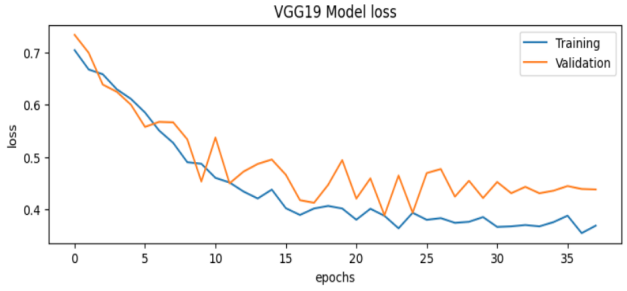
The training and validation accuracy and loss plots were generated for each of the six binary classification models performing to visualize their performance during the training process. From the series of plots above, we can see most of the models were able to fit the data relatively well. The hyperparameter optimization was successful allowing the models to properly fit the data. Refer to Figure 19 (a-f) below.



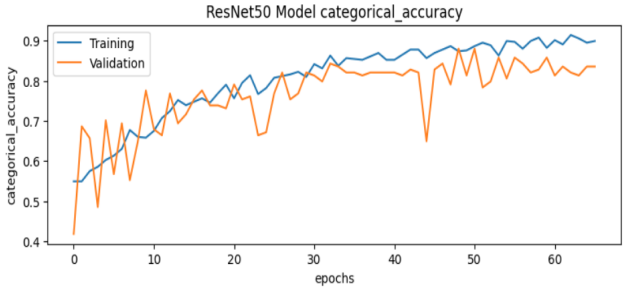


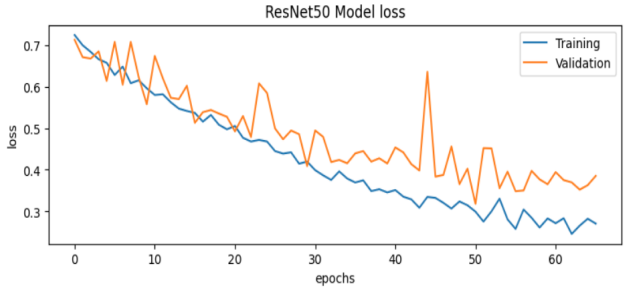
(a) Figure 19 A: **VGG16**: Accuracy and Loss for Training and Validation (Binary Classification)



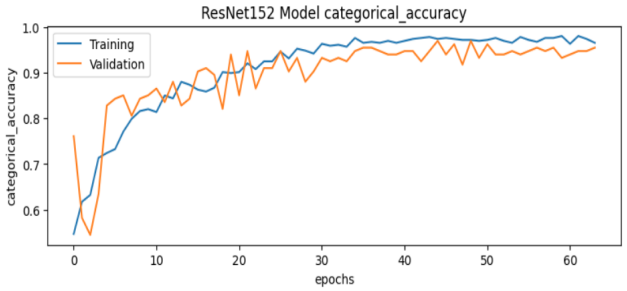


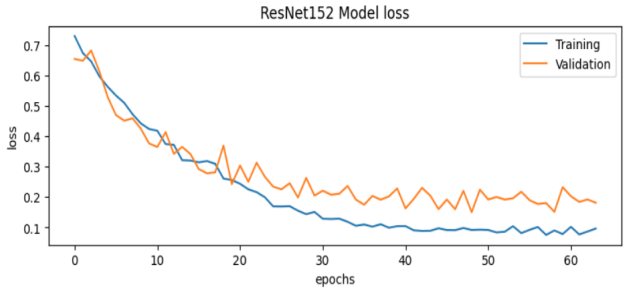
(b) Figure 19 B: **VGG19**: Accuracy and Loss for Training and Validation (Binary Classification)



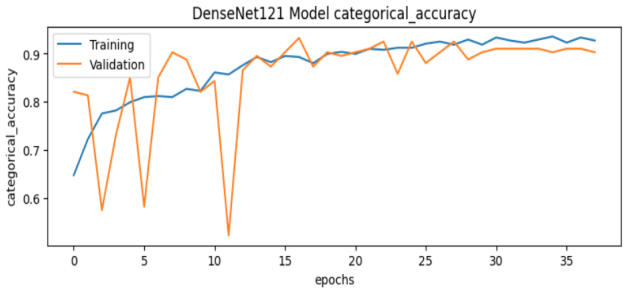


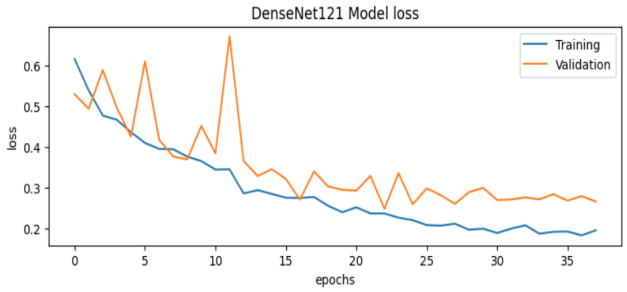
(c) Figure 19 C: **ResNet50**: Accuracy and Loss for Training and Validation (Binary Classification)



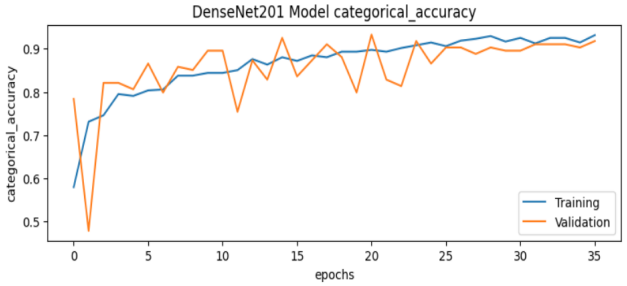


(d) Figure 19 D: **ResNet152**: Accuracy and Loss for Training and Validation (Binary Classification)





(e) Figure 19 E: **DenseNet121**: Accuracy and Loss for Training and Validation (Binary Classification)



(f) Figure 19 F: **DenseNet201**: Accuracy and Loss for Training and Validation (Binary Classification)

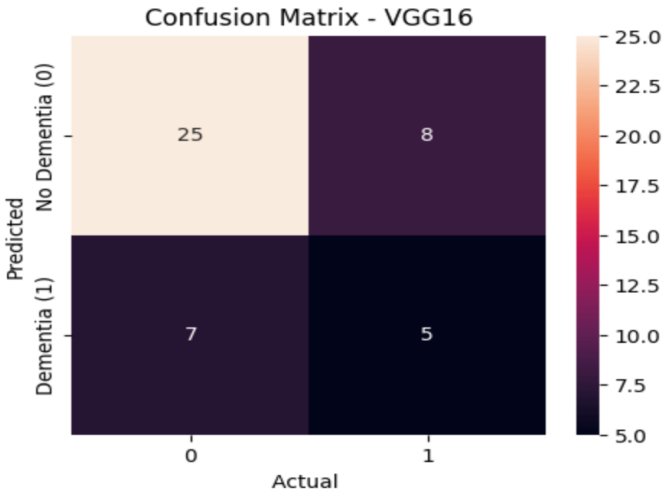
According to the obtained results below, the ResNet152 model outperformed the other models in binary classifying AD by achieving the highest F1-score and sensitivity. The VGG architectures took significantly longer to train and did not seem to perform well on the test data. This could be due to the deep structure of the VGG models, which can lead to overfitting and gradient vanishing/exploding problems. On the other hand, the ResNet and DenseNet models have shortcut connections, which can mitigate these issues by facilitating gradient flow and allowing information to skip layers. DenseNet and ResNet have different numbers of layers and different levels of connectivity, which can affect their performance. DenseNet121 and DenseNet201 have similar architectures with similar numbers of layers, which may explain why they had similar performance on the task of binary classifying AD. DenseNet121 and DenseNet201 have similar architectures with similar numbers of layers, which may explain why they had similar performance on the task of binary classifying AD. On the other hand, ResNet152 and ResNet50 have very different numbers of layers and levels of connectivity. ResNet152 has significantly more layers than ResNet50, which may explain why it much performed better on the task. Additionally, ResNet152 has been shown to perform better than ResNet50 on various computer vision tasks due to its increased depth and complexity. Therefore, the significant difference in performance between ResNet152 and ResNet50 may be attributed to the differences in their architectures.

Model	Process	F1-score	Precision	Sensitivity/Recall	AUC	Accuracy
VGG16	Binary Classification	40.00%	38.46%	41.67%	0.587	66.67%
VGG19	Binary Classification	48.00%	46.15%	50%	0.644	71.11%
ResNet50	Binary Classification	58.33%	58.33%	58.33%	0.716	77.78%
⋆ **ResNet152**	Binary Classification	⋆ **80.00%**	⋆ **76.92%**	⋆ **83.33%**	⋆ **0.871**	88.89%
DenseNet121	Binary Classification	66.67%	60.00%	75.00%	0.784	80.00%
DenseNet201	Binary Classification	72.00%	69.23%	75.00%	0.814	84.44%

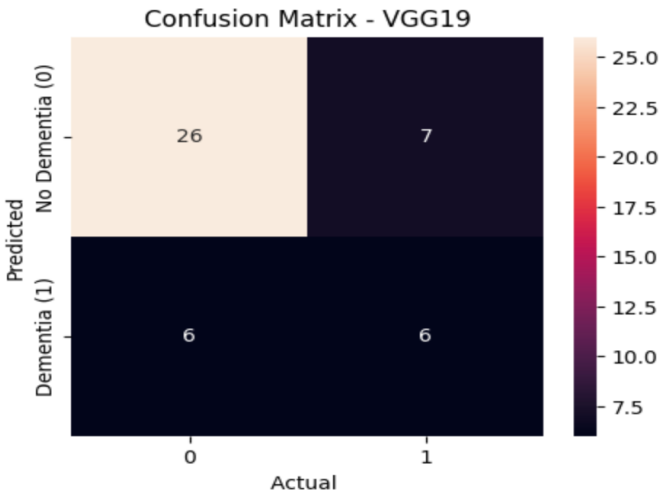
Table 1 : Binary Classification Comparison of Performance Metric for the transfer learning models

The performance metrics above were calculated using the formulas in the Results section and the confusion matrix heat maps generated by each of the six different models . Refer to Figure 20 (a-f) below.

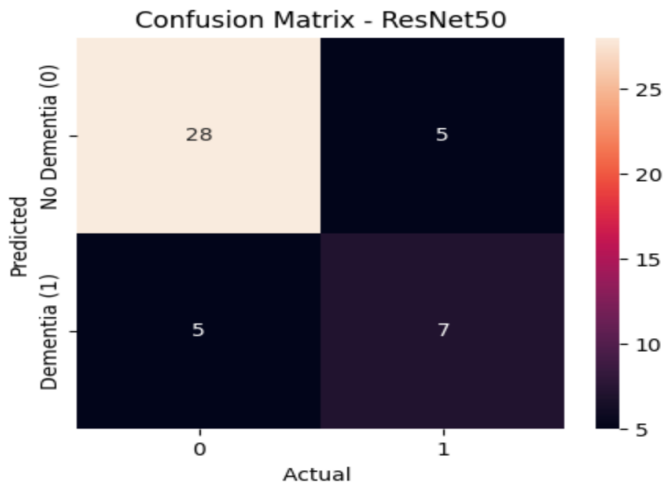
Test Set Distribution: **33 Healthy (Class 0)** **12 Alzheimer's Disease (Class 1)**



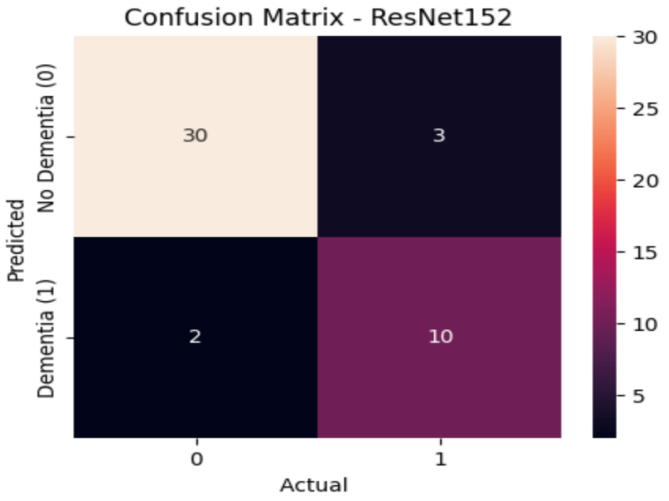
(a) Figure 20 A: **VGG16**



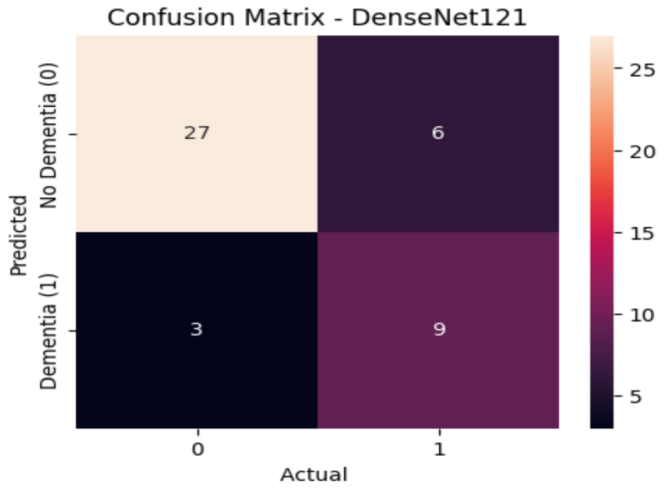
(b) Figure 20 B: **VGG19**



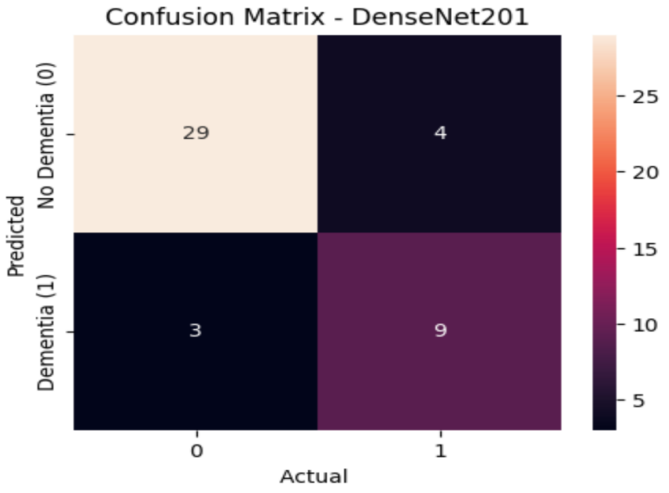
(c) Figure 20 C: **ResNet50**



(d) Figure 20 D: **ResNet152**



(e) Figure 20 E: **DenseNet121**



(f) Figure 20 F: **DenseNet201**

Multi-Class Classification

The training and validation accuracy and loss plots were generated for each of the six binary classification models performing to visualize their performance during the training process. From the series of plots above, we can see most of the models were able to fit the data well (better than the binary models). The hyperparameter optimization was successful allowing the models to properly fit the data. Refer to Figure 21 (a-f) below.

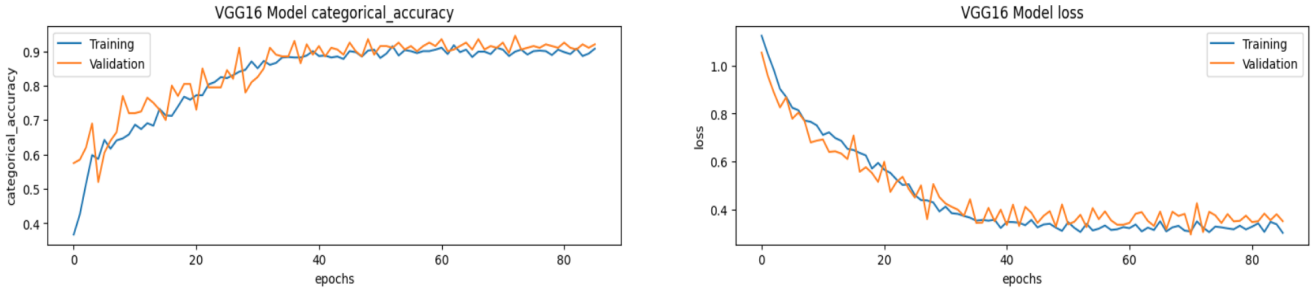


Figure 21 A: **VGG16**: Accuracy and Loss for Training and Validation (Multi-Class Classification)

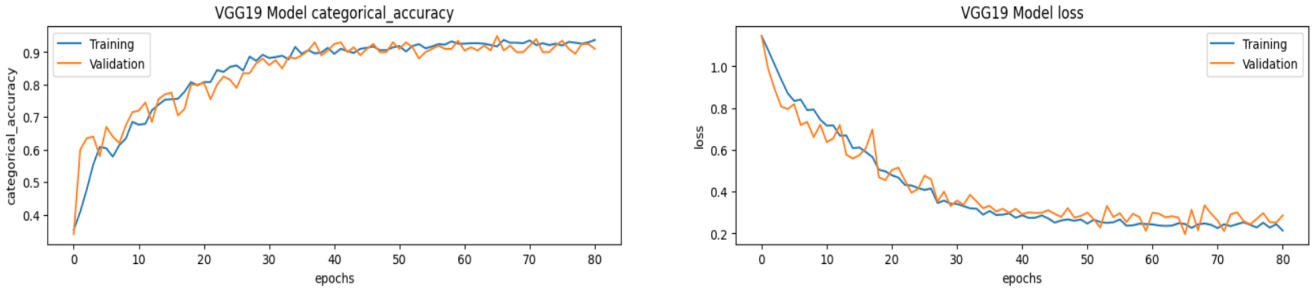


Figure 21 B: **VGG19**: Accuracy and Loss for Training and Validation (Multi-Class Classification)

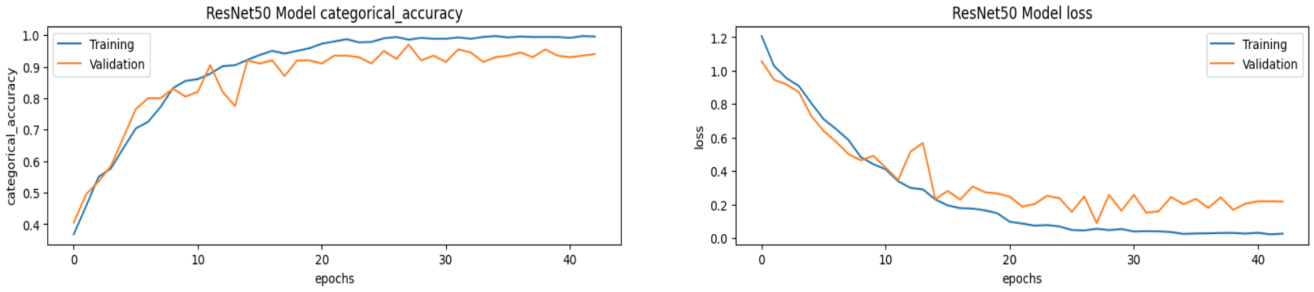


Figure 21 C: **ResNet50**: Accuracy and Loss for Training and Validation (Multi-Class Classification)

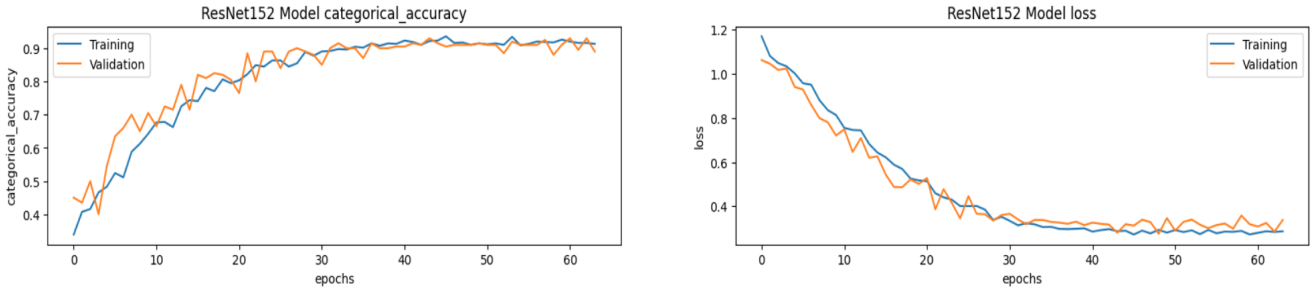
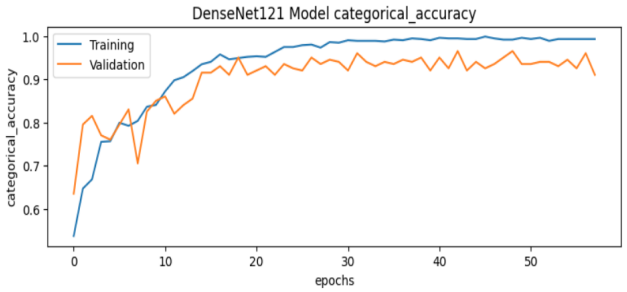


Figure 21 D: **ResNet152**: Accuracy and Loss for Training and Validation (Multi-Class Classification)



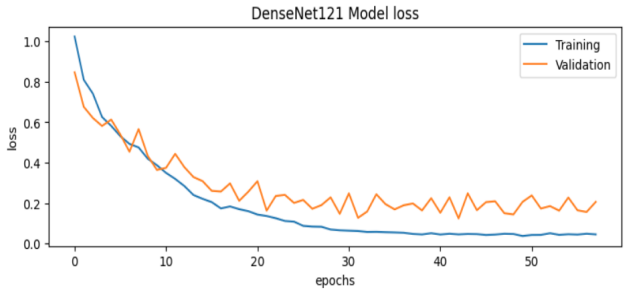


Figure 21 E: **DenseNet121**: Accuracy and Loss for Training and Validation (Multi-Class Classification)

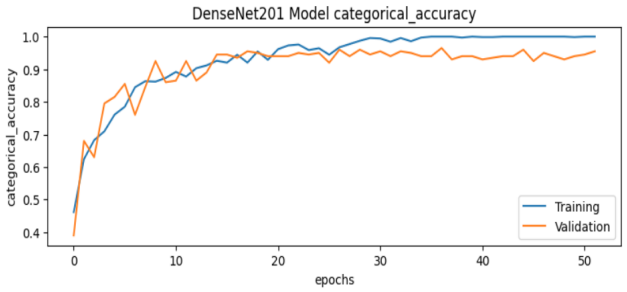


Figure 21 F: **DenseNet201**: Accuracy and Loss for Training and Validation (Multi-Class Classification)

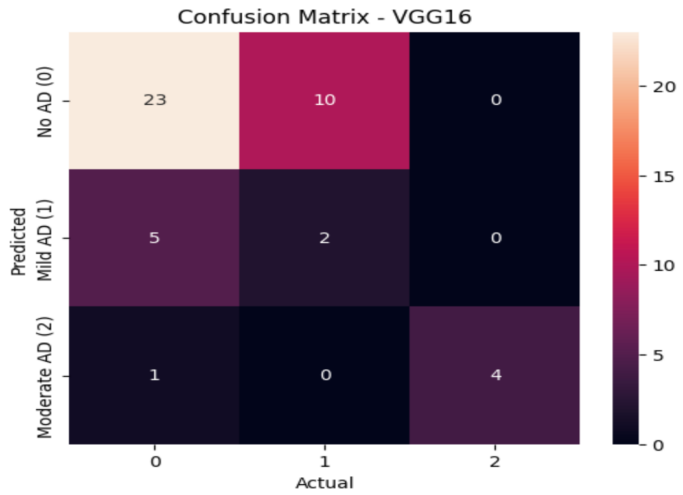
Model performance in terms of precision, recall, F1-score AUC, and accuracy metrics improved for all the models for multi-class classification compared to binary classification. This is not surprising since the multi-class classification task has more classes, which increases the complexity of the classification problem. Also, predicting the severity of Alzheimer's disease is a more clinically relevant task compared to a simple binary classification. In addition, the differences in performance between the models for binary classification were more apparent, while for multi-class classification, the models had similar performance. This could be because the multi-class classification task has a larger number of samples (training data adjusted data for each class), and therefore, the models had more data to learn from. Similar to the binary classification results, the VGG architectures under performed, and the Dense121 and Dense201 had similar performances. The results suggest that the ResNet152 model is the best fit for this classification task due to its high F1-score and sensitivity. A high sensitivity value indicates that the model is able to correctly identify a maximum number of patients with the disease, minimizing the risk of false negatives, which can be detrimental to the patient's health outcomes. Therefore, the superior performance of ResNet152 is an important finding, which may have practical implications for the diagnosis and treatment of AD.

Model	Process	F1-score	Precision	Sensitivity/Recall	AUC	Accuracy
VGG16	Multi-class Classification	61.38%	65.33%	59.42%	0.67	64.44%
VGG19	Multi-class Classification	67.12%	70.2%	66.2%	0.726	71.11%
ResNet50	Multi-class Classification	71.46%	70.51%	72.99%	0.783	77.78%
⋆ **ResNet152**	Multi-class Classification	⋆ **88.22%**	⋆ **85.03%**	⋆ **92.21%**	⋆ **0.934**	91.11%
DenseNet121	Multi-class Classification	80.77%	77.4%	85.43%	0.878	84.44%
DenseNet201	Multi-class Classification	82.73%	79.79%	86.44%	0.887	86.67%

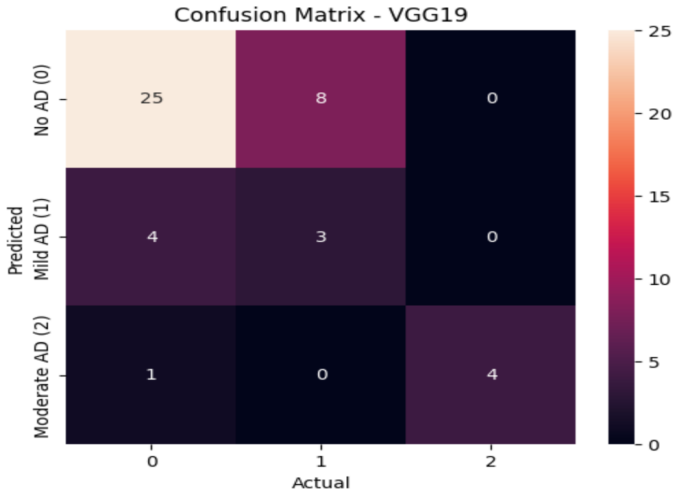
Table 2 : Multi-Class Classification Comparison of Performance Metric for the transfer learning models

The performance metrics above were calculated using the formulas in the Results section and the confusion matrix heat maps generated by each of the six different models The F1-score, recall, precision and accuracy metrics were calculated for each individual class, averaged across, and displayed in Table 2. Refer to Figure 22 (a-f) below.

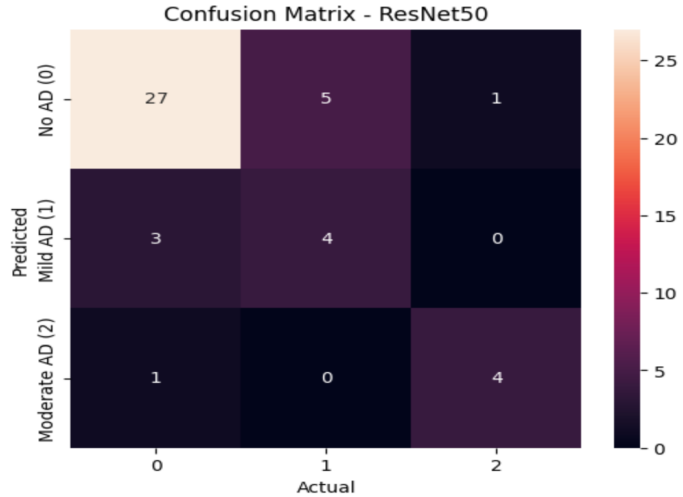
Test Set Distribution: **33 Healthy (Class 0)** **7 Mild AD (Class 1)** **5 Moderate/Severe AD (Class 2)**



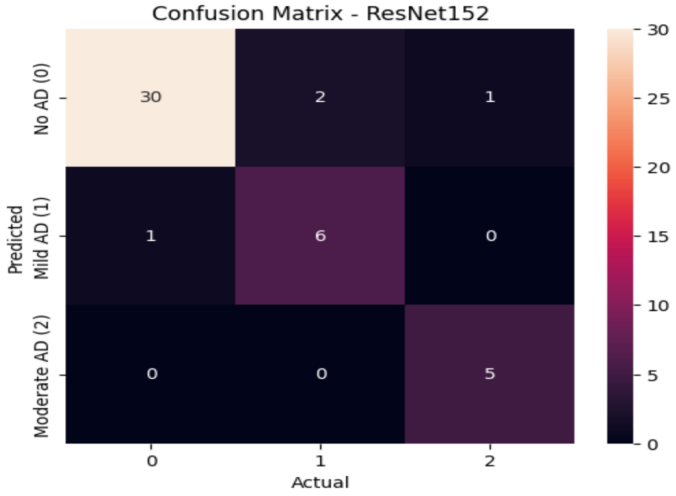
(g) Figure 22 A: **VGG16**



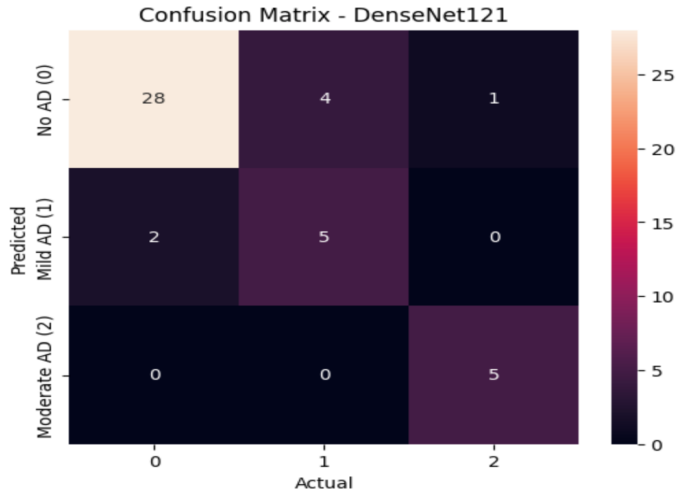
(h) Figure 22 B: **VGG19**



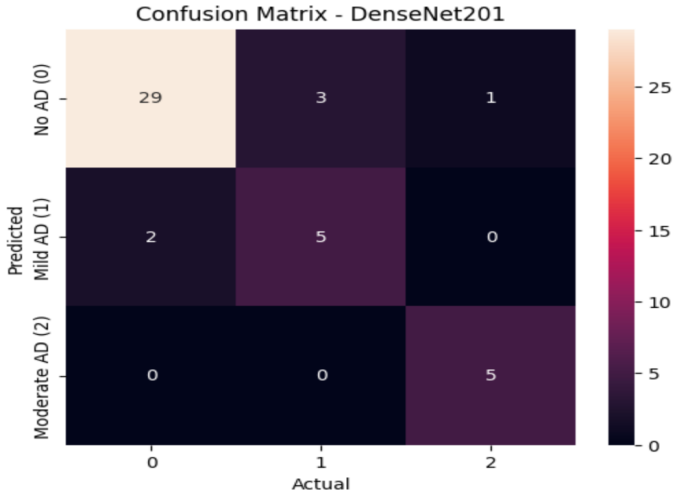
(i) Figure 22 C: **ResNet50**



(j) Figure 22 D: **ResNet152**



(k) Figure 22 E: **DenseNet121**



(l) Figure 22 F: **DenseNet201**

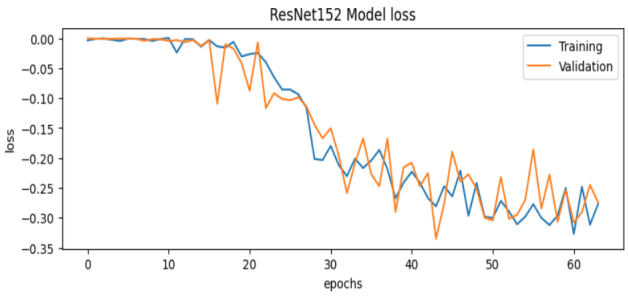
Regression

For the regression analysis, the values for the class rating were re-categorized as: Class 0 = 0.0, Class 1 = 0.5, and Class 2 = 1.0. This allows a method of classifying using ordinal values, as opposed to mutually exclusive classes. This would provide a more comprehensive understanding of a patients classification of AD. If the model classified a subject as healthy, but the value returned was 0.3, there would be an indication of MCI, possibly AD. The Concordance correlation coefficient metric is implemented as the loss function for the regression models, as an alternative to MSE. Unlike MSE, CCC measures not only the similarity between predicted and actual values but also the degree to which they are proportionally related. This is important when dealing with data that has high inter-subject variability or measurement errors, such as medical imaging data. Additionally, CCC has been shown to be a more robust measure than MSE for assessing the agreement between predicted and actual values in medical imaging applications, making it a useful metric for evaluating regression models in this domain. The CCC is returned as a negative value in order to allow the model to minimize the loss function (i,e. maximize the metric).

NOTE: I did not train the VGG architectures for the regression task. VGG architectures are not suitable for regression problems as they have a very deep structure and consist of a large number of parameters. They are one-way information networks and due to the high number of layers, the VGG models are prone to overfitting, which can cause poor generalization to new data. This makes them less suited for regression tasks. ResNet and DenseNet are better suited for regression tasks due to their architectures (residual and skip connections), which allows them to learn more effectively and generalize better to new data. Refer to Figure 23 (a-d).

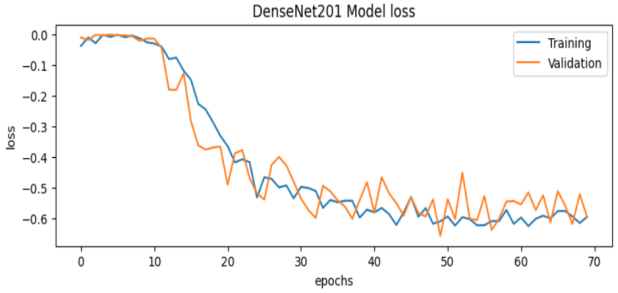
The performance metrics were calculated below by manually classifying the output of the regression model for the 3 classes as follows: if $output < 0.35 \rightarrow$ **Class 0**; if $0.35 \leq output \leq 0.65 \rightarrow$ **Class 1**; if $output > 0.65 \rightarrow$ **Class 2**.

(m) Figure 23 A: **ResNet50**



(n) Figure 23 B: **ResNet152**

(o) Figure 23 C: **DenseNet121**



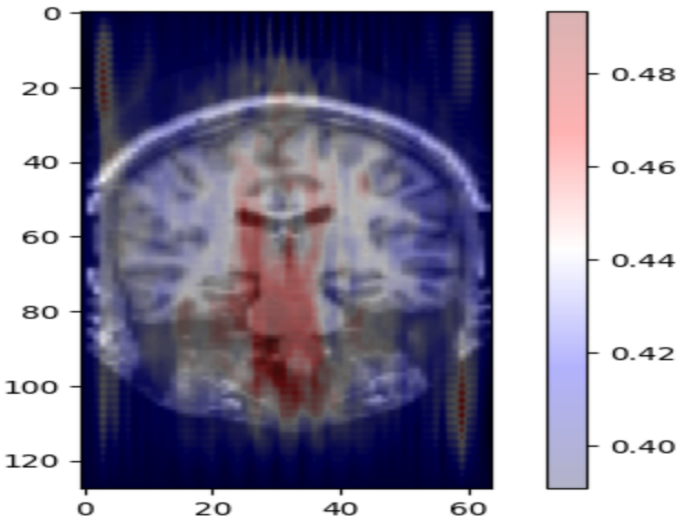
(p) Figure 23 D: **DenseNet201**

Model	Process	F1-score	Precision	Sensitivity/Recall	AUC	Accuracy
ResNet50	Regression	44.81%	44.47%	46.98%	0.604	57.78%
⋆ **ResNet152**	Regression	⋆ **67.24%**	⋆ **65.67%**	⋆ **69.23%**	⋆ **0.765**	77.77%
DenseNet121	Regression	54.3%	54.44%	54.77%	0.661	66.67%
DenseNet201	Regression	62.11%	64.03%	61.56%	0.717	73.73%

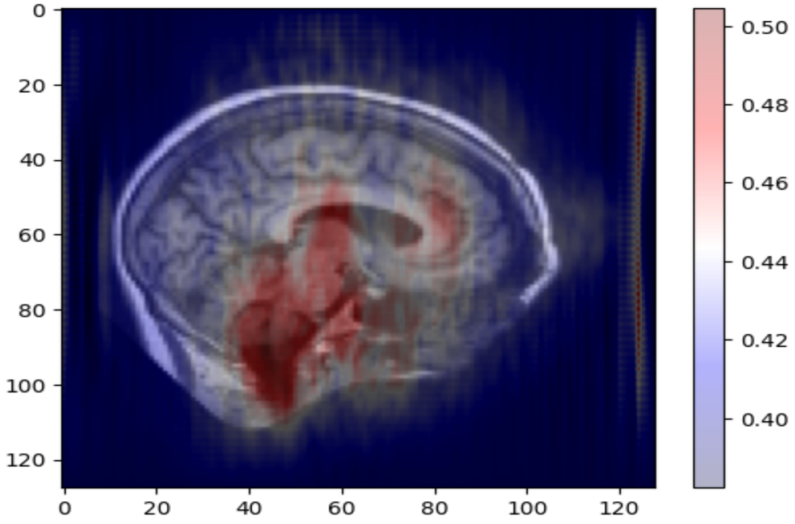
Table 3 : Regression Comparison of Performance Metric for the transfer learning models

Saliency Maps

Saliency maps are an important tool for interpreting deep learning models. They provide a way to visualize which parts of an input image are most important for a particular output classification. Saliency maps are generated by computing the gradient of the output classification with respect to the input image. This gradient represents the contribution of each pixel in the image to the output classification. By visualizing this gradient, it is possible to identify the pixels that have the strongest influence on the classification decision. The saliency map is typically generated by computing the gradient of the output classification with respect to the input image and then normalizing the gradient to produce a heat map. The averaged heat map is then overlaid on the input image for visualization.

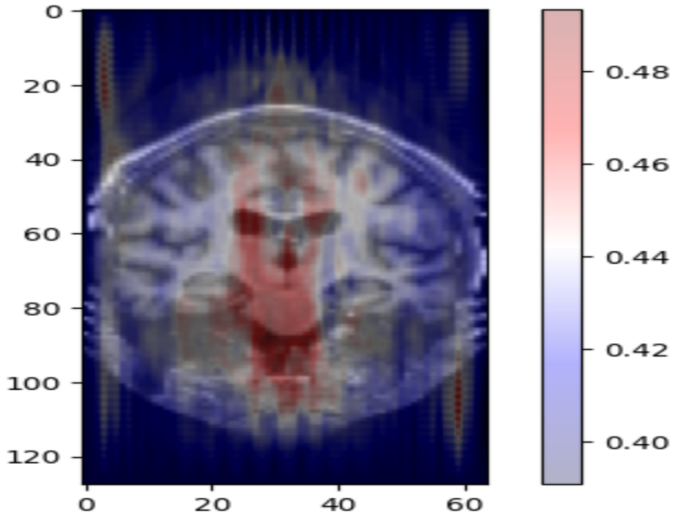


(q) Figure 24 A: **Class 0**: Coronal View

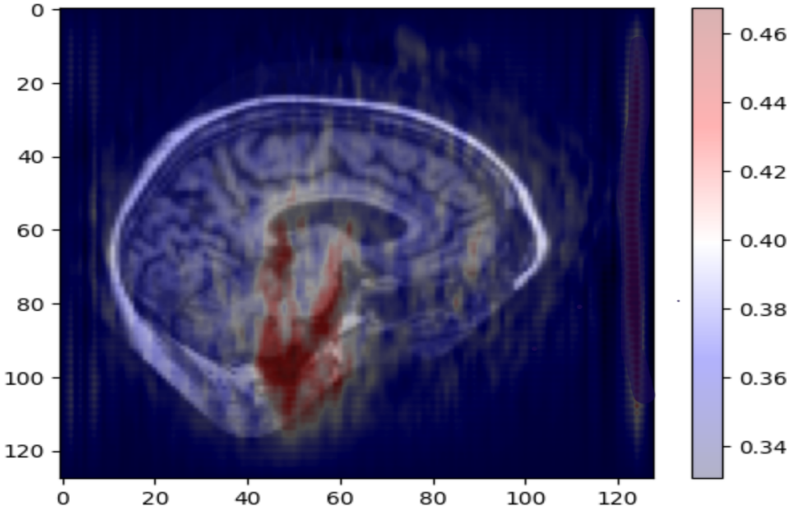


(r) Figure 24 B: **Class 0**: Sagittal View

Saliency maps were generated for the top-performing model, ResNet152, by capturing a variety of axial, coronal, and sagittal view slices of the brain. However, the axial slices provided little to no interpretability for the model, so they were left out. The heat maps were calculated for all images in the test data set, averaged together and then displayed in Figure 24 (a-f). Despite the non-informative nature of the axial slices, the generated saliency maps for the remaining views proved to be quite useful in identifying the regions of the brain that were significant for classification. By visualizing the areas of the brain that the model is attending to, we can gain insight into what aspects of the brain are important for predicting Alzheimer's disease.

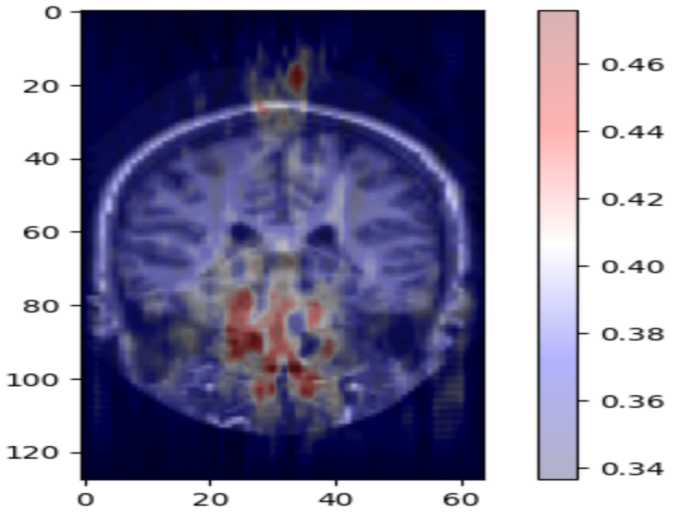


(s) Figure 24 C: **Class 1**: Coronal View

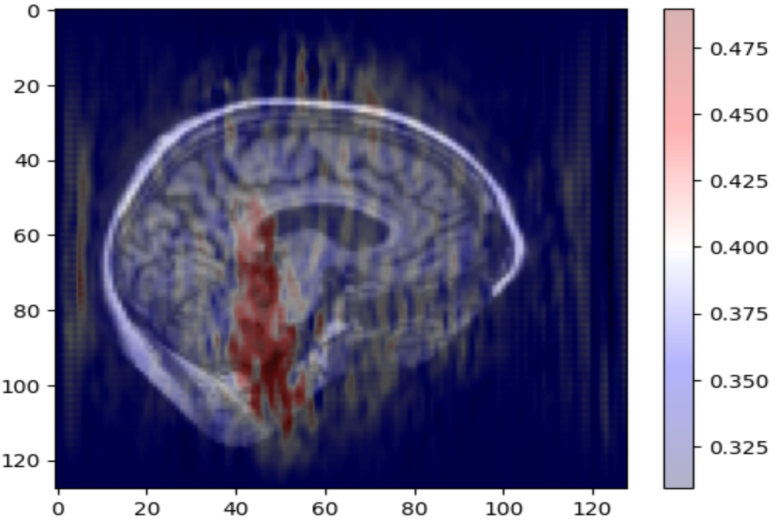


(t) Figure 24 D: **Class 1**: Sagittal View

It is clear to see that the model is focusing on (red) areas of the brain associated with AD, such as the medial temporal lobe, and ventricles (discussed in the problem introduction). Looking the coronal slices, the model seems to be focused on the lateral ventricles, the fluid filled spaces in the brain, as well the middle portion of the slice, which is where the hippocampus and amygdala (MTL) are located. The blue areas surrounding the image seem to be looking at the structure of the of the cerebral cortex, which is a clear indicator of the presence of Alzheimer's disease. We can see that the model is placing more emphasis on certain smaller regions of the brain for the moderate to severe AD class (2) and the mild AD (1) class than it is for the no AD class (0). This information can be used to develop a deeper understanding of the disease, and to develop new treatments and interventions.



(u) Figure 24 E: **Class 2**: Coronal View



(v) Figure 24 F: **Class 2**: Sagittal View

It is interesting to note the model is also taking in a significant amount of noise from the negative (empty) space. This can attribute to a decrease in classification accuracy, generalization, and overall model performance. One of the challenges of interpreting models using saliency maps is that the results can be sensitive to the choice of input image. Small changes in the input image can result in large changes in the saliency map. This can make it difficult to draw conclusions about which features are truly important for the classification. Additionally, saliency maps only provide a local explanation of the model's behavior. They do not provide a global understanding of how the model is making its classification decisions. Despite these limitations, saliency maps remain a valuable tool for interpreting deep learning models. They provide a way to visualize the model's internal representation of the input data and can help identify which parts of the input are most important for making classification decisions. Overall, the saliency maps provide a powerful tool for interpreting the results of the model, and for gaining insight into the underlying biology of the disease.

GPU Backend

The Tesla M40 is a professional-grade GPU from Nvidia's Tesla series, designed primarily for deep learning and scientific computing workloads. It is based on the Maxwell architecture and has 3072 CUDA cores, 12 GB of GDDR5 memory, and a memory bandwidth of 288 GB/s. I personally assembled and configured this machine system to be used for this project. I made sure to install all the necessary drivers and compatible libraries, including NVIDIA and CUDA drivers, to ensure optimal performance.

The NVIDIA System Management Interface (nvidia-smi) is a command line utility, based on top of the NVIDIA Management Library (NVML), intended to aid in the management and monitoring of NVIDIA GPU devices. As we can see from the output of Figure 25, the total memory capacity of the GPU is 12288MiB ($\sim$ 12.884902 GB), and the process of training one DL model in one process is using 100% of the GPU's capacity.

For NVIDIA-SMI explanation: (A mebibyte equals 2^{20} or 1,048,576 bytes.) MiB

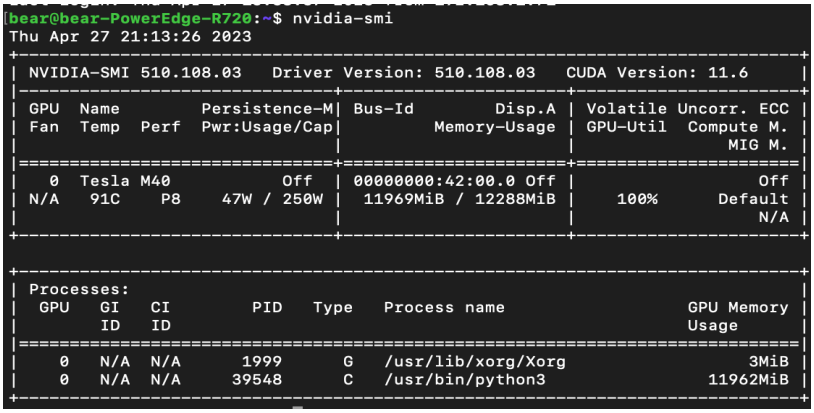


Figure 25: NVIDIA System Management Interface

Model	Batch Size	Memory Usage (GB)	Time (s)
VGG16	2	1.643 GB	35,067 s
VGG16	8	6.073 GB	32,263 s
VGG19	2	1.942 GB	41,676 s
VGG19	8	7.612 GB	38,425 s
ResNet50	2	1.584 GB	8,372 s
ResNet50	8	5.444 GB	6,924 s
DenseNet121	2	1.478 GB	4,260 s
DenseNet121	8	5.777 GB	3,194 s
DenseNet201	2	1.774 GB	6,432 s
DenseNet201	8	6.795 GB	4,923 s
ResNet152	2	2.391 GB	15,429 s
ResNet152	8	7.873 GB	13,227 s

Table 4: Model Memory Usage and Training Times.

Table 3 presents the results of varying batch sizes on the memory usage and training time of binary classification models. Increasing batch size resulted in higher memory usage but reduced overall training time. The same trend was observed in multi-class and regression models. However, it should be noted that the overall memory usage varied depending on the extent of fine-tuning, i.e., freezing of pre-trained layers. Similarly, the training times were not absolute, as the early stopping callbacks procedures varied for each model, despite setting a fixed number of epochs.

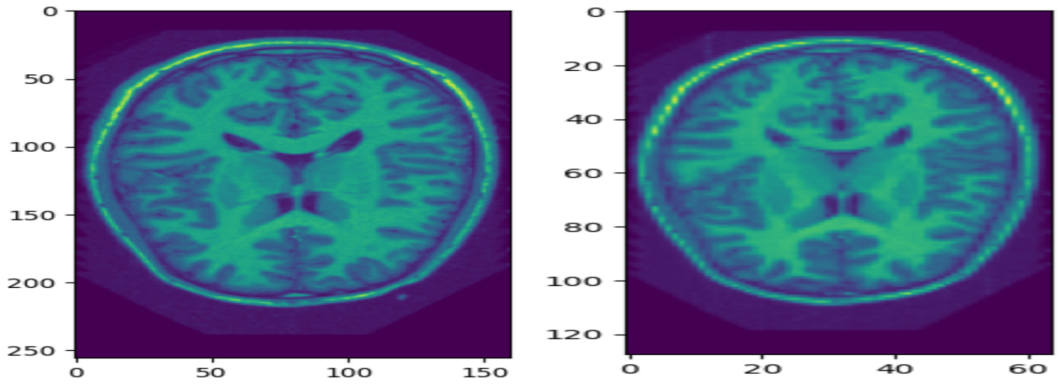
Conclusion

In conclusion, the results suggest that the ResNet152 model outperformed all the other models in the binary classification, multi-class classification and regression tasks. by achieving the highest F1-score and sensitivity. Moreover, it is also essential to consider both precision and recall scores for evaluating this model's performance in a classification task, especially for disease diagnosis. In this case, a high recall (sensitivity) score is very important as we aim to minimize the false negative rate, which is critical in detecting patients with Alzheimer's disease. Therefore, the superior performance of ResNet152 is an important finding, which may have practical implications for the diagnosis and treatment of AD. In terms of disease prediction, the ability to accurately predict the severity of Alzheimer's disease is crucial for patient care and treatment planning. A more accurate prediction of the disease severity would help clinicians in deciding the most appropriate treatment plan and improve the patient's quality of life. Therefore, the improved performance of the models, especially for the multi-class classification approach is significant from a clinical perspective. However, it is important to consider other factors such as training time and resource utilization when deciding on the best model for deployment. There is necessity for further research and validation in order to confirm these results and assess the generalizability of the models presented above to different datasets and populations.

Limitations

The use of deep learning (DL) architectures for the classification of Alzheimer's disease is a promising approach in medical imaging. However, there are limitations to such projects that must be considered. One limitation in this project is that the training data set was imbalanced. SMOTE oversampling could have been used to balance the data set. SMOTE generates synthetic samples in the minority class by interpolating between existing samples. This would have helped to avoid overfitting on the majority class and improved the accuracy of the model. I believe it could have been the better option instead of randomly oversampling the minority classes with replacement which allowed the model to train over the same image a few times, leading to an inability to generalize for the test data.

Memory usage was a significant limitation of this project. Running one model took up 100% of the GPU memory capacity, so only one model could be run at a time. This made hyper-parameter tuning challenging due to time constraints. Grid search algorithms could not be used because it was computationally expensive to find the optimal hyperparameters for every model. With a better and more efficient GPU, it would be possible to run multiple models in parallel and perform grid search algorithms for hyperparameter optimization. Another limitation due to the memory constraints was the resizing of the 3D MRI images down from [256 x 256 x 160 x 1] to [128 x 128 x 64 x 1]. This resizing led to a loss of information, and the original images could not be recovered. Although this approach helped in reducing the memory required for training, it could have negatively impacted the accuracy of the model., The overall resolution had decreased as the pixels in the scan were a lot more prominent. Figure 20 below allows us to truly visualize the loss of information in for **PatientID: 0005_MR1** after scaling down the scan →



(a) Full Image (Axial slice 128): $256 \times 256 \times 160 \times 1$

(b) Resized Image (Axial slice 64): $128 \times 128 \times 64 \times 1$

Another limitation of this project is that no data image augmentation was performed. Image augmentation is the process of artificially increasing the size of the data set by applying random transformations to the images, such as rotations, scaling, and flipping. In this project, the 3D MRI images were already T1 weighted and skull stripped, and all that was done was normalization of the images. Image augmentation could have improved the performance of the model, especially as the data set was relatively small with only 434 images, and the pre-trained model weights were from another dataset that was not similar to the one this project implemented. I believe that performing image augmentations would have helped with more efficient feature extraction. Another challenge in this project was deciding which specific slice of the brain to select for the visualization of saliency maps. It was a time-consuming process to find the exact coronal slice where the hippocampus and medial temporal lobe were to be visualized clearly. The saliency maps were averaged across all test subjects for the classes and displayed. However, saliency maps have limitations as they only highlight areas of the image that contribute most to the final classification score. It is not a complete visualization of the neural network's decision-making process.

I am immensely grateful for the opportunity to work on this project, which has provided me with a wealth of knowledge and insights about neurodegenerative diseases and deep learning applications. I feel extremely fortunate to have had the guidance and support of my mentor and research advisor, Professor Kyle Hasenstab, throughout this journey. His support and encouragement have been instrumental in helping me achieve my goals and develop the skills needed to be successful in the field of deep learning. I look forward to applying the knowledge and skills I have gained through this experience to future projects. Thank you for taking the time to read through this report.

References

Journal Articles

A. Farooq, S. Anwar, M. Awais and S. Rehman, "A deep CNN based multi-class classification of Alzheimer's disease using MRI," 2017 IEEE International Conference on Imaging Systems and Techniques (IST), Beijing, China, 2017, pp. 1-6, doi: 10.1109/IST.2017.8261460.

AbdulAzeem, Y., Bahgat, W.M. Badawy, M. A CNN based framework for classification of Alzheimer's disease. Neural Comput Applic 33, 10415–10428 (2021). https://doi.org/10.1007/s00521-021-05799-w

Bengio, Y. (2009). Learning deep architectures for AI. Foundations and Trends® in Machine Learning, 2(1), 1–127. https://doi.org/10.1561/2200000006

Braak, H., Alafuzoff, I., Arzberger, T., Kretzschmar, H., Del Tredici, K. (2006). Staging of alzheimer disease-associated neurofibrillary pathology using paraffin sections and immunocytochemistry. Acta Neuropathologica, 112(4), 389–404. https://doi.org/10.1007/s00401-006-0127-z

B. Bhuvaneshwari and A. Kavitha, "Investigations on the brain connectivity patterns in progression of Alzheimer's disease using functional MR imaging and graph theoretical measures," 2017 IEEE 16th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC), Oxford, UK, 2017, pp. 151-160, doi: 10.1109/ICCI-CC.2017.8109744.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L., 2009, June. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (pp. 248–255). Ieee.

Dou, Q., Chen, H., Yu, L., Zhao, L., Qin, J., Wang, D. (2016). Automatic detection of cerebral microbleeds from MR images via 3D convolutional neural networks. IEEE Transactions on Medical Imaging, 35(5), 1182-1195.

F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 1800-1807, doi: 10.1109/CVPR.2017.195.

F. M. J. M. Shamrat et al., "AlzheimerNet: An Effective Deep Learning Based Proposition for Alzheimer's Disease Stages Classification From Functional Brain Changes in Magnetic Resonance Images," in IEEE Access,vol. 11,pp.16376-16395, 2023, doi: 10.1109/ACCESS.2023.3244952.

Ghosh S., P. Boulanger, S. T. Acton, S. S. Blemker and N. Ray, "Automated 3D muscle segmentation from MRI data using convolutional neural network," 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 2017, pp. 4437-4441, doi: 10.1109/ICIP.2017.8297121.

He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778). https://arxiv.org/abs/1512.03385

He, K., Gkioxari, G., Dollár, P., Girshick, R. (2017). Mask R-CNN. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969). https://arxiv.org/abs/1703.06870

Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708). https://arxiv.org/abs/1608.06993

Huang, G., Liu, Z., Weinberger, K. Q. (2018). CondenseNet: An efficient densenet using learned group convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2752-2761). https://arxiv.org/abs/1711.09224

J. Duan et al., "Automatic 3D Bi-Ventricular Segmentation of Cardiac Images by a Shape-Refined Multi- Task Deep Learning Approach," in IEEE Transactions on Medical Imaging, vol. 38, no. 9, pp. 2151-2164, Sept. 2019, doi: 10.1109/TMI.2019.2894322.

Krizhevsky ASI, Hinton G. Imagenet classification with deep convolutional neural networks. Proceedings of the Advances in Neural Information Processing Systems, 2012; 1097–105

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., ... Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. Medical image analysis, 42, 60-88

Lee, J. H., Grosse, R. B., Ranganath, R., Ng, A. Y. (2018). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In Proceedings of the 26th international conference on machine learning (pp. 609-617)

Li C., L. Song, G. Zhu, B. Hu, X. Liu and Q. Wang, "Alzheimer's level classification by 3D PMNet using PET/MRI multi-modal images," 2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), Changchun, China, 2022, pp. 1068-1073, doi: 10.1109/EEBDA53927.2022.9744769.

Lin, M., Chen, Q. and Yan, S., 2013. Network in network. arXiv preprint arXiv:1312.4400

Liu W., J. Zhang and Y. Zhao, "A Comparison of Deep Learning and Traditional Machine Learning Approaches in Detecting Cognitive Impairment Using MRI Scans," 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), Los Alamitos, CA, USA, 2022, pp. 998-1001, doi: 10.1109/COMPSAC54236.2022.00154.

Morris, J. C. (1993). The Clinical Dementia Rating (CDR): current version and scoring rules. Neurology, 43(11), 2412-2414.

Marcus, D. S., Wang, T. H., Parker, J., Csernansky, J. G., Morris, J. C., Buckner, R. L. (2007). Open access series of imaging studies (OASIS): Cross-sectional MRI data in young, middle aged, nondemented, and demented older adults. Journal of Cognitive Neuroscience, 19(9), 1498–1507. https://doi.org/10.1162/jocn.2007.19.9.1498

M. D. Cirillo, D. Abramian and A. Eklund, "What is The Best Data Augmentation For 3D Brain Tumor Segmentation?," 2021 IEEE International Conference on Image Processing (ICIP), Anchorage, AK, USA, 2021, pp. 36-40, doi: 10.1109/ICIP42928.2021.9506328.

Shahajad M., D. Gambhir and R. Gandhi, "Features extraction for classification of brain tumor MRI images using support vector machine," 2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence), Noida, India, 2021, pp. 767-772, doi: 10.1109/Confluence51648.2021.9377111.

Pan, S.J. and Yang, Q., 2010. A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10), pp.1345–1359.

Qi H. et al., "Non-Rigid Respiratory Motion Estimation of Whole-Heart Coronary MR Images Using Unsupervised Deep Learning," in IEEE Transactions on Medical Imaging, vol. 40, no. 1, pp. 444-454, Jan. 2021, doi: 10.1109/TMI.2020.3029205.

Ronneberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention (pp. 234-241). Springer.

Shen D, Wu G, Suk HI. Deep learning in medical image analysis. Annu Rev Biomed Eng 2017;19(1):221–248. Simonyan, K., Zisserman, A. (2015, April 10). Very deep convolutional networks for large-scale image recognition. arXiv.org. https://arxiv.org/abs/1409.1556

Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B., Liang, J. (2016). Convolutional neural networks for medical image analysis: full training or fine tuning?. IEEE Transactions on Medical Imaging, 35(5), 1299-1312.

Ullah H. M. T., Z. Onik, R. Islam and D. Nandi, "Alzheimer's Disease and Dementia Detection from 3D Brain MRI Data Using Deep Convolutional Neural Networks," 2018 3rd International Conference for Convergence in Technology (I2CT), Pune, India, 2018, pp. 1-3, doi: 10.1109/I2CT.2018.8529808.

Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., Summers, R. M. (2017). ChestX-ray8: Hospital-scale chest X-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2097-2106).

Wang R. and C. F. Cheung, "3D Super-resolution Optical Imaging Using Deep Image Prior," 2021 International Conference of Optical Imaging and Measurement (ICOIM), Xi'an, China, 2021, pp. 5-8, doi: 10.1109/ICOIM52180.2021.9524418.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H., "How transferable are features in deep neural networks?", arXiv e-prints, 2014. doi:10.48550/arXiv.1411.1792.

Z. Fan, J. Su, K. Gao, D. Hu and L. -L. Zeng, "A Federated Deep Learning Framework for 3D Brain MRI Images," 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 2021, pp. 1-6, doi: 10.1109/IJCNN52387.2021.9534376.

Zhang, H., Cisse, M., Dauphin, Y. N., Lopez-Paz, D. (2018). Mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A. (2016). Learning deep features for discriminative localization. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp.2921-2929).

Websites & GitHub Repositories

Oasis Brains. OASIS Brains - Open Access Series of Imaging Studies. (n.d.). Retrieved April 30, 2023, from https://www.oasis-brains.org/data

The Brain Extraction Tool (BET). BET - FslWiki. (n.d.). Retrieved April 30, 2023,
from https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/BET

https://www.drivendata.org/competitions/65/clog-loss-alzheimers-research/page/215/

https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd

https://www.analyticsvidhya.com/blog/2021/06/confusion-matrix-for-multi-class-classification/

https://github.com/mudit14224/3d-image-classification

https://github.com/Dyex719/Alz-Net

https://github.com/srajan-kiyotaka/Alzheimer-Disease-Prediction/blob/master/Alzheimer_Disease.ipynb

https://github.com/Mikehem/GradCam/blob/main/notebooks/GradCam-Keras-VGG16.ipynb

https://github.com/ZFTurbo/classification_models_3D

https://github.com/qubvel/classification_models

https://towardsdatascience.com/hitchhikers-guide-to-residual-networks-resnet-in-keras-385ec01ec8ff: :
text=A%20building%20block%20of%20a,layer%20in%20the%20neural%20network.

https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

https://www.analyticsvidhya.com/blog/2021/05/tuning-the-hyperparameters-and-layers-of-neural-network-deep-learning/: :text=The%20hyperparameters%20to%20tune%20are,layers%20can%20affect%20the%20accuracy.

https://medium.com/analytics-vidhya/explained-output-of-nvidia-smi-utility-fc4fbee3b124

https://www.yourdatateacher.com/2021/05/19/hyperparameter-tuning-grid-search-and-random-search/

https://medium.com/swlh/how-to-use-smote-for-dealing-with-imbalanced-image-dataset-for-solving-classification-problems-3aba7d2b9cad

---

(END)