Detecting Alzheimer's Disease using Artificial Neural Networks

Advisor: Professor Xiao-Hua "Helen" Yu

Mia Keegan
Sally Lee

California Polytechnic State University, San Luis Obispo
Department of Electrical Engineering Senior Project

Abstract:

This project aims to use artificial neural networks (ANN) in order to detect Alzheimer's disease. More specifically, convolutional neural networks (CNN) will be utilized as this is the most common ANN and has been used in many different image processing applications. The purpose of using artificial neural networks as a detect method is so that an intelligent way for image and signal analysis can be used. A software that implements CNN will be developed so that users in medical settings can utilize this software to detect Alzheimer's in patients. The input for this software will be the patient's MRI scans. In addition, this is a project that is relevant with the current trends of an increase in development surrounding artificial intelligence. As technology has become more advanced, there has been an increase in medical developments as well. From the simulation, the hyperbolic tangent activation function provided the best results. The performance resulting from the two classifications when using the hyperbolic tangent function, on average was validation best accuracy of 81.10%, validation stopped tuning accuracy of 81.10%, training best accuracy of 100.00%, testing best accuracy of 68.94%, F-1 score of 70.06%, precision of 71.00%, and recall of 70.06%. This project will open doors to more applications of this detection method. More diseases other than Alzheimer's disease can utilize artificial neural networks (ANN) to detect diseases early on so that lives can be restored and saved.

Table of Contents

List of Figures

List of Tables

Chapter 1 : Introduction

The detection of Alzheimer's disease using artificial neural networks (ANNs) is an important problem that should be studied because Alzheimer's disease is a major public health challenge that affects a growing number of people worldwide. It is the 7th leading cause of death in the United States [1]. Early detection is essential for timely intervention and treatment, as it can help slow the progression of the disease and improve patient outcomes. However, the detection of Alzheimer's disease is complex and requires the analysis of large amounts of data from medical imaging, cognitive testing, and other sources. ANNs have the potential to provide a more accurate and efficient means of detecting Alzheimer's disease, as they can process large amounts of data with high accuracy and speed. Moreover, the use of ANNs for the detection of Alzheimer's disease can help improve our understanding of the disease and its underlying mechanisms, which can lead to the development of more effective treatments. Therefore, the detection of Alzheimer's disease using artificial neural networks is an important problem that has the potential to significantly impact public health.

With the rise of the problem, comes an increase in the amount of research development regarding this issue. Researchers from all around the world have run many different simulations using a combination of the CNN and other neural networks. The varying networks have provided a wide range of results from 70-90%. For instance, Marusina's and Bukhalov's [16] used both a CNN model and an Ensemble Learning algorithm to accurately classify patients. These individuals used the ADNI database and saw a low accuracy of around 73%. Another pair of researchers Chaihtra and Shetty [17] approached the issue by using various machine learning strategies. They preprocessed their images and were able to get a high accuracy level of 91%, which was provided when using the Deep Neural Network DenseNet121. Another group of researchers used different feature extraction methods and used the Kaggle database for their MRI images. Pranao, Harish, Dinesh, Sasikala and Kumar [18] fell short with an accuracy of 75% by using the SVM classifier. There is still much research going on in regards to this problem and there are many promising results that show that using CNN to detect Alzheimer's can help patients since early detection is vital.

Chapter 2 contains the literature review. After extensive literature review, the various advantages, and limitations of convolutional neural networks (CNNs) have been studied and examined. Convolutional neural networks (CNNs) have shown promise in detecting Alzheimer's disease from medical imaging data. Some of the advantages of using CNNs include their ability to automatically learn discriminative features from the data and their high accuracy in detecting subtle changes in brain structures. CNNs can also be used for automated, rapid, and objective screening, which can save time and reduce inter-observer variability. However, there are also some limitations to using CNNs for Alzheimer's disease detection.

One limitation is the need for large amounts of training data, which can be difficult to obtain, especially in the case of rare subtypes of Alzheimer's disease. Another limitation is that CNNs may not always be able to identify subtle changes in the brain that are not easily visible in medical images. Finally, the interpretability of CNNs is limited, which can make it difficult to understand how the network is making its predictions. In summary, while CNNs have shown promise in detecting Alzheimer's disease, there are also some limitations to their use, which need to be addressed to ensure their successful implementation in clinical settings. More detailed information and examples will be included in the following literature review chapter.

Chapter 3 provides technical background information on convolutional neural networks (CNNs) and training algorithms. CNNs are a type of artificial neural network that are widely used for image recognition and other computer vision tasks. The chapter introduces the basic architecture of a CNN, including convolutional layers, pooling layers, and fully connected layers, and explains how these components work together to extract features from an image and make a prediction. The chapter also covers important training algorithms for CNNs, including backpropagation, stochastic gradient descent, and batch normalization. These algorithms are essential for training a CNN to accurately recognize patterns in images and other visual data. Overall, the background chapter serves as a foundation for the reader to understand the technical concepts and terminology that are necessary to grasp the subsequent chapters that deal with the applications of CNNs.

Chapter 4 is the approach and simulation results chapter, which is a crucial component of the study. It will outline the entire methodology used in the research. It also explains how the parameters of the algorithm were determined, as well as the structure of the neural network used. In this chapter, all test conditions, simulations, and plots are presented. The data will be organized into tables and plots, and each variable on the axis of the plots will be explained in detail. Rather than simply listing the results, this chapter will include thorough explanations and discussions of the meaning of the results. Comparisons will also be made between different simulations and tests to draw conclusions about the effectiveness of the algorithm and neural network structure. Overall, the approach and simulation results chapter provides the reader with a clear understanding of the methodology and findings of the research and will be a critical component in evaluating the effectiveness of the proposed approach.

Chapter 5 is the conclusions and future work chapter, which is an important part of any study, as it summarizes the key findings and highlights the significance of the study. In this chapter, the methodology and results of the project will be briefly summarized, with a focus on explaining why the proposed method is better than other existing methods. The conclusion will draw on the results presented in the approach and simulation results chapter and provide a clear evaluation of the effectiveness of the

proposed method. The conclusion will also touch upon the limitations of the study and any areas where further research is needed to improve the proposed method. The future work section of the chapter will provide directions for future research, based on the limitations identified in the study. These directions will be focused on improving the proposed method, expanding its scope, or applying it in new domains. Overall, the conclusions and future work chapter provides a comprehensive overview of the study and its findings, as well as highlighting potential areas for future research. It serves as a guide for researchers who wish to build upon the work presented in the study and will help to further advance the field.

Chapter 2 : Literature Review

This chapter will explain the differences and similarities between different approaches to using CNN to detect early onset Alzheimer's Disease. The benefits and weakness will be discussed about each design structure and the results of these design structures.

In Marusina's and Bukhalov's paper [16], they explain their approach and methodology on how they constructed their CNN model to achieve their goal of giving an accurate classification on whether a patient's MRI detects early onset Alzheimer's Disease. The MRI images were acquired from the ADNI database. They used Python 3.8 and TensorFlow to program their study. The structure of their program starts with the dataset going through an image processing where the images will be converted to be the same dimensions and resolution and then will be broken into three sets of 2D images. These images will then be inputted into both a CNN model and an Ensemble Learning (EL) algorithm. The EL algorithm helps recognize highlights and will deliver a weighted vote for classification scores. The outputs from the CNN and EL will be delivered to the testing stage which will use a five-fold cross-validation. The results from the testing stage will then be compared. The test the quality of the CNN, they used four diverse conditions with a variety of different preparation with Alzheimer's Disease and Normal Cognitive patches. They found the CNN received higher accuracy results when prepared with Alzheimer's Disease and Normal Cognitive patches. Using the EL algorithm in parallel with the CNN was interesting choice and they should have done more testing to see the difference in accuracy when using the EL algorithm and when not. The accuracy of the CNN model, 73%, was fairly low compared with other models done. More layers in the CNN might be needed to increase the accuracy of the model.

In Chaihtra's and Shetty's 2021 paper [17], details an approach to building their program using multiple different Deep Learning Neural Networks. The paper explains all the research done to construct their system such as using different machine learning strategies: Logistic Regression (LR), Decision Tree (DT), and Support Vector Machines (SVM). The data used for the CNN network was acquired from ANDI database. Their program will have their classification separated into four categories: Mild Dementia, Moderate Dementia, Non-Dementia, and Very Mild Dementia. Their proposed system begins with pre-processing the MRI images by scaling images as color image channels. The pre-processed images will then be inputted into the Deep Neural Networks such as DenseNet121, MobileNet, Xception, and Inception-V3 for training. Then the model will be tested, and the classification will be given. After experimenting, their results pointed to that Transfer Learning and Fine-tuning generated the best accuracy results. The results showed the Deep Neural Network DenseNet121 gave the highest accuracy of 91% on test data. By using and testing the accuracy of multiple Deep Neural Networks allowed for them to find the highest accuracy Deep Neural Network instead of just using one Deep Neural Network and not

knowing whether it will return the most accurate results. The paper did not go into much detail about the pre-processing stage of the system, hard to know whether the images' resolutions were all the same. Splitting the input MRI images into multiple 2D images could have resulted into more accurate results.

In Pranao's, Harish's, Dinesh's, Sasikala's and Kumar's 2022 paper[18], three different methodologies constructed and tested. The first methodology used many different types of feature extract methods such as: Local Binary Pattern (LBP), Tamura, Gray Level Co-Occurrence Matrix (GLCM), Gray Level Run Length Matrix (GLRLM), and Segmentation based Fractal Texture Analysis (SFTA). After features are extracted, an initial label name will be given based on the different classifications: mild, moderate, very mild and non-demented. The images of the MRI scans were taken from Kaggle. The data frames will then be concatenated and converted into a feature. The training of the model will be then carried out by four classifier algorithms including SVM, LR, Random Forest (RF), and XG Boost. Then the model will be tested. The highest accuracy results of this methodology were 69.7% for Binary and 62% for Multiclass. The second methodology had similar structure except the images were initially converted into vectors and were processed as vectors for the rest of the model. The highest accuracy results for this method were 67% for Binary and 73% for Multiclass. For the last methodology, the final output layer of the CNN was modified to work as SVM classifier. The highest accuracy result for this method was 75%. It can be concluded from these results that the third methodology is the most accurate. The feature extraction was helpful in extracting the smaller details of the images and helped the program become more accurate. In the preprocessing stage, a filter was used to help get rid of noise within the image which can need to more accurate results.

This paper explains one methodology that relies more heavily on CNN model and other deep learning models [19]. This methodology begins with going through a dataset pre-processing stage where images will get resized, noise will be removed, image will be segmented, and morphology (smoothing edges) will happen. Then the images will be inputted into either the CNN model or the other deep learning models that include ResNet101, DenseNet121, and VGG16. The output of these models will then go into a hold-out validation process stage. The results will then be tested, and the results will be outputted. After running the different deep learning models, it was found that the CNN model had the highest accuracy, AUC, recall and lowest loss. In this paper, they explained how results other than accuracy are very important such as AUC and loss function. The AUC measures how well the model differentiates between positive and negative classes. A higher value of AUC is wanted. The loss function measures the deviation between validation and training values. A smaller value for the loss is better. To get these results the Keras python library was used which will be used in extracting these results in our structure.

In this paper a description of how the 3D-CNN, HadNet, was developed. The first step of this methodology is to send the images through a pre-processing stage. In this pre-processing stage cross-alignment will be done to all the images to reduce data variability. Next a skull-stripping procedure will be done since the skull will not be need in the detection of Alzheimer's Disease. Then the images will be inputted into the HadNet for processing of the MRI voxels. In this stage the features will be extracted to help make a classification. HadNet is made up of five different types of processing layers. The architecture of the HadNet can be separated into three sections: STEM, MAIN, and HEAD. STEM will down samples the MRI for heavy processing. The MAIN will acquire the hidden features. The HEAD will then implement its classification. Then the output will go into testing stage and will give the results of the testing. The final accuracy of this model is 83%. This model was tested with only 530 participants MRIs which is significantly less than most programs. Most models will be using around 6000 – 5000 participants MRIs. This can the accuracy result to be less trustworthy. There were less pre-processing steps down such as noise removal, segmenting the images and morphology. This methodology keeps the image in a 3D format which most models have converted into a 2D format.

In Ganesh's, Nithi's, Akshay's and Rao's 2022 paper [21], it describes how three different CNN models were used to be trained. These three models include VGG-16, Inception V3, and Xception. VGG-16 is a CNN that is used especially for image classification and object detection. VGG-16 is made up of sixteen different convolutional layers. Inception V3 is an image classification model with twelve convolutional layers. Xception is a CNN model with seventy-one layers. The first step in this structure is pre-processing where data augmentation will be done on the images. Data augmentation is where existing images will be augmented in different ways to produce more images to feed the model. These data augmented images will then be fed into the different CNN models. Then the MRI will be classified into either of the four categories: Mild Dementia, Moderate Dementia, No Dementia, and Very Mild Dementia. The next stage will be testing where the results will be given. The types of results that were looked at were accuracy and loss. The highest validation accuracy result was 75% for the VGG-16 model. Although the data augmentation was able to produce a billion of samples, the model is not actually given any new samples and that having less samples but having them be all completely different samples might allow for the model to be better trained and give more accurate results.

Similarly, the other papers that were reviewed, this paper uses three different CNN models [22]. The three CNN model that were used and tested in this paper was VGG-19, Inception V3, and ResNet50. The MRI images will be acquired from the ADNI database. The overall model is simple and starts with acquiring the MRI files. Then the pre-processing stage where the 3D files will be segmented into 2D images. The images will then be extracted, and an image extension will be attached. Then the images will

be grouped into three classes: Alzheimer's Disease, Mild Cognitive Image, and Normal Cognitive. The classes will then be fed into the three different type of CNN models. Then the classifications will go into the testing stage and give the results. Keras which is the high-level API of TensorFlow will be used to create and train the models of deep learning. Using VGG-19 CNN model was found to have the highest accuracy. After the first five iterations the validation accuracy was 78%, then after the next five iterations the validation accuracy was 89% and the last five iterations gave a validation accuracy of 98%. Although these validation accuracy rates are very high, the dataset that was fed through the models was only 54 patients. This is significantly smaller group of samples compared to other projects. This can cause misleading accuracy rates and would like to see a higher sample group used. The programs used such as Keras will be used by our project too.

Many of the design structured used similar methods such as using CNN models such as VGG-19 and Xception. Some designs decided to use more extract methods for prepping the data than others. All these different methodologies had their benefits and weaknesses.

Chapter 3 : Background

Artificial neural networks (ANNs) are a class of machine learning models that are inspired by the structure and function of the human brain. ANNs consist of interconnected nodes, or "neurons," which perform mathematical operations on the input data to produce output predictions. One popular type of ANN is the convolutional neural networks (CNNs), which is particularly well-suited for image and video processing tasks. Artificial neural networks have been proven to be efficient and the most used is convolutional neural networks (CNNs) [5]. CNNs have a multiple hierarchical network structure and is feedforward, which means that the input goes in the forward direction and goes through different layers [6]. Afterwards, the output is then outputted into its own layer. The typical structure of convolutional neural networks include: input layer, convolution layer, sampling layer (pooling layer), fully connected layer and output layer [7]. This is shown in Figure 1 below.

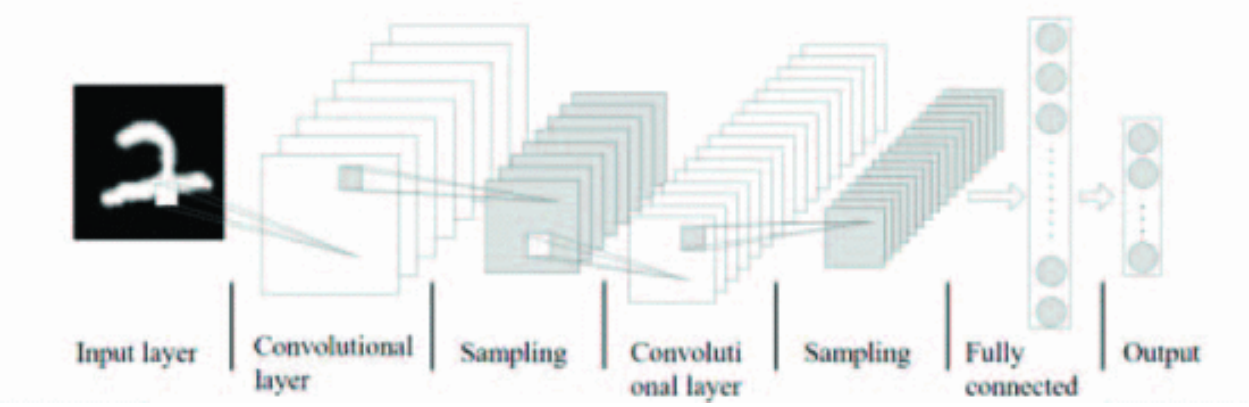


Figure 3.1: Typical structure of convolution neural network [7]

CNNs are designed to process inputs that have a grid-like topology, such as images, and are made up of multiple layers of interconnected neurons. The first layer of a CNN performs a set of convolutions, which apply a set of filters to the input image to extract features. The output of the convolutional layer is then passed through a series of activation functions, such as ReLU, to introduce non-linearity into the model. This process is repeated through several layers of the network, with the output of each layer being passed as input to the next layer, until the final layer produces a prediction. Since CNN is a feedforward network, it can extract topological properties and can recognize different types of patterns [6]. Therefore, many use CNN for image processing purposes since it has shown efficient performances for image processing and pattern recognition. Previously networks were text based so having this network that can process images and videos the usage of this network has increased exponentially such as the application of this network in medical settings [6]. In addition, CNNs also have fewer connections and parameters which allows for easier learning and training [8]. There are various CNN architectures proposed, such as

LeNet, AlexNet, ImageNet and GoogleNet [6]. These architectures have shown success in the medical field. For instance, AlexNet has been used to detect lung cancer and both AlexNet and ImageNet have been used to detect diabetic retinopathy in blood vessels [9][10]. These architectures are commonly used to detect medical conditions and show promising results that CNN can be applied to Alzheimer's disease detection as well.

Training a CNN involves adjusting the weights of the network so that it can accurately classify or detect features in the input data. This is typically done using a process called backpropagation, which involves computing the error between the predicted output and the true output and adjusting the weights of the network to reduce this error. The process of adjusting the weights is often done using an optimization algorithm, such as stochastic gradient descent, which iteratively adjusts the weights to minimize the error between the predicted and true outputs.

One of the key advantages of CNNs is their ability to learn and identify complex features in images and videos, such as edges, textures, and shapes. This makes them well-suited for a wide range of computer vision tasks, such as object detection, image classification, and semantic segmentation. Additionally, the ability to train CNNs using large datasets has made them a popular tool for many practical applications, such as self-driving cars, medical imaging, and facial recognition. CNNs and other ANNs are powerful tools for machine learning, offering the ability to identify complex patterns in large datasets and make accurate predictions on new data. As research in the field of machine learning continues to advance, we can expect to see even more powerful and effective ANNs being developed and applied to a wide range of real-world problems.

The Alzheimer's Disease Neuroimaging Initiative (ADNI) website, located at https://adni.loni.usc.edu/, is a major resource for researchers and clinicians working to understand the development and progression of Alzheimer's disease. The ADNI project is a large-scale collaborative effort involving multiple institutions, aimed at gathering and analyzing data on the biological, genetic, and environmental factors that contribute to Alzheimer's disease. The ADNI database includes a wide range of data from over 2,000 participants, including healthy elderly individuals, individuals with mild cognitive impairment, and individuals with Alzheimer's disease. Participants were between the ages of 55 and 90 who were recruited from the United States and Canada. The data is collected from multiple sites across the United States, using standardized methods and protocols. The data includes clinical, imaging, genetic, and cognitive data, and is made available to qualified researchers and scientists for analysis and research purposes.

The ADNI database is a rich resource for studying the development of Alzheimer's disease and related cognitive and neurological changes. The data includes information on various biomarkers, such as

brain imaging, cerebrospinal fluid (CSF) biomarkers, and genetic markers. The database also includes detailed clinical information, including medical history, medication use, and cognitive test results. The imaging data includes various modalities, such as magnetic resonance imaging (MRI), positron emission tomography (PET), and diffusion tensor imaging (DTI). One of the strengths of the ADNI database is its longitudinal nature, as participants are followed over time, allowing for the examination of disease progression and the identification of predictive biomarkers. The data in the ADNI database has been used in numerous research studies and clinical trials, aimed at developing new treatments and interventions for Alzheimer's disease. For example, the data has been used to identify early predictors of cognitive decline, to investigate the impact of lifestyle factors on disease progression, and to develop and test new drugs and interventions. In addition to the data in the ADNI database, the ADNI website also provides a wide range of resources and tools for researchers and clinicians. These resources include training materials, protocols, and standard operating procedures (SOPs) for data collection and analysis, as well as access to the data and the ability to request additional data or samples. The website also provides access to various software tools and pipelines for data analysis and processing. The ADNI database and website are important resources for researchers and clinicians working to understand and treat Alzheimer's disease. The rich and diverse data in the database has already led to numerous discoveries and breakthroughs and is likely to continue to be a valuable resource for years to come. This specific database was selected because of its accessibility, number of subjects and number of classifications.

Open-source code will be used for this project because it offers several advantages over proprietary or closed-source software. One of the primary advantages of open-source code is that it is freely available and can be accessed, modified, and distributed by anyone. This allows for greater collaboration and sharing of knowledge within the software development community, as well as promoting innovation and faster development. Open-source code is also often more reliable and secure than closed-source software, as it is reviewed and tested by a large community of developers, who can identify and fix bugs and vulnerabilities more quickly. In addition, open-source code often has a larger and more active user community, which can provide feedback and support to developers, leading to faster and more effective development. Another advantage of open-source code is that it can be customized and tailored to specific needs, as developers have access to the underlying code and can modify it as needed. This can lead to greater flexibility and cost savings for businesses, as they do not need to rely on proprietary software that may be expensive or inflexible.

Specifically, the open-source code used for this is Tensor Flow and Keras. TensorFlow and Keras are both popular open-source software libraries for building and training machine learning models. TensorFlow, developed by Google, is a powerful and flexible framework that allows developers to create

complex neural network architectures for a variety of tasks, such as image recognition, natural language processing, and more. TensorFlow offers a high-level API that allows for easier model construction and training, as well as lower-level APIs for more advanced model customization. TensorFlow also offers a wide range of tools and resources to support the machine learning development process, such as TensorBoard for visualizing model performance and TensorFlow Hub for sharing and discovering pre-trained models. Keras, on the other hand, is a high-level neural network API that can run on top of TensorFlow. It provides a simpler and more intuitive interface for building and training deep learning models, making it an excellent choice for beginners or those who want to quickly prototype and test their models. Keras offers a wide range of built-in layers, activation functions, and optimizers, and it allows for easy customization of neural network architectures. It also supports both CPU and GPU acceleration, making it a versatile tool for machine learning development. Overall, TensorFlow and Keras are both powerful and flexible tools that are widely used in the machine learning community. While TensorFlow provides more flexibility and control over the machine learning development process, Keras offers a simpler and more intuitive interface that can help developers get started more quickly.

Chapter 4 : Results

This chapter will go over the general approach to the detection and the results. The process includes slice extraction, data labeling, preprocessing, Discrete Wavelet Transform (DWT), CNN model creation, and model evaluation. The data used was obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database. ADNI1 consists of 800 participants and there were 2,294 complete brain scans provided in the dataset. Within the dataset, 476 scans represented the mild Alzheimer's disease classification, and 705 scans represented cognitively normal subjects. Each brain scan from the dataset was originally formatted in the Neuroimaging Informatics Technology Initiative (NifTI) and from each brain scan the axial perspective slices were extracted. The labels were provided in a comma-separated values (CSV) file with the corresponding image identification number. The images were preprocessed, and the CNN was used for classification. All the MRI scans were classified into the following two categories: Alzheimer's Disease (AD) and Normal Controls (NL). The original formatted brain scans were sliced and extracted using Python. The slices were saved in a JPEG format for processing, training, and testing. The basic methodology that was implemented is shown below in Figure 4.1. As mentioned previously, the classifier chosen for this approach is the Convolutional Neural Network. The CNN model can be shown below in Figure 4.2.

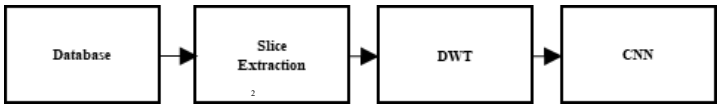


Figure 4.1: Basic Methodology

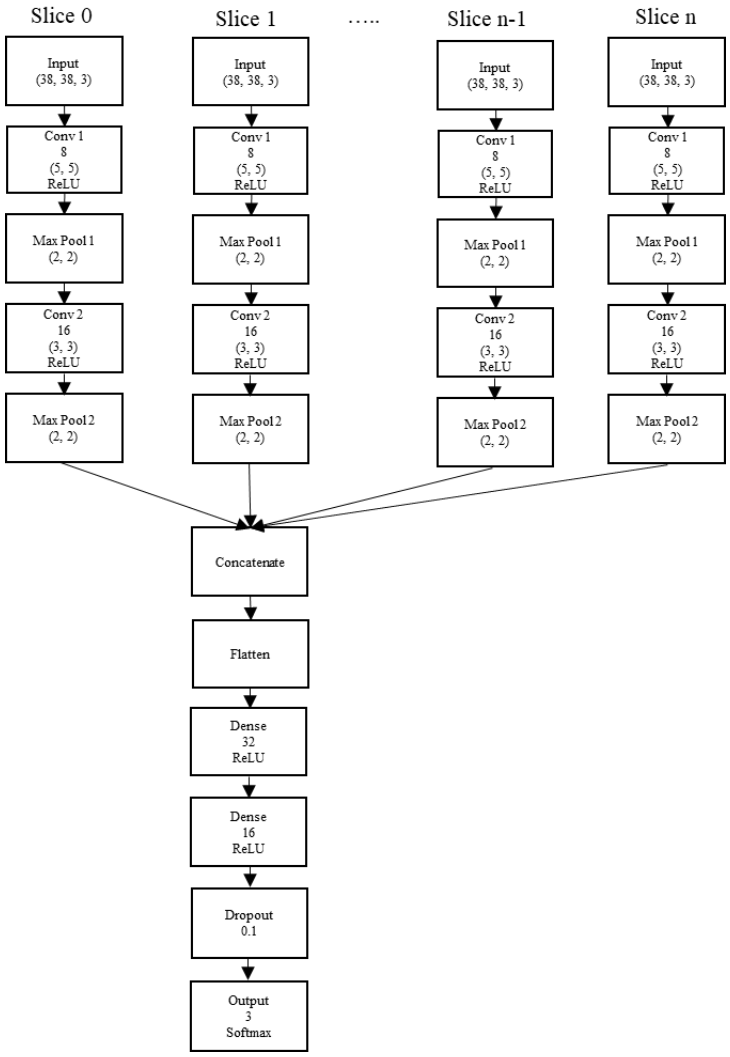


Figure 4.2: Convolutional Neural Network Model

Table 4.1: Training/Model Parameters for Testing Activation Functions

Training/Model Parameters	
Epochs	15
Batch Size	10
Slices	32
Slice Start Index	90
Dropout Rate	10%
Training/Validation/Testing Split	70/10/20%
Wavelet	Haar
Coefficients	HH2, HL2, LH2
Input Size per Brain Scan	38x38x3x32

Table 4.1 shows the parameters that are used to test the various activation functions in table form that were all mentioned above. The number of epochs used throughout the testing was set to 15 and the batch size was set to 10 for this series of testing. The slice count selected for comparison was 32 slices starting at the slice index 90. The training dataset represented 70% of the complete data, the validation set represented 10% of the complete dataset, and the testing set represented 20% of the dataset.

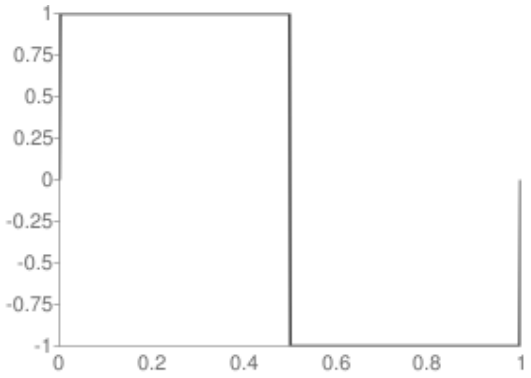


Figure 4.3: Haar Wavelet

The specific wavelet that was used is the Haar wavelet. This is shown in Figure 4.3. The Haar wavelet is the oldest and simplest wavelet. The images already went through preprocessing as DWT was used for feature extraction for this approach. The preprocessing changes the dimensions of the original image from 150 by 150 to 38 by 38. The coefficients from the second level of decomposition from the DWT were used as the inputs for the CNN model. The reasoning for choosing the second level is because noise is significantly reduced from the first level approximations coefficients. The coefficients that were extracted when using the DWT are the diagonal details (HH2), horizontal details (HL2), and the vertical details (LH2). These three coefficients were stacked together to form the three channels of the input. The size of the input from the Haar wavelet is 38 by 38. Therefore, the input size of each axial slice was 38x38x3 and the complete input size is dependent on the activation function that is being tested. These three channels provide different views of the MRI brain scan at the different slice indices. The images that were used from the dataset ranged from 190 to 256 slices depending on the MRI brain scan. In this approach, 190 slices were considered the maximum count. Previous testing done by Nardone [25] determined that 32 slices would be the ideal input for the classifier, which is why the initial input size was determined to be 38x38x3x32.

The performance of the different activation functions was measured. The parameter that is being assessed is the activation function (transfer function) of the neurons in the dense layer. The performance of three different activation functions will be assessed: ReLu, Hyperbolic Tangent function, and Logistic function. ReLu is the Rectified Linear Unit. It is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. The hyperbolic tangent function also known as Tanh is a function that takes any real value as input and outputs values in the range -1 to 1. The logistic function also known as the sigmoid function takes any real value as input and output values in range of 0 to 1. The activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input. The simplest activation function is the linear activation, and no transform is applied. A network comprised with linear activation functions are typically easy to train. Nonlinear activation functions are usually preferred to learn more complex structures in the data. Testing and validation of the performance of these functions can help with the selection process of the activation function that will provide the best results.

Simulations:

The average performance metrics for each activation function was observed. The accuracy and loss were computed during training for each model with two classifications. The loss was minimized during training to update the parameters and the accuracy was used to distinguish the best performing model. The metrics used to measure the performance of the activation functions include accuracy, precision, recall or sensitivity, and the f1-score. The three activation functions went through training, validation and testing.

ReLu simulation results:

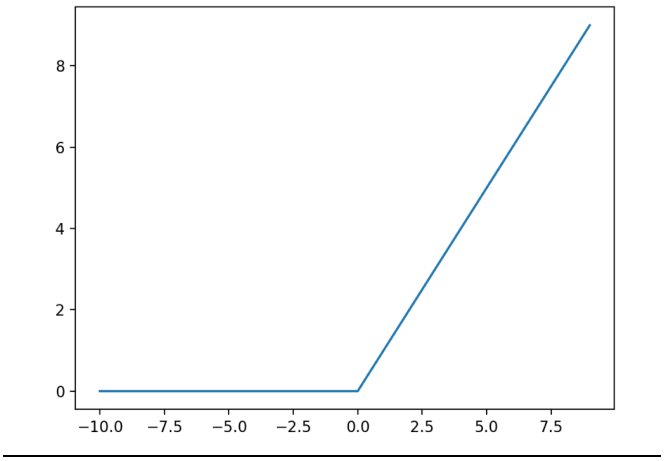


Figure 4.4: The Inputs vs Outputs for the ReLu Activation Function [24]

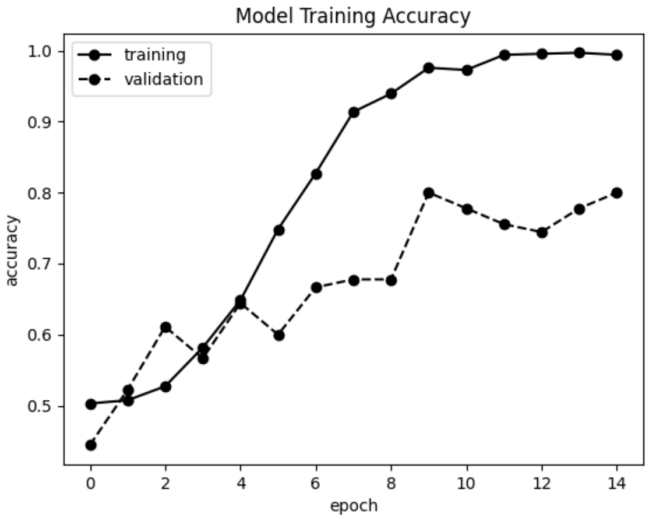


Figure 4.5: Model Training Accuracy for the ReLu Activation Function & Two Classes

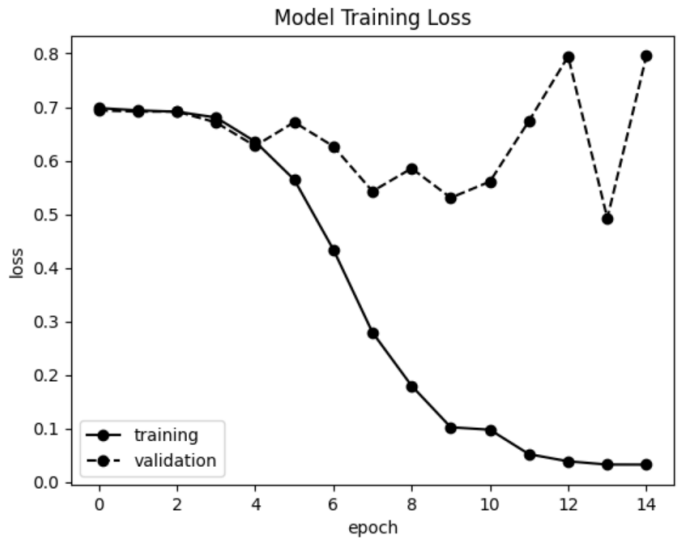


Figure 4.6: Modeling Training Loss Graph for the ReLu Activation Function & Two Classes

Table 4.2: Test Confusion Matrix for the ReLu Activation Function & Two Classes

True label \ Predicted label	AD	NL
AD	60	25
NL	43	62

Table 4.3: The training Confusion Matrix for the ReLu Activation Function & Two Classes

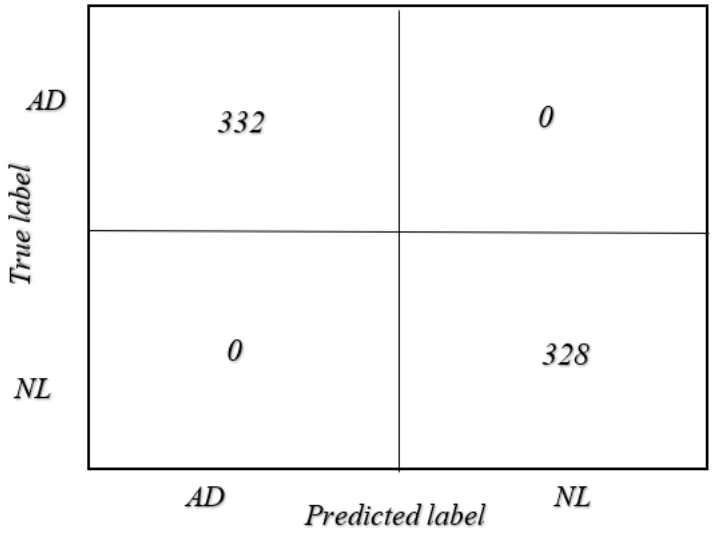
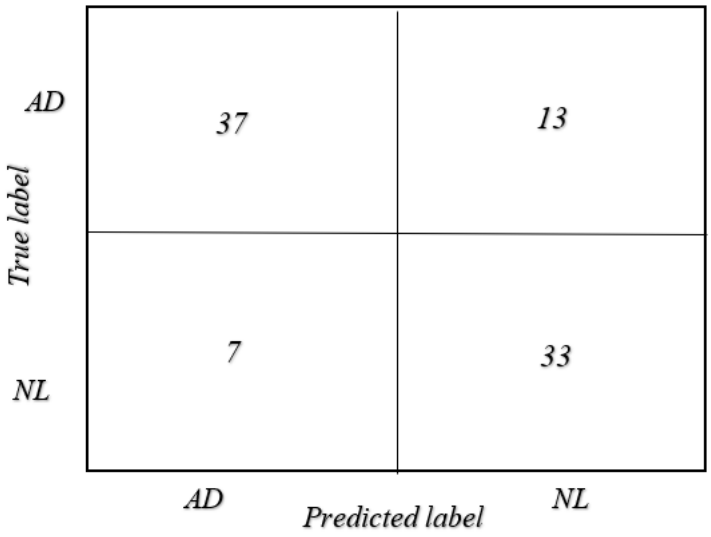


Table 4.4: The validation confusion matrix for the ReLu Activation Function & Two Classes



Hyperbolic Tangent Activation Function (Tanh):

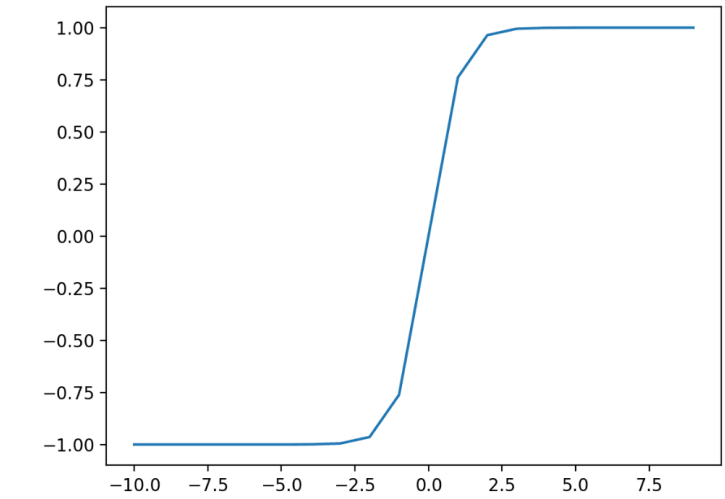


Figure 4.7: The plot of the Inputs vs. Outputs for the Tanh Activation Function [24]

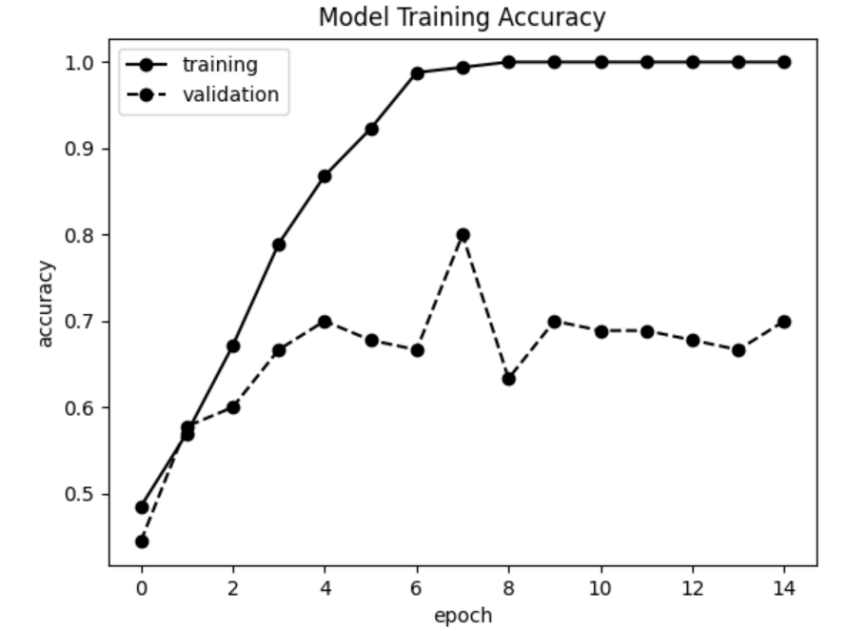


Figure 4.8: The Model Training Accuracy Graph for the Tanh Activation Function & Two Classes

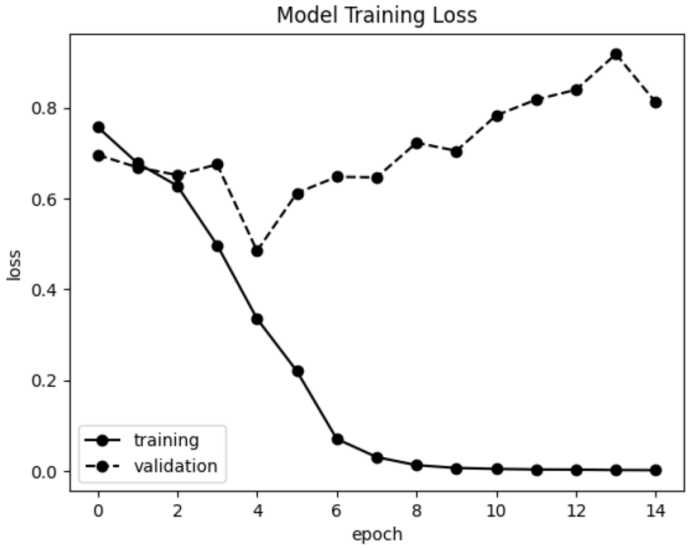


Figure 4.9: The Model Training Loss Graph for the Tanh Activation Function & Two Classes

Table 4.5: The test confusion matrix for the Tanh Activation Function & Two Classes

True label \ Predicted label	AD	NL
AD	58	27
NL	32	73

Table 4.6: The training confusion matrix for the Tanh Activation Function & Two Classes

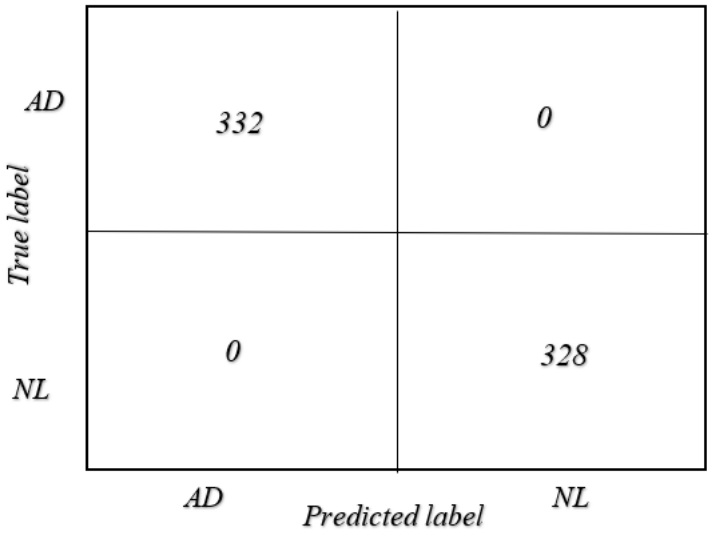
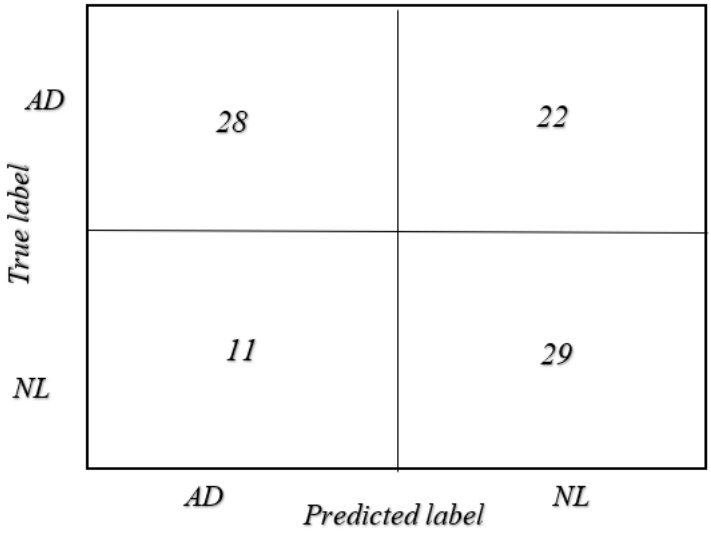


Table 4.7: The validation confusion matrix for the Tanh Activation Function & Two Classes



Logistic Activation Function (Sigmoid):

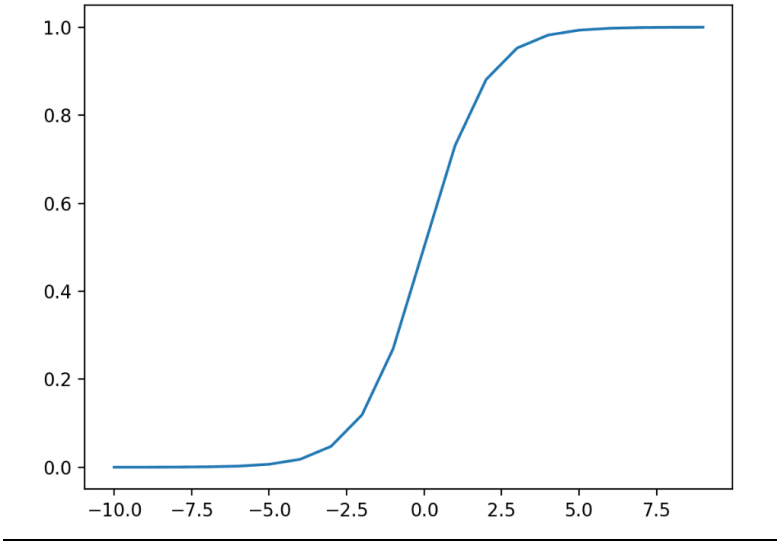


Figure 4.10: The plot of Inputs vs. Outputs for the Sigmoid Activation Function [24]

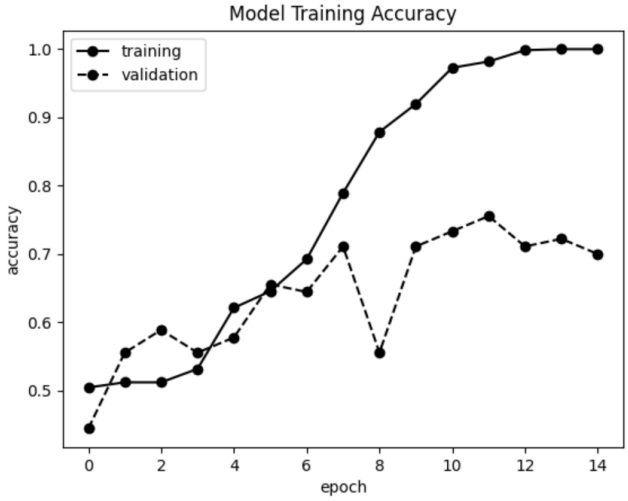


Figure 4.11: The Model Training Accuracy Graph for the Sigmoid Activation Function & Two Classes

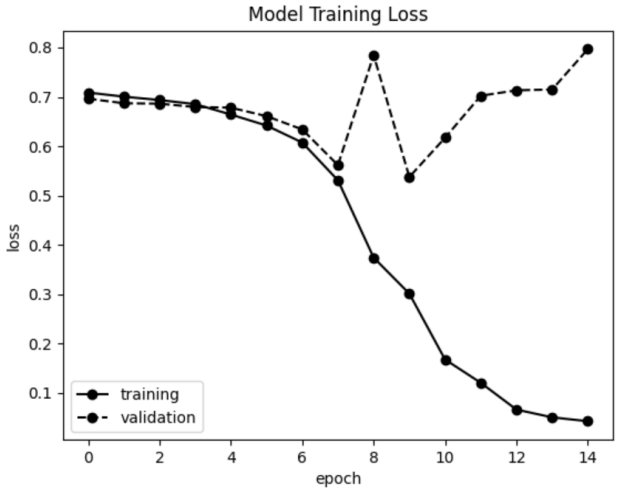


Figure 4.12: The Model Training Loss Graph for the Sigmoid Activation Function & Two Classes

Table 4.8: Testing Confusion Matrix for the Sigmoid Activation Function & Two Classes

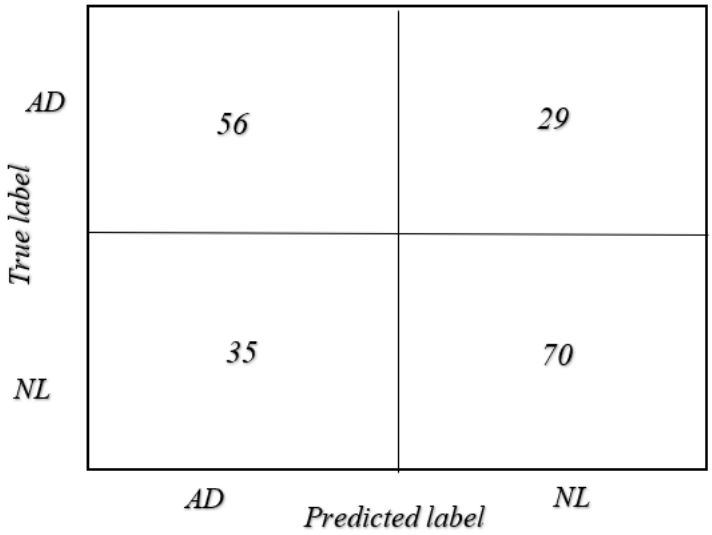


Table 4.9: The training confusion matrix for the Sigmoid Activation Function & Two Classes

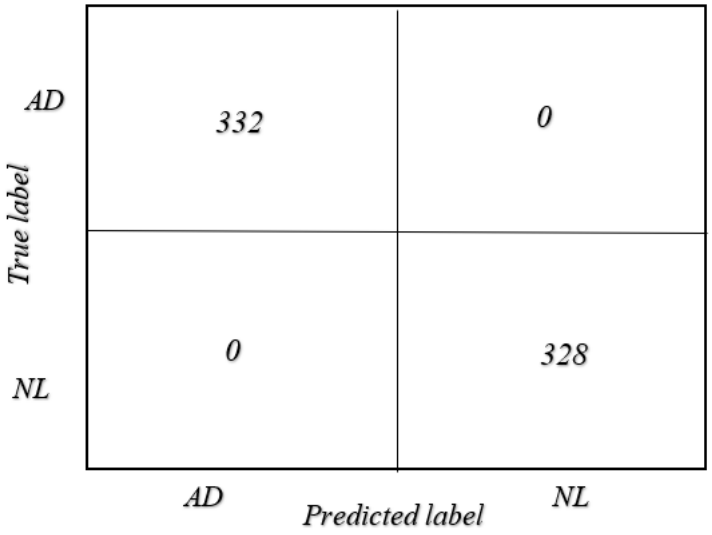


Table 4.10: The validation confusion matrix for the Sigmoid Activation Function & Two Classes

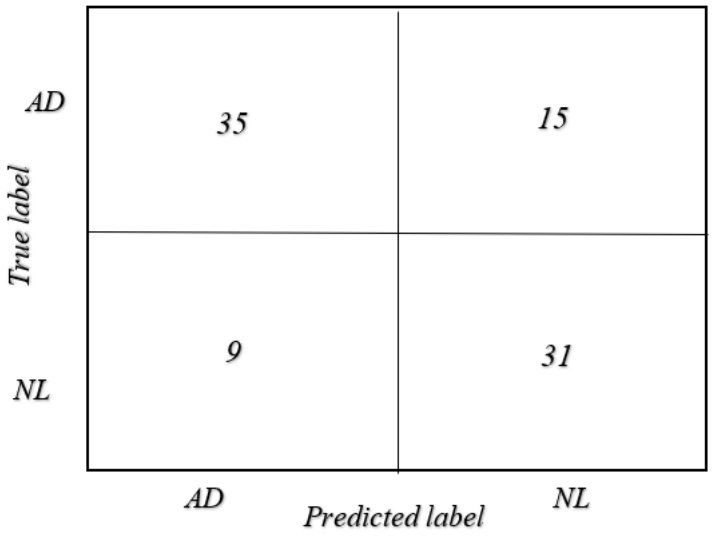


Table 4.11: Activation Functions Results

Activation Function	Validation Best Accuracy	Validation Stopped Tuning Accuracy	Training Best Accuracy	Testing Best Accuracy
ReLu	0.810	0.810	0.998	0.6421
Tanh	0.811	0.811	1.000	0.6894
Sigmoid	0.743	0.712	0.999	0.6631

Table 4.12: Activation Functions Testing F1-Score, Precision, and Recall/Sensitivity Results

Activation Function	F1-Score	Precision	Recall/Sensitivity	F1-Score
ReLu	0.6586	0.6675	0.6579	0.6586
Tanh	0.7006	0.7100	0.7006	0.7006
Sigmoid	0.7053	0.7053	0.7053	0.7053

Table 4.2 and 4.3 compare the performance of the three different activation functions (ReLu, Hyperbolic Tangent function, and Logistic function). Table 4.2 looks at the different categories of accuracy while Table 4.3 shows the F1-score, precision, and recall/sensitivity, which comes from testing. Based off the results, the function with the overall best performance is the hyperbolic tangent function (Tanh).

Table 4.13: Tanh Activation Function with Different Batch Sizes Results

Batch Size	Validation Best Accuracy	Validation Stopped Tuning Accuracy	Training Best Accuracy	Testing Best Accuracy
5	0.810	0.810	1.000	0.6789
10	0.811	0.811	1.000	0.6736
20	0.743	0.712	0.998	0.6421

Table 4.14: Tanh Activation Function with Different Batch Sizes Testing F1-Score, Precision, and Recall/Sensitivity Results

Batch Size	Testing F1-Score	Testing Precision	Testing Recall/Sensitivity
5	0.6586	0.6791	0.6793
10	0.7006	0.7006	0.7100
20	0.7053	0.6773	0.6975

The hyperbolic tangent function was rerun with the same training parameters with alterations to the batch sizes. The function was run with a batch size of 5 and 20. Table 4.4 and 4.5 showcase the difference in the various batch sizes. From the results, it can be concluded that the batch size of 10 provides the best results. Although there is little to no difference between the testing, training, and validation accuracies of the different batch sizes of 5 and 10, there are significant drops in the f1-score, precision, and recall/sensitivity. The accuracy levels were much lower for the batch size of 20.

Table 4.15: Tanh Activation Function with Different Slice Count Sizes

Slice Count Size	Validation Best Accuracy	Validation Stopped Tuning Accuracy	Training Best Accuracy	Testing Best Accuracy
16	0.7831	0.7125	0.981	0.6526
32	0.811	0.811	0.999	0.6947
64	0.735	0.655	0.997	0.7210

Table 4.16: Tanh Activation Function with Different Slice Count Sizes Testing F1-Score, Precision, and Recall/Sensitivity Results

Slice Count Size	Testing F1-Score	Testing Precision	Testing Recall/Sensitivity
16	0.6469	0.6507	0.6526
32	0.7006	0.7100	0.7006
64	0.7218	0.7292	0.7211

Although there was previous testing that was referenced to use the slice count of 32 for the initial input. The function was run with a slice count of 16 and 64 as well. Table 4.6 and 4.7 showcase the difference in the various slice count size. Table 4.6 shows the different accuracies at testing, validation, and training. Table 4.7 provides the testing results of F1-score, precision, and recall/sensitivity. Although the F1-score, precision and recall/sensitivity are higher for the slice count of 64, the different types of accuracy are best when the slice count is at 32. Therefore, from the results, it can be concluded that the slice count size of 32 provides the overall best results. In addition, the higher the slice count the longer it will take for the code to run. Overall, the simulation results showcase the difference in performance of the three activation functions in the dense layer and provides significant results that the hyperbolic tangent function provide the best results.

Chapter 5 : Conclusion and Future Work

Summary

From the results presented in the section above, there are promising results that show that the CNN model can assist with the detection of Alzheimer's disease. The model was trained to assess two classifications. The model parameter that was varied was the activation function used in the dense layer. It was found that the hyperbolic tangent activation function provided the best results. The performance resulting from the two classifications when using the hyperbolic tangent function, on average was validation best accuracy of 81.10%, validation stopped tuning accuracy of 81.10%, training best accuracy of 100.00%, testing best accuracy of 68.94%, f-1 score of 70.06%, precision of 71.00%, and recall of 70.06%.

Future Work

In the future, the performance of the model could be tested using brain scans from different datasets. ADNI is not the only dataset that is accessible to the general public. Other examples of datasets are NIMH Repository and OASIS. The current inputs are MRI scans, but there are alternative inputs that could also be assessed. For instance, other common brain scanning techniques could be used such as PET, CT, and EEG. In addition, other wavelets could also be used. The current model used Haar, but there are other wavelets such as Coiflet, Biorthogonal, etc.

References

[1]	"Alzheimer's disease," Centers for Disease Control and Prevention, 27-Sep-2022. [Online]. Available: https://www.cdc.gov/dotw/alzheimers/index.html#:~:text=Alzheimer's%20disease%20is%20the%20most,of%20death%20for%20all%20adults.
This reference is relevant as it provides a statistic on the impact of Alzheimer's disease, which is the issue that the project is addressing. This article and the author of the article ,the CDC, has authority and credibility since CDC is the national public health agency of the United States. It is a United States federal agency and is under the Department of Health and Human Services.	
[2]	Johnson, K. A., Fox, N. C., Sperling, R. A., & Klunk, W. E. (2012). Brain imaging in Alzheimer disease. Cold Spring Harbor perspectives in medicine, 2(4), a006213. https://doi.org/10.1101/cshperspect.a006213
This reference is relevant as it provides information on the current testing methods of Alzheimer's disease, which is the issue that the project is addressing. This publication , Cold Spring Harbor Perspectives in Medicine, and the authors of this publication have credibility as they are written by leading experts in each field and commissioned by a board of eminent scientists and physicians.	
[3]	Radiological Society of North America (RSNA) and American College of Radiology (ACR), "Alzheimer's disease," Radiologyinfo.org. [Online]. Available: https://www.radiologyinfo.org/en/info/alzheimers#:~:text=In%20the%20early%20stages%20of,the%20temporal%20and%20parietal%20lobes).
This reference is relevant as it provides more information on the effects of Alzheimer disease, more specifically it contains information on how to detect Alzheimer disease based on MRI scans and what areas of the brain will be most affected by the disease. The article also explains what to look at when doing a PET scan and CT scan.	
[4]	Dementia, World Health Organization. Sep. 19, 2019. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/dementia
This reference is relevant as it provides the basic information on Alzheimer's disease, which is the issue that the project is addressing. This article and the author of the article ,the World Health Organization, has authority and credibility since WHO is an organization of the United Nations that is responsible for international public health.	
[5]	H. Shin *et al*., "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning," in *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285-1298, May 2016
This reference is relevant since it gave an overview of CNN. The article explains as to why CNN architectures have become more commonly used and can be very effective compared to its counterparts.	
[6]	D. Arora, M. Garg and M. Gupta, "Diving deep in Deep Convolutional Neural Network," 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2020, pp. 749-751, doi: 10.1109/ICACCCN51052.2020.9362907.
This reference is relevant as it provides more information on convolutional neural networks and why CNN is relevant to this project as this network has efficient performance in image processing and pattern recognition.	

This article and the authors of this article have credibility as this was published in the IEEE database, which is a reputable organization that publishes trusted content.	
[7]	H. Li, "Computer network connection enhancement optimization algorithm based on convolutional neural network," 2021 International Conference on Networking, Communications and Information Technology (NetCIT), 2021, pp. 281-284, doi: 10.1109/NetCIT54147.2021.00063.
This reference is relevant as it included basic information on the typical structure of convolutional neural networks. This reference explains the different layers of this network. This is relevant as it shares information on a similar application and shares the goals of this current project. This article and the authors of this article have credibility as this was published in the IEEE database, which is a reputable organization that publishes trusted content.	
[8]	A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Advances in Neural Information Processing Systems, Lake Tahoe, Nevada, 2012, pp. 1097-1105.
This reference is relevant since the article went into more detail about the reasons why convolutional neural networks are more efficient than others. The advantage of having easier learning and training makes up for the slightly less accurate results.	
[9]	A. Agarwal, K. Patni and R. D, "Lung Cancer Detection and Classification Based on Alexnet CNN," 2021 6th International Conference on Communication and Electronics Systems (ICCES), 2021, pp. 1390-1397, doi: 10.1109/ICCES51350.2021.9489033.
This reference is relevant as it provides an example of AlexNet being used in a different medical context, specifically lung cancer detection. This is relevant as it shares information on a similar application and shares the goals of this current project. This article and the authors of this article have credibility as this was published in the IEEE database, which is a reputable organization that publishes trusted content.	
[10]	C. Jayakumari, V. Lavanya and E. P. Sumesh, "Automated Diabetic Retinopathy Detection and classification using ImageNet Convolution Neural Network using Fundus Images," 2020 International Conference on Smart Electronics and Communication (ICOSEC), 2020, pp. 577-582, doi: 10.1109/ICOSEC49089.2020.9215270.
This reference is relevant as it provides an example of AlexNet and ImageNet being used in a different medical context, specifically diabetic retinopathy detection. This is relevant as it shares information on a similar application and shares the goals of this current project. This article and the authors of this article have credibility as this was published in the IEEE database, which is a reputable organization that publishes trusted content.	
[11]	S. built by: Salary.com, "Entry electrical engineer salary in California," *Salary.com*. [Online]. Available: https://www.salary.com/research/salary/alternate/entry-electrical-engineer-salary/ca. [Accessed: 29-Nov-2022].
This reference is relevant as it provides information on entry level electrical engineering salaries in California. To determine the labor cost for our project we need to do research on what is a basic salary for an entry level electrical engineer.	
[12]	Coing, "Working Hours by country and industry," *Clockify*, 01-Sep-2017. [Online]. Available: https://clockify.me/working-

	hours#:~:text=On%20average%2C%20a%20full%2Dtime%20employee%20in%20the%20United%20Stats,more%20than%20other%20OECD%20countries. [Accessed: 29-Nov-2022].
This reference is relevant as it provides information on how the average amount of hours worked a year. This value will be used to determine the total labor cost of the project.	
[13]	R. Singhla, P. Singh, R. Madaan and S. Panda, "Image Classification Using Tensor Flow," 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), Coimbatore, India, 2021, pp. 398-401, doi: 10.1109/ICAIS50930.2021.9395939.
This reference is relevant as it provides information regarding the basic introduction to Tensor Flow. The paper goes over how Tensor Flow is used to classify images. It describes the different steps that go into image classification using Tensor Flow.	
[14]	B. Ambore, A. D. Gupta, S. M. Rafi, S. Yadav, K. Joshi and R. D. Sivakumar, "A Conceptual Investigation on the Image Processing using Artificial Intelligence and Tensor Flow Models through Correlation Analysis," 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2022, pp. 278-282, doi: 10.1109/ICACITE53722.2022.9823649.
This reference is relevant as it provides information regarding the basic introduction to Tensor Flow and image classification. This study paper provides an insight into image processing accuracy by “single-stage object detection model” or SSD MobileNetV2 and “Regional-based Convolutional Neural Network” or RCNN.	
[15]	Nagarathna and Kusuma, “Comparative study of detection and classification of Alzheimer’s disease using Hybrid model and CNN .” [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7169508. [Accessed: 03-Feb-2023].
This reference is relevant as it provides good information on a project very similar. Also provided more information on the structures of CNN and some coding information on CCN.	
[16]	D. Chaihtra and S. Vijaya Shetty, "Alzheimer’s Disease Detection from Brain MRI Data using Deep Learning Techniques," 2021 2nd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2021, pp. 1-5, doi: 10.1109/GCAT52182.2021.9587756. [Accessed: 15-Feb-2023].

This reference is relevant as it will be used during the literature review section. This paper explains the methodology on another approach to using CNN for early detection of Alzheimer's Disease. The paper also explains the results of their method and any future work needed on their project.	
[17]	N. C. R and K. M, "Comparative study of detection and classification of Alzheimer's disease using Hybrid model and CNN," 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON), Bengaluru, India, 2021, pp. 43-46, doi: 10.1109/CENTCON52345.2021.9688082. [Accessed: 15-Feb-2023].
This reference is relevant as it will be use din the literature review section. This paper details a method using CNN to detect early onset Alzheimer's Disease. The model of the CNN used many different types of Deep Learning Neural Networks.	
[18]	N. P. D, H. M. V, D. C, S. S and A. K. S, "Alzheimer's Disease Prediction Using Machine Learning Methodologies," 2022 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2022, pp. 1-6, doi: 10.1109/ICCCI54379.2022.9740942. [Accessed: 15-Feb-2023].
This reference is relevant as it will be used as one of the literature review papers. This paper details three different methodologies that were created and tested. All three had different approaches used. This paper explained different feature extraction methods.	
[19]	M. Mamun, S. Bin Shawkat, M. S. Ahammed, M. M. Uddin, M. I. Mahmud and A. M. Islam, "Deep Learning Based Model for Alzheimer's Disease Detection Using Brain MRI Images," 2022 IEEE 13th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, NY, USA, 2022, pp. 0510-0516, doi: 10.1109/UEMCON54665.2022.9965730. [Accessed: 15-Feb-2023].
This reference is relevant as it will be used as one of the literature review papers. This paper goes into detail about specific methodology of mostly relying on CNN model and other deep learning models. The paper explains the importance of other types of result other than accuracy.	
[20]	I. Sahumbaiev, A. Popov, J. Ramírez, J. M. Górriz and A. Ortiz, "3D-CNN HadNet classification of MRI for Alzheimer's Disease diagnosis," 2018 IEEE Nuclear Science Symposium and Medical Imaging Conference Proceedings (NSS/MIC), Sydney, NSW, Australia, 2018, pp. 1-4, doi: 10.1109/NSSMIC.2018.8824317. [Accessed: 15-Feb-2023].

This reference is relevant as it will be used as one of the literature review papers. In this paper, they describe the process and methodology of creating a 3D CNN architecture. In this 3D CNN architecture there are five different processing layers.	
[21]	C. H. S. C. A. Rama Ganesh, G. Sri Nithin, S. Akshay and T. Venkat Narayana Rao, "Multi class Alzheimer disease detection using deep learning techniques," 2022 International Conference on Decision Aid Sciences and Applications (DASA), Chiangrai, Thailand, 2022, pp. 470-474, doi: 10.1109/DASA54658.2022.9765267. [Accessed: 15-Feb-2023].
This reference is relevant as it will be used as one of the literature review papers. This paper discussed the use of three different CNN models and the results from the different model. Explain different programs used to build the models and collect the data from the models.	
[22]	M. T. Abed, U. Fatema, S. A. Nabil, M. A. Alam and M. T. Reza, "Alzheimer's Disease Prediction Using Convolutional Neural Network Models Leveraging Pre-existing Architecture and Transfer Learning," 2020 Joint 9th International Conference on Informatics, Electronics & Vision (ICIEV) and 2020 4th International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Kitakyushu, Japan, 2020, pp. 1-6, doi: 10.1109/ICIEVicIVPR48672.2020.9306649. [Accessed: 15-Feb-2023].
This reference is relevant as it will be used as one of the literature review papers. In this paper they discussed the three different CNN models that will be used. The three CNN models that were tested were VGG-19, Inception V3, and ResNet50.	
[23]	P. K. Rajendran, “Alexnet tensorflow 2.1.0,” *Medium*, 20-Jun-2020. [Online]. Available: https://medium.com/analytics-vidhya/alexnet-tensorflow-2-1-0-d398b7c76cf. [Accessed: 12-Mar-2023].
This reference is relevant since the structure for building the python code for the AlexNet CNN was used in the CNN python code function.	
[24]	J. Brownlee, “How to choose an activation function for deep learning,” MachineLearningMastery.com, https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/ [accessed May 17, 2023].
[25]	M. Nardone, "ALZHEIMER’S DETECTION WITH THE DISCRETE WAVELET TRANSFORM AND CONVOLUTIONAL NEURAL NETWORKS” 2022 [Accessed via Google Drive Cal Poly Thesis]

Appendix

Appendix A: Code

```
#
# cnn_model.py
# Author: Melissa Nardone
# Description: Creates and trains a CNN model for Alzheimer's detection.
# Usage: cnn_model.py [test_id]
#


import os
import pandas as pd
import tensorflow as tf
import numpy as np
import cv2
import time
import sys


from sklearn.model_selection import train_test_split
from tensorflow import keras
from keras.models import Model
example_model = tf.keras.Sequential()
BatchNormalization = tf.keras.layers.BatchNormalization
Conv2D = tf.keras.layers.Conv2D
MaxPooling2D = tf.keras.layers.MaxPooling2D
Activation = tf.keras.layers.Activation
Flatten = tf.keras.layers.Flatten
Dropout = tf.keras.layers.Dropout
Dense = tf.keras.layers.Dense
Concatenate = tf.keras.layers.Concatenate
Input = tf.keras.layers.Input
from tensorflow.python.keras import backend as K
Sequence = tf.keras.utils.Sequence
from tensorflow.python.keras.callbacks import (ModelCheckpoint)
from matplotlib import pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn import metrics
plot_model = tf.keras.utils.plot_model


###### Model and training parameters #####
BALANCE_DATA = 1  # equalize the number of brain scans in each category

IMG_SHAPE = (38, 38, 3)  # input coefficient size per slices
SLICE_COUNT = 32  # slice count
BATCH_SIZE = 20
EPOCHS = 15
INDEX_OFFSET = 15  # 90 # slice initial offset index

SRC_DIR = 'C:\\Users\\miake\\OneDrive\\Desktop\\level2'  # image directory
pointing to the DWT coefficients
```

```
# Define the classifications and classification labels
# label_dict = {'AD': 0, 'MCI': 1, 'NL': 2}
label_dict = {'AD': 1, 'NL': 0}

classification_labels = list(label_dict.keys())


###### Generate Model #####
inputs = []
outputs = []

# CNN Model
for i in range(SLICE_COUNT):
    input = Input(IMG_SHAPE)
    inputs.append(input)
    conv1 = Conv2D(8, (5, 5), activation='relu')(input)
    max_pool1 = MaxPooling2D((2, 2))(conv1)
    conv2 = Conv2D(16, (3, 3), activation='relu')(max_pool1)
    max_pool2 = MaxPooling2D((2, 2))(conv2)
    outputs.append(max_pool2)

if SLICE_COUNT == 1:
    flatten = Flatten()(outputs[0])
else:
    comb = Concatenate()(outputs)
    flatten = Flatten()(comb)

dense1 = Dense(32, activation='tanh')(flatten)
dense2 = Dense(16, activation='tanh')(dense1)
dropout = Dropout(0.3)(dense2)

output = Dense(len(label_dict), activation='softmax')(dropout)

model = Model(inputs=[inputs], outputs=[output])

print(model.summary())
tf.keras.utils.plot_model(model, to_file="test.png", rankdir="TB",
show_shapes=True)


##### Data Generator #####
def generate_dataframe():
    df = pd.DataFrame(columns=['img_id', 'classification'])

    # loop through all mri images
    for (dirpath, _, filenames) in os.walk(SRC_DIR):
        for file in filenames:
            data = file.rsplit('_')
            id = int(data[1])
            slice = data[2].rsplit('.')[-2]
            classification = os.path.split(dirpath)[1]

            if int(slice[5]) == 0:
                if classification in list(label_dict.keys()):
                    df = df.append({'img_id': id, 'classification':
classification}, ignore_index=True)
```

```
    df = df.sort_values(by=['img_id'], ignore_index=True)

    return df


df = generate_dataframe()

img_ids = df['img_id'].unique()
img_ids.sort()
image_df = df

##### Remove dataset imbalances #####
if BALANCE_DATA:
    class_df_list = []
    class_counts = []

    # get the id count for each classification
    for label in list(label_dict.keys()):
        class_df = image_df[image_df['classification'] == label]
        class_df = class_df.reset_index(drop=True)
        class_df_list.append(class_df)
        class_counts.append(class_df.shape[0])

    min_class_count = min(class_counts)
    min_class_idx = class_counts.index(min_class_count)
    min_class_label = list(label_dict.keys())[min_class_idx]

    # generate dataframe with balanced classes
    balanced_image_df = pd.DataFrame(columns=['img_id', 'classification'])
    for df in class_df_list:
        if df.shape[0] > min_class_count:
            df = df.drop(range(min_class_count, df.shape[0]))  # drop image
ids to make all classes equivalent
        print(df['classification'][0] + ': ' + str(df.shape[0]))
        # add dataframe back
        balanced_image_df = balanced_image_df.append(df)

    balanced_image_df = balanced_image_df.reset_index(drop=True)
    print(balanced_image_df)

    image_df = balanced_image_df

# train/test/validation split, 70%, 20%, 10%
train_images, test_validate_images = train_test_split(image_df,
test_size=0.3, random_state=42)
validation_images, test_images = train_test_split(test_validate_images,
test_size=0.67, random_state=42)

print('\nTraining IDs: ', len(train_images))
print('Validation IDs: ', len(validation_images))
print('Test IDs: ', len(test_images))

# save training/validation/test ids and labels to CSV
train_images.to_csv('training_images.csv', index=False)
validation_images.to_csv('validation_images.csv', index=False)
test_images.to_csv('test_images.csv', index=False)
```

```
# loading images helper
def load_image(path):
    if IMG_SHAPE[2] == 1:
        im = cv2.imread(path, 0)   # greyscale
    else:
        im = cv2.imread(path)

    im = np.array(im) / 255  # normalize

    return im


##### Data Generator #####
class DataGenerator(Sequence):
    def __init__(self, img_ids, labels, to_fit=True, batch_size=10,
dim=(IMG_SHAPE[0], IMG_SHAPE[1]),
                 n_channels=IMG_SHAPE[2], shuffle=True):
        self.img_ids = img_ids
        self.num_inputs = SLICE_COUNT
        self.num_classes = len(label_dict)
        self.labels = labels
        self.to_fit = to_fit
        self.batch_size = batch_size
        self.dim = dim
        self.n_channels = n_channels
        self.shuffle = shuffle
        self.on_epoch_end()

    def __len__(self):
        return int(np.floor(len(self.img_ids) / self.batch_size))

    def __getitem__(self, index):
        # Generate indexes of the batch
        indexes = self.indexes[index * self.batch_size:(index + 1) *
self.batch_size]

        # Locate the list of image _ids
        img_ids_temp = [self.img_ids[k] for k in indexes]

        # Generate data
        X = self._generate_X(img_ids_temp)

        if self.to_fit:
            y = self._generate_y(indexes)
            return X, y
        else:
            return X

    def on_epoch_end(self):
        self.indexes = np.arange(len(self.img_ids))
        if self.shuffle == True:
            np.random.shuffle(self.indexes)

    def _generate_X(self, img_ids_temp):
        # list, with an array for each input
        X = [np.empty((self.batch_size, *self.dim, self.n_channels)) for i
```

```
in range(self.num_inputs)]

        for i, img_id in enumerate(img_ids_temp):
            label_idx = np.where((self.img_ids == img_id))[0][0]
            classification = self.labels[label_idx]

            # load images
            for slice in range(SLICE_COUNT):
                path = os.path.join(SRC_DIR, classification,
                                    'img_' + str(img_id) + '_slice' +
str(slice + INDEX_OFFSET) + '.jpg')
                image = load_image(path)
                X[slice][i,] = image.reshape(IMG_SHAPE)

        return X

    def _generate_y(self, indexes):
        y = np.empty((self.batch_size, self.num_classes), dtype=int)
        # Generate data
        for i, index in enumerate(indexes):
            # Store sample
            y[i,] = label_dict[self.labels[index]]

        y = y.T

        # perform one-hot encoding
        return (keras.utils.to_categorical(y[0],
num_classes=len(classification_labels)))


def split_classification_data(data):
    x = data['img_id'].to_numpy()
    y = data['classification'].to_numpy()

    return x, y


X_train, y_train = split_classification_data(train_images)
X_test, y_test = split_classification_data(test_images)
X_valid, y_valid = split_classification_data(validation_images)

train_generator = DataGenerator(X_train, y_train, batch_size=BATCH_SIZE)
validation_generator = DataGenerator(X_valid, y_valid,
batch_size=BATCH_SIZE)

##### Train Model #####
training_metrics = ['categorical_accuracy']

# compile model
opt = tf.keras.optimizers.Adam()

model.compile(loss='categorical_crossentropy',
              optimizer=opt,
              metrics=training_metrics)

start = time.time()
```

```
# Test ID
test_id = 'tanh batch size 20'

# for saving only the model with the best performance
model_path = 'model_' + test_id
checkpoint = ModelCheckpoint(model_path, monitor='categorical_accuracy',
verbose=1, save_best_only=True, mode='max')
callbacks_list = [checkpoint]

print("Batch_size:", BATCH_SIZE, " \tStart time:", time.strftime("%H:%M:%S",
time.gmtime(start)))

# train the model
H = model.fit(
    train_generator,
    steps_per_epoch=len(train_generator),
    validation_data=validation_generator,
    validation_steps=len(validation_generator),
    epochs=EPOCHS,
    callbacks=callbacks_list)

end = time.time()
total_time = end - start
print("Training Complete. \tTotal Time: ", time.strftime("%H:%M:%S",
time.gmtime(total_time)))

# Model testing
history = H
print(history.history.keys())

# summarize history for accuracy
plt.figure()
plt.plot(history.history['categorical_accuracy'], 'ok-')
plt.plot(history.history['val_categorical_accuracy'], 'ok--')
plt.title('Model Training Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['training', 'validation'], loc='upper left')
plt.savefig('model_accuracy_' + test_id + '.png')

# summarize history for loss
plt.figure()
plt.plot(history.history['loss'], 'ok-')
plt.plot(history.history['val_loss'], 'ok--')
plt.title('Model Training Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['training', 'validation'], loc='lower left')
plt.savefig('model_loss_' + test_id + '.png')

# Get the best saved model
model = tf.keras.models.load_model('model_' + test_id)

train_generator = DataGenerator(X_train, y_train, batch_size=BATCH_SIZE,
shuffle=False)
validation_generator = DataGenerator(X_valid, y_valid,
batch_size=BATCH_SIZE, shuffle=False)
```

```
test_generator = DataGenerator(X_test, y_test, batch_size=BATCH_SIZE,
shuffle=False)

##### Model Testing #####

##### Generate Test Confusion Matrix #####
# generate predictions for test set for confusion matrix
y_pred = model.predict(test_generator)

y_pred = [classification_labels[y[0]] for y in (y_pred > 0.5).astype(int)]
metrics_dict = metrics.classification_report(y_test[0:len(y_pred)], y_pred,
target_names=classification_labels,
                                             digits=4, output_dict=True)
print(metrics.classification_report(y_test[0:len(y_pred)], y_pred,
target_names=classification_labels, digits=4))

matrix = confusion_matrix(y_test[0:len(y_pred)], y_pred,
labels=classification_labels)

accuracy = metrics_dict['accuracy']
f1_score = metrics_dict['weighted avg']['f1-score']
precision = metrics_dict['weighted avg']['precision']
recall = metrics_dict['weighted avg']['recall']

# save confusion matrix
plt.figure()
disp = ConfusionMatrixDisplay(confusion_matrix=matrix,
display_labels=classification_labels)
disp = disp.plot(cmap=plt.cm.Greys, colorbar=False)
plt.savefig('test_confusion_matrix_' + test_id + '.png')

##### Generate Validation Confusion Matrix #####
y_pred = model.predict(validation_generator)

y_pred = [classification_labels[y[0]] for y in (y_pred > 0.5).astype(int)]

matrix = confusion_matrix(y_valid[0:len(y_pred)], y_pred,
labels=classification_labels)

# save confusion matrix
plt.figure()
disp = ConfusionMatrixDisplay(confusion_matrix=matrix,
display_labels=classification_labels)
disp = disp.plot(cmap=plt.cm.Greys, colorbar=False)
plt.savefig('validation_confusion_matrix_' + test_id + '.png')

##### Generate Training Confusion Matrix #####
y_pred = model.predict(train_generator)


y_pred = [classification_labels[y[0]] for y in (y_pred > 0.5).astype(int)]

matrix = confusion_matrix(y_train[0:len(y_pred)], y_pred,
labels=classification_labels)

# save confusion matrix
plt.figure()
```

```
disp = ConfusionMatrixDisplay(confusion_matrix=matrix,
display_labels=classification_labels)
disp = disp.plot(cmap=plt.cm.Greys, colorbar=False)
plt.savefig('training_confusion_matrix_' + test_id + '.png')

# save the test performance in a CSV
test_performance_dict = {'training time': [time.strftime("%H:%M:%S",
time.gmtime(total_time))], 'accuracy': [accuracy],
                         'f1_score': [f1_score], 'precision': [precision],
'recall': [recall]}
test_peformance_df = pd.DataFrame(test_performance_dict)
test_peformance_df.to_csv('test_performance_' + test_id + '.csv')

print('TEST ID: ' + test_id)
```

```
#
# extract_slices.py
# Author: Melissa Nardone
# Description: Extracts the MRI slices from the ADNI dataset.
#

import os
import numpy as np
import nibabel as nib
import cv2

from matplotlib import pyplot as plt

SRC_DIR = ''
DEST_DIR = ''


def extract_slices():
    print('Extracting and saving slices...')
    total_slices = 0

    # loop through all mri images
    files = list()
    for (dirpath, dirnames, filenames) in os.walk(SRC_DIR):
        files += [os.path.join(dirpath, file) for file in filenames]

    for file in files:
        img = nib.load(file)
        data = img.get_fdata()

        # get filename
        dirname, fname = os.path.split(file)
        fname = fname.replace('.nii', '')

        # get diagnosis directory
        diagnosis = os.path.split(dirname)[1]
```

```
        # axial slices
        for idx in range(75, 176, 1):
            slice = data[idx, :, :]

            # normalize image data
            if (np.max(slice) != 0):
                slice = 255 * ((slice - np.min(slice)) / (np.max(slice)
- np.min(slice)))

            # convert image to jpeg
            dest_dir = os.path.join(DEST_DIR, diagnosis, fname +
'_axial_slice' + str(idx) + '.jpg')
            cv2.imwrite(dest_dir, slice)
            total_slices += 1

        # sagittal slices
        for idx in range(33, 134, 1):
            slice = data[:, idx, :]

            # normalize image data
            if (np.max(slice) != 0):
                slice = 255 * ((slice - np.min(slice)) / (np.max(slice)
- np.min(slice)))

            # convert image to jpeg
            dest_dir = os.path.join(DEST_DIR, diagnosis, fname +
'_sagittal_slice' + str(idx) + '.jpg')
            cv2.imwrite(dest_dir, slice)
            total_slices += 1

        # coronal slices
        for idx in range(75, 176, 1):
            slice = data[:, :, idx]

            # normalize image data
            if (np.max(slice) != 0):
                slice = 255 * ((slice - np.min(slice)) / (np.max(slice)
- np.min(slice)))

            # convert image to jpeg
            dest_dir = os.path.join(DEST_DIR, diagnosis, fname +
'_coronal_slice' + str(idx) + '.jpg')
            cv2.imwrite(dest_dir, slice)
            total_slices += 1

    print('\tTotal slices: ' + str(total_slices))

extract_slices('')
```

```
#
# extract_slices.py
# Author: Melissa Nardone
```

```
# Description: Extracts the MRI slices from the ADNI dataset.
#

import os
import shutil

SRC_DIR = 'ADNI/'
DEST_DIR = ''

def extract_source_files(base_dir):
    print('Extracting source files...')
    total_src_files = 0

    img_dir = os.path.join(base_dir, SRC_DIR)
    dst_dir = os.path.join(base_dir, DEST_DIR)

    # loop through all mri images
    files = list()
    for (dirpath, dirnames, filenames) in os.walk(img_dir):
        files += [os.path.join(dirpath, file) for file in filenames]

    for file in files:
        _, fname = os.path.split(file)
        dest_file = os.path.join(dst_dir, fname)
        shutil.copyfile(file, dest_file)
        total_src_files += 1

    print('\tTotal files: ' + str(total_src_files))
```

```
#
# wavelet_transform.py
# Author: Melissa Nardone
# Description: Performs feature extraction using the wavelet transform.
#

import os
import cv2
import pywt

import numpy as np

WAVELET = 'haar'
SRC_DIR = ''
DEST_DIR = ''

def wavelet_transform():
    print('Performing the discrete wavelet transform...')

    count = 0
    wavelet_type = WAVELET

    # loop through all mri images
```

```
    files = list()
    for (dirpath, _, filenames) in os.walk(SRC_DIR):
        files += [os.path.join(dirpath, file) for file in filenames]

    count = 0
    for file in files:
        img_data = cv2.imread(file, 0) # read in grayscale image
        img_resized = cv2.resize(img_data, (150, 150), interpolation =
cv2.INTER_AREA)

        # wavelet transform [LH, HL, HH]
        coeffs = pywt.wavedec2(img_resized, level=2,
wavelet=wavelet_type)

        classification = file.rsplit("\\")[-2]
        _, fname = os.path.split(file)
        fname = fname.replace('.jpg', '')

        # extract the approximation coefficients
        # path = os.path.join(DEST_DIR, 'LL', 'level2', classification,
fname + '.jpg')
        # cv2.imwrite(path, coeffs[0])

        for level in range(1, 3):
            # form an image with the 3 channels representing the detail
coefficients
            path = os.path.join(DEST_DIR, 'level' + str(3-level),
classification, fname + '.jpg')
            data = np.array([coeffs[level][0], coeffs[level][1],
coeffs[level][2]])
            cv2.imwrite(path, data.T)

        count = count + 1

    print('\tTotal images: ' + str(count))

wavelet_transform()
```

Appendix B: Mia Keegan's ABET Analysis

Detecting Alzheimer's Disease using Artificial Neural Networks

ABET Analysis

By: Mia Keegan

Advisor: Professor Helen Yu

The goal of our project is to be able to build a software program that will detect early onset Alzheimer's disease from a Magnetic Resonance Image (MRI) scan. The program will take an input of MRI scan and will use artificial neural networks to determine the probability of early onset Alzheimer's disease. The program will output a document filled with percentages of the different patterns seen in the MRI and probability of the disease being detected.

One of the engineering specifications for this project is to have an accuracy rate of 90%. This is a high percentage and can be difficult since there can be false positives or false negatives in the database that will be used to train the program. These false positives or false negatives can lead to inaccuracies with the program results. Another constraint to the project is that the MRI scans in the database will be in Nifiti file format. This is a 3D file format. This file format is not very common so determining how to handle the file will be important. Another difficulty was making sure all the files from the MRI database are the same size file. The program takes in a specific size file type so making sure the file size is accurate is important. Inaccuracy in file size can cause the program to not run and give out a result.

This project will be fully on the computer. There will be no direct need for natural resources. The only required equipment will be a computer. The program will be installed onto an already preexisting computer at the hospital. The only human labor will be the coders of the program and any doctor or hospital employee who uses the program. There should be no need for cash capital since a lot of the coding will be done on python and all the open-source code is free. The cost of the product will start accruing as technology start improving which means the program will need to be revised to keep up with the improvements. Since this product is software base, there will need to be updates made either yearly or in a shorter period. Computers will have updates, so the software program must be able to keep up with those updates. The estimated development time was 150 hours. That results in about $3,000.00 in labor cost. There will be additional labor cost as maintenance has to be done on the software program.

There are about six thousand hospitals located in the US. There are about thirty-three health centers that specialize in Alzheimer's Disease. I expect that in the first year there will be about fifty software packages that are sold. But as different hospitals use the software and proves the product's reliability there will be more products sold every year. The manufacturing cost for each device will be mostly due to labor cost from the programmers. But the majority of the labor cost was done in the development stage of the project. Since there will be little to no cost for commercial manufacturing, the product will be free to download. The profit will come from advertising. There will be no specific cost to operate the program itself, but the power used by the computer to run the program will be what costs the customer. This program will be run in hospitals who have equipment that consumes a lot more power than the computer will to run the program. So, the cost to the customer to run the program will be negligible.

One environmental concern is the effect that desktop computers power consumption has on omission levels. An average desktop computer uses on average 200 Watts per hour. So an computer that is on for eight hours a day will use around 600 kW and that will result in 175 kg of CO_2 per a year [1]. Global warming is a very important issue around the world and there needs to be a quick reduction in emissions to reduce the effects. So, by having our program run on a desktop computer at a hospital will only increase the CO_2 emissions. The program will not be run every day at each hospital but will increase the emissions.

Unlike other products, there will be no actual physical manufacturing of the product. The product will instead be all programmed. This means that it is very important that the code for the programs gets copied correctly. A slight change in the code can cause issues with the results of the product. The results of this product can be very sensitive to whomever is receiving them. So, if there was code that was changed that can cause inaccurate results and can affect the end user. It is very important that the code of the product is copied exactly over.

As time goes on, technology improves and evolves. This means that software programs will need to be able to keep up with the change in technology. This means that has computers improved, there might need to be changes to the software to be compatible with the newest technology. This means that constant updates will need to be made and this can be difficult to keep up with. This can also be difficult for the customers since they will need these updates installed or will need the program reinstalled if the computers get switched out. These programs will run off computers that use electricity. Not all electricity is from renewable sources, such as oil or gas. The continued use of these resources is not sustainable. An upgrade that would help increase the sustainability of the product is making the program's power consumption as low as possible. This will reduce the consumption of non-renewable energy sources.

Since the results of our product will determine whether someone has a serious disease, it is important that the patient knows all the information about the program and the results that are output. This program does not have a 100% accuracy rate which means that there could be false positives or negatives. The program will have a 90% accuracy rate which means most results will be correct, but it is important to let the patient know that it is not 100% accurate. If the doctor or hospital employee were not to give this information to the patient this will being going against the IEEE Code of Ethics promise to uphold the highest standards of integrity [2]. This can affect the health of the public.

Continuing about the product not being 100% accurate, if there is a false positive or negative this can mislead the next steps. If there was a false positive, the patient might start treatment to help slow the pace of the disease. The problem is that the patient does not actually have the disease and could end up having had side effects to the treatment that was not needed. For false negatives, this means treatment might be delayed until the disease is in later stages. This means that the disease might progress into worst symptoms when it could have been treated and delay those symptoms. The product is 90% accurate, meaning that the majority of the time their results will be correct. As improvement get made the accuracy rate will increase and reduce the number of false positives and negatives.

This product will mainly impact individuals who are believed to have early stages of Alzheimer's disease. Some other stakeholders include doctors and hospital employees since this can help give their patients more information. But this also means that the doctors and hospital employees' reputation rely on how the product being accurate. Some social concern could be that such an important diagnosis should be made by a doctor and not a computer whose accuracy rate is not 100%. There are many people who believe that computers are not to be trusted and that they are taking jobs away from humans. This product will not replace anyone job but more aid doctors and save doctors time by them not having to hand look at

MRI to find the diagnosis. Overall, there are a lot of positive effects the product will have on society. This product allows patients who test positive for Alzheimer's disease to start treatment that will manage their symptoms and hopefully prevent some cognitive decline from the disease.

After doing some independent research, there were open-source databases known as AlexNet, ImageNet and GoogLeNet. ImageNet is a large database with images that AlexNet and GoogLeNet use. These databases are convolutional neural networks architecture with layers that help detect patterns within images. ImageNet organizes challenges where architectures like AlexNet and GoogLeNet can compete and prove the accuracy of their architecture. These architectures allowed for easier coding.

Appendix C: Sally Lee's ABET Analysis

Sally Lee

ABET Analysis of Senior Project Design

The project is aimed to use artificial neural networks (ANN) to detect Alzheimer's disease. More specifically, convolutional neural networks (CNN) will be utilized since this is the most used ANN. The main function of the software program is to take an input, which is the patient's MRI scans and then output the classification of those scans. Within this main function, there are three smaller functions. These functions include converting the input file, processing the image, and classifying it, and then finally the function that displays the information in an output file. The purpose of using artificial neural networks as the detection method is to provide an intelligent way to analyze images and signals, specifically in medical applications. This project is designed to be used in medical settings and the user is expected to be someone working the field with little to no knowledge on the specifics of how the neural network works. Therefore, the software program will be straightforward and user friendly for this application.

When initially starting this project, there was a significant challenge with understanding the basics of artificial neural networks. This information was not learned in a classroom setting, which required all outside research and learning to fully understand the concept. Furthermore, there were also limitations surrounding the medical information pertaining to Alzheimer's disease. Much research was needed to get a grasp on information surrounding the MRI scan and its specifications. Since this is project that is focused on software and open-source code is going to be used, there were limitations surrounding the type of language that would be used to code the program. Since the purpose of the project is to use artificial neural networks and use it in an image processing application, the direction of the approach of the project was very straightforward. When choosing a specific network and understanding what needs to be done, the project goal limited the options as certain networks are more commonly used for these types of applications.

The project results in various economic impacts. In terms of human capital, the knowledge gathered regarding artificial neural networks, Alzheimer's disease and MRI scans assist in the development of the project. The knowledge gained from this project can pass between individuals. In addition, there is not much human capital since the project aims to have a computer classify MRI scans in

place of a human. Regarding financial capital, since there were no physical components of the project the only financial capital would be the necessary labor costs and the singular material cost for MATLAB. For manufactured or real capital, the project needs certain tools and technology ensuring completion, such as MATLAB. However, the program itself should be able to run on any computer. In terms of natural capital, this is limited to the current usage of the Earth' resources in computers that have already been built and are in use. Costs and benefits accrue during various portions of the project's lifecycle. Costs accrue during the development of the project because of the labor going into it as well the acquisition of software to develop the project. The only input to the system is the MRI scans of a patient. The only cost accrued during this project was purchasing MATLAB with the neural network toolbox. The student edition costs roughly $100. This project does not earn any profit, but the goal of this project is to assist in patient care of Alzheimer's disease since early detection is vital to treating the patient. The final product will emerge once and will last for however long it is needed to use since the user determines the life of the use of the software in their workplace. No additional maintenance or operation costs occur apart from regular computer maintenance. The total time spent on the project from beginning planning stages is nine months. The original estimated development time and the actual development time of the project will be roughly six months. To be more specific, the time spent on the development of this project will be around 20 weeks. After the project ends, future work can be done to improve the results or add more features to the program.

Since this project has an outcome of a software program, there is no manufacturing involved with the program. This means that the project would not be manufactured on the commercial basis. Therefore, there are no estimates and specifics regarding the manufacturing of the project on a commercial basis. In addition, software distribution typically has little to no cost. Many different software programs are readily available to download on any computer.

There are little to no environmental impacts associated with the use of this project. The only natural resource needed is electricity to run a computer because that is the sole source of power required to run a software program. Thus, this project only uses natural resources indirectly to generate power (e.g., coal, oil, natural gas). In addition, the amount of energy used should not be substantial enough to have any significant effects on the environment. The project does not harm or improve any of the natural resources and ecosystem species. The detection of Alzheimer's disease through a neural network does not impact other species other than humans.

As mentioned previously, the product that comes out of this project is a software program. This means that there is no physical component to the project and "manufacturability" is limited to software

distribution. The only issue that could arise with software distribution is issues with the installation of the software program. Therefore, there are no major issues or challenges associated with manufacturing.

Regarding the topic of sustainability, the project has no direct impacts on the sustainable use of resources. The project is a software program, which requires no physical maintenance since there are no actual physical parts. Again, the only manufacturing is the distribution of the software. Therefore, there are no issues or challenges associated with keeping this system sustainable. In terms of indirect impacts, the only concern would be the disposal of a computer after the product end life. In terms of upgrades for the design, there is always ways to configure the hidden layers of the network. This is a challenge since this means that there are endless possibilities to upgrade the design.

According to IEEE Code of Ethics Section II, an engineer is required "to treat all persons fairly and with respect, to not engage in harassment or discrimination, and to avoid injuring others." This is particularly crucial concerning the development of the neural network to detect Alzheimer's disease. The network is classifying an image (MRI scan) that is has processed. The validity of the neural network classification of the patient's MRI scan must be well known to all users of the system. If a wrong classification is made, the safety of the patient can be put at risk. In these situations, the ethical framework, Utilitarianism, applies. This framework is relevant because the project's ability to detect Alzheimer's disease encourages the safety for large groups of people, which complies with the Utilitarianism concept of "greatest good for the greatest number". The project and this concept share a common interest in treating everyone equally and making decisions to ensure the safety of the public.

In terms of the design itself and the distribution of the software, there are no health and safety concerns. The only concern is the risks that come with the actual outcome after the use of the project. In particular, the issue of a false positive or false negative is a concern. This is an issue since if a patient is misdiagnosed, a problem could arise since early detection is vital for treatment of Alzheimer's disease.

The project impacts society since anyone is susceptible to having Alzheimer's disease. A social and political issue with the use of this project could be that people are not open to this method of detection since it might be viewed as inadequate, and some people could have strong opinions about a computer analyzing their scans instead of a medical professional. The direct stakeholders of this are anyone who is looking to get diagnosed via this project as well as the intended users of the project. Therefore, the project can harm stakeholders if inaccurate classifications are provided. On other hand, the project can benefit stakeholders by properly detecting the disease, which allows the patient to get the help they need. Indirect stakeholders are those who are also looking at ways to use artificial neural networks to

detect Alzheimer's disease. The stakeholders should benefit equally if the project proves to be successful. The project does not create any inequities.

Throughout these ten weeks, there was an extensive list of new tools and techniques that were learned and used to assist in the progress of the course of the project. For instance, the ADNI database is very useful in the development of the project. This database includes various datasets of patient files. In addition, there was a lot of information regarding techniques that deal with implementing artificial neural networks through MATLAB. There are many different toolboxes and resources within MATLAB that were learned that will help with the development of the project as well.