



## SAFETY SHEET 2022-2024

Illinois Junior Academy of Science

**Directions:** The student is asked to read these introductions carefully and fill out the bottom of this sheet. The science teacher and/or advisor must sign in the indicated space. By signing this sheet, the sponsor assumes all responsibilities related to this project.

**Safety and the Student:** Experimentation or design may involve an element of risk or injury to the student, test subjects and to others. Recognition of such hazards and provision for adequate control measures are joint responsibilities of the student and the sponsor. Some of the more common risks encountered in research are those of electrical shock, infection from pathogenic organisms, uncontrolled reactions of incompatible chemicals, eye injury from materials or procedures, and fire in apparatus or work area. Countering these hazards and others with suitable safety practices is an integral part of good scientific research. In the **chart** below, list the principal hazards associated with your project, if any, and what specific precautions you have used as safeguards. Be sure to read the entire section in the *Policy and Procedure Manual of the Illinois Junior Academy of Science* entitled "Safety Guidelines for Experimentation" before completing this form.

Possible hazards	Precautions taken to deal with each hazard
<ol style="list-style-type: none"><li>1. Electrical shock from the operation of a computer/laptop.</li><li>2. Download of malicious software, virus, or malware that damages the computer and/or other related software systems.</li><li>3. Long time computer usage resulting in eye strain</li></ol>	<ol style="list-style-type: none"><li>1. Use caution when handling wires and electrical outlets.</li><li>2. Make sure to use undamaged and properly insulated cables/outlets.</li><li>3. Use credible platforms and sources provided in the procedure when downloading necessary items.</li><li>4. Ensure that proper anti-malware/anti-virus (firewall) systems are enabled on the computer.</li><li>5. Take frequent breaks and limit screen time.</li></ol>

Specific safety practices related to materials requiring endorsement sheets should be detailed on the specific endorsement sheet and not included on this safety sheet.

Please check off any other possible endorsements needed. Include these documents in your paper and on your board.

- ☐ Humans as Test Subjects –for any projects involving humans including survey administration;
- ☐ Microorganisms-for any projects involving bacteria, viruses, yeasts, fungi or protozoa;
- ☐ Non-Human Vertebrates -for any projects involving fish, amphibians, reptiles, birds or mammals;
- ☐ Tissue Culture-for any projects involving growing eukaryotic tissues or cell cultures;
- ☐ Letter from institution where research was done or IJAS SRC, if an exception to the IJAS rules has been granted...

SIGNED

*Delearghya Roy*

*Shweta Tummala*

Student Exhibitor(s)

SIGNED

*Dr. Susan Brontman (2-5-24)*

Sponsor \*

\*As a sponsor, I assume all responsibilities related to this project.

**This Sheet Must Be Typed and** This form **must** be displayed on the front of the exhibitor's display board. It may be reduced to half a sheet of paper 8.5 inches (vertical) X 5.5 inches (horizontal).

# **Comparative Analysis of CNNs for Classifying Alzheimer's Disease**

Experimental Design

Debarghya Roy and Shrihan Tummala

Adlai E. Stevenson High School | Grade 11

Sponsor: Dr. Brontman

---

---

## **Table of Contents**

<b>Acknowledgments.....</b>	<b>1</b>
<b>Purpose.....</b>	<b>2</b>
<b>Hypothesis.....</b>	<b>2</b>
<b>Essential Components.....</b>	<b>3</b>
<b>Review of Literature.....</b>	<b>3</b>
Machine Learning.....	4
Neural Networks.....	5
Loss Functions and Optimizers.....	7
Image Classification CNNs.....	7
ResNet50.....	7
VGG16.....	8
Xception.....	9
DenseNet169.....	10
Alzheimer’s Disease.....	10
Attention Mechanisms.....	12
<b>Materials.....</b>	<b>15</b>
<b>Procedure.....</b>	<b>15</b>
Balancing Data:.....	15
Kaggle Setup:.....	17
Loading the Dataset:.....	18
Running the code:.....	18
<b>Results.....</b>	<b>20</b>
<b>Discussion.....</b>	<b>25</b>
<b>Conclusion.....</b>	<b>26</b>
<b>Reference List.....</b>	<b>29</b>
<b>Appendix A: Code.....</b>	<b>33</b>
Balancing Data Code.....	33
Kaggle Code.....	34

---

### **Acknowledgments**

We would like to extend our heartfelt gratitude to our families for their support in helping us pursue our research in this field. And most importantly, we would like to thank our sponsor Dr. Brontman for investing her time into providing valuable and prompt feedback to help us improve our project. This project would not have been possible without her guidance and assistance.

---

## **Purpose**

The purpose of this experiment is to determine the ability of various Convolutional Neural Networks (CNNs) in detecting the various stages of Alzheimer's disease. This research is relevant since it can reveal the merits and shortcomings of various CNNs in this subset of analyzing Alzheimer's Disease. Alzheimer's disease is the most common type of dementia and is generally found in elderly citizens. As the world's population grows, more and more people start becoming vulnerable to this disease. It starts off with memory loss but can result in the patient losing control of their motor skills. It also results in abnormal protein aggregation in the brain, which is present in Magnetic Resonance Imaging (MRIs), allowing CNNs to classify them. This experiment will highlight the best CNN, thereby progressing future development on Alzheimer's and potentially slowing down its spread through early detection and subsequent preventative measures.

## **Hypothesis**

If the dataset of MRI Segmentation images is fit into a VGG16, ResNet50, DenseNet169, and a Xception convolutional neural network model, then the VGG16 will have highest percent accuracy in detecting the stage of Alzheimer's. This is because the VGG16 architecture's depth, encompassing numerous layers, contributes to its efficacy in computer vision tasks. The multitude of layers allows for hierarchical feature extraction, where the lower layers capture basic image features, and higher layers detect more complex patterns. This depth of analysis increases the network's capability to model intricate relationships within data while reducing overfitting through regularization techniques.

---

## **Essential Components**

**Independent Variable:** The various types of CNNs being compared (VGG16, ResNet50, Xception, DenseNet169) in their ability to classify Alzheimer's disease into Non-Dementia, Very Mild Dementia, Mild Dementia, and Moderate Dementia.

**Dependent Variable:** The percent accuracy of the CNN models' predictions measured after the model is trained and run on test data.

**Constants:** The programming language used (Python 3.10); the computer used; the software the program is run on (Jupyter Notebook); the libraries used (Tensorflow, NumPy, Matplotlib); the 70:20:10 ratio for the training, testing, and validation datasets respectively; the dataset (Balanced Alzheimer MRI Dataset); and the CBAM attention layer for every model.

**Comparison Group:** A Dense-Net 169 classification algorithm that classifies Alzheimer's disease into Non-Dementia, Very Mild Dementia, Mild Dementia, and Moderate Dementia.

## **Review of Literature**

With technology and software improving each year, machine learning has become a prominent aspect of society. This subset of Artificial Intelligence (AI) involves the development of algorithms that allow computers to learn from data and recognize patterns from them in order to make predictions for a set task more efficiently. It has already found its way into various industries and is expanding its usage and capabilities as time passes; more specifically, healthcare, where its applications range from sensors worn by patients to finding trends in patients' data to identify warning signs. (*Machine Learning: What it is and Why it Matters*, n.d). Detecting Alzheimer's early on has become necessary as each year it affects millions of people globally, being the primary cause of dementia. An underlying issue of Alzheimer's is that it often

---

remains undetected until it has progressed significantly, making it challenging to treat effectively (*What is the Difference between Dementia and Alzheimer's Disease?* 2023). If doctors were able to diagnose Alzheimer's before its devastating symptoms struck, it would allow the patient to plan out their future and slow down the growth of the disease through various lifestyle changes. Although there are many machine learning models out there that use various biomarkers such as changes in the Autonomous Nervous System, the retina, or from MRI and PET scans, they do not account for the similarity in symptoms to other diseases like brain stroke. (Vrahatis et al., 2023). Thus, there is a necessity to improve current machine learning algorithms to decrease the likelihood of false positives through the inclusion of attention mechanisms which help models focus on specific areas of the data, in this case, areas affected by Alzheimer's.

### **Machine Learning**

Machine learning allows computers to make predictions on data based on learned experiences, without constant human involvement. Rather, it involves the development of models based on algorithms that analyze data to find trends and patterns and make the aforementioned predictions solely based on the data it is trained on (Tantawi, 2023). It uses an iterative approach in order to continuously improve its understanding of the data until a strong pattern is found that encompasses a vast majority of the data (*Machine Learning: What it is and Why it Matters*, n.d).

Supervised learning is the largest subset of machine learning which involves training algorithms with all labeled examples. The algorithm then makes predictions and compares them to the given outputs to measure its accuracy which is followed by a repeated process where it makes new predictions to improve its accuracy and create a precise model. On the other hand, unsupervised learning is trained only on unlabeled examples where there is no given output, and

---

it must find a pattern based on only the inputs in order to make a conclusion (*Machine Learning: What it is and Why it Matters*, n.d).

Deep learning is another recently emerging subset of machine learning aimed at replicating the human brain's ability to learn (Dunn, 2023). It can process large amounts of unstructured data unlike supervised and unsupervised learning and involves multi-layered neural networks where each layer works to improve the model's ability to make a prediction. Deep learning has shown promising results in automation tasks and making accurate predictions with minimal human intervention and has also found its way into several practical applications such as digital assistants and self-driving cars (*What is Deep Learning?*, n.d.).

### **Neural Networks**

Neural networks are another subset of machine learning that make up the core of deep learning algorithms. They also go by either artificial neural networks (ANNs) or simulated neural networks (SNNs) (*What are Neural Networks?*, n.d.). Similar to deep learning, these networks also gain their name from their attempt at replicating biological neurons in the human brain in areas such as information processing, memory retention, and pattern recognition (Bondarenko, 2023). Similar to some of the other subsets of machine learning, neural networks also rely on training data for improved accuracy (*What are Neural Networks?*, n.d.).

Multi-layer Perceptrons (MLPs) are quite a predominant type of neural network that consists of several layers. The first layer is known as the input layer, the last is the output layer, and all other layers are known as hidden layers. Weights are assigned to nodes when the input layer is first created and help determine the significance of any given variable; the larger the weight, the more influence it has over the output (Hardesty, 2017). As the model progresses in a



linear fashion, in that it only progresses to the next layer from the preceding one, it is also known as a feedforward neural network (*What are Neural Networks?*, n.d.).

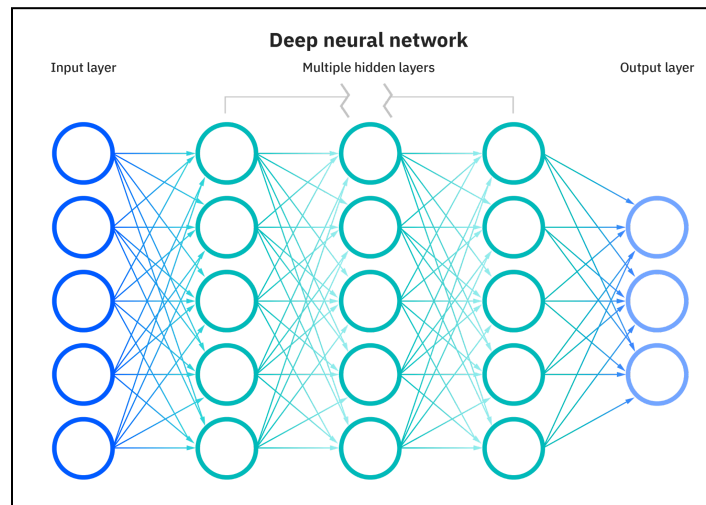


Figure 1. Layers of Artificial Neural Networks (Source: *What are Neural Networks?*, 2023)

These layers contain interconnected nodes known as sigmoid neurons, each with its own assigned weights and thresholds. Each input is multiplied by its associated weights and then added together, creating the output which then goes through the activation function. If a node's output passes a set threshold, the node activates and passes it to the next layer where it becomes the input of a new node. This process, known as the feedforward neural network, is repeated until the output layer is reached (*What are Neural Networks?*, n.d.).

Then there are convolutional neural networks (CNNs) which function similarly to the MLPs but are more targeted towards image and pattern recognition (*What are Neural Networks?*, n.d.). CNNs typically consist of multiple layers, which include the convolutional layer, activation layer (such as ReLU), pooling layer, and the fully connected (FC) layer (Wood, n.d.). CNNs employ kernels, also known as filters, to analyze the input image for patterns like edges, gradients, and shapes (Wood, n.d.). These kernels slide across the image, performing mathematical operations that highlight significant features while preserving spatial information

---

(Chase et al., 2023). By repeatedly applying those operations, CNNs can detect progressively more complex features as the input passes through each layer until the target feature is identified (*What are Convolutional Neural Networks?*, n.d.).

### Loss Functions and Optimizers

A loss function is used while training neural networks to evaluate the difference between the model's output and what the output is expected to be. For situations with more than one class, the Categorical Crossentropy Loss Function is generally used. It outputs a numerical value, which, when closer to 0, signifies that the model is performing well (Shah, 2023).

The optimizer takes the output of the loss function and improves the parameters of the neural network so that it can improve. Adaptive Moment Estimation (Adam) is generally considered one of the best optimizers as it combines the advantages of gradient descent and stochastic gradient descent so that it takes less time to train the model (Patil, 2023).

### **Image Classification CNNs**

#### ResNet50

Although CNNs have demonstrated exceptional performance in extracting features and classifying images, several CNNs were hindered by problems such as the vanishing gradient problem, which occurs when gradients become increasingly small as they go back through the network, leading to inefficient training (Mukherjee, 2022). ResNet50, introduced in 2015, addressed the vanishing gradient problem through the concept of residual connections. These connections directly link layers within the network, allowing the network to learn residual functions. Residual functions introduce shortcut connections between layers of the network

which allow the gradients to flow more easily through the network, making it less likely that they will vanish (Kundu, 2023).

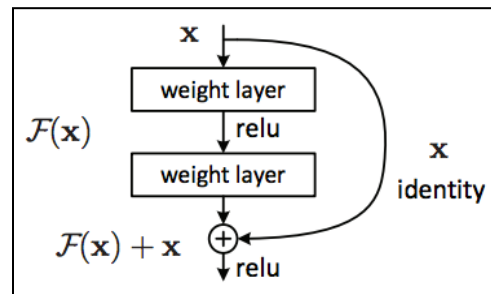


Figure 2. ResNet block: Skip learning (Source: Mukherjee, 2022)

Additionally, skip learning (Figure 2) is utilized to skip over some of the layers in the network that impede the model's performance when calculating the gradients which can also help to reduce the computational cost of training the network (Mukherjee, 2022). Both the residual functions and skip learning have been shown to improve the accuracy of ResNet50 and make it more robust to overfitting.

## VGG16

The first prototype of VGG16 was developed in 2014 by Karen Simonyan & Andrew Zisserman in their paper *VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION*. As the title of the paper indicates, the model focused more on deep layers with multiple small 3x3 convolutional filters, outperforming other models like AlexNet. These smaller filters result in more activation layers, reducing the network's tendency to overfit. (Understanding VGG16: Concepts, Architecture, and Performance, n.d.)

Overall, it is 16 layers deep with more than 138 million parameters. It takes in a 224x224 image as input, runs it through several convolutional and pooling layers, and mainly uses the ReLu activation function. VGG was the first model that introduced the concept of adding layers

to increase accuracy. However, this results in a high training time, even more so considering it lacks attributes such as the skip learning present in ResNet. (*Understanding VGG16: Concepts, Architecture, and Performance*, n.d.)

### Xception

Xception, an evolution in CNN design developed by Google, excels due to its innovative use of Depthwise Separable Convolution. This approach comprises two steps: Depthwise Convolution and Pointwise Convolution (Fabien, 2019). Depthwise Convolution handles spatial information channel-wise, processing each channel independently and significantly reduces computational load by focusing on individual channel spatial operations. Subsequently, Pointwise Convolution condenses information gathered from Depthwise Convolution by employing  $1 \times 1$  convolutions, which reshapes channel dimensions, merging and refining the processed information (Tsang, 2018).

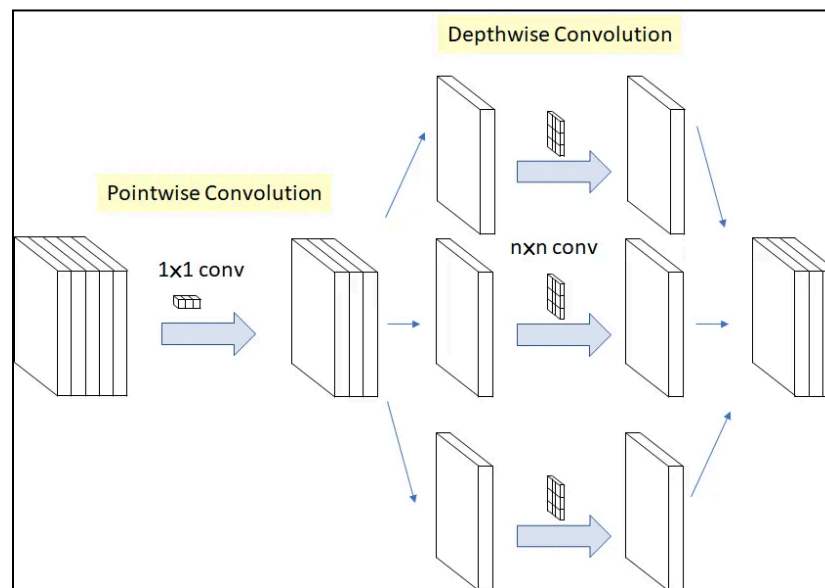


Figure 3. Modified Depthwise Separable Convolution. (Source: Tsang, 2018)

---

The modified Depthwise Separable Convolution in Xception (Figure 3), resembling an 'extreme' Inception module, starts with Pointwise Convolution before Depthwise Convolution. This alteration, inspired by Inception-v3, brings minor operational differences but boosts performance. Xception excludes the activation function ReLU which is generally used to allow neural networks to model complex patterns in data that are not linearly separable by introducing non-linearity. This exclusion enhances computer efficiency as the Depthwise Separable Convolution steps already capture complex patterns without reliance on additional non-linearities without significantly compromising performance. Additionally, Xception also integrates residual connections across its architecture similar to Resnet50, aiding gradient flow and training (Tsang, 2018).

#### DenseNet169

The underlying principle behind DenseNet models is that every single layer is connected to every other layer in front of it. Similar to skip connections in the ResNet Model, this feature ensures feature reusability from earlier layers. This way, valuable information from previous layers is not lost, thus increasing the accuracy of the model and allowing for larger models without fear of feature loss between layers. It also results in fewer parameters overall than traditional CNNs, since the network can be more compact by having fewer channels. (Ruiz, 2018)

#### **Alzheimer's Disease**

Neural networks have recently seen more usage in medical image analysis, more specifically CNNs. Their ability to identify complex patterns in large datasets has paved the way for diagnosing diseases. In the case of Alzheimer's disease, a widespread disease afflicting 35

---

million people worldwide, CNNs offer the potential to advance diagnostic measures.

Alzheimer's leaves its victims incapable of remembering the existence of their closest relatives.

There are currently no known interventions for curing Alzheimer's. However, the early diagnosis of Alzheimer's can allow doctors to prescribe medicine to slow the spread of the disease

significantly. Alzheimer's can also be detected via signature chemicals found in cerebrospinal fluid obtained through lumbar punctures. However, this method is physically invasive and should be avoided if possible, especially if the patient is old. So, researchers have turned their attention

to another key feature that Alzheimer's leaves behind: the growth of abnormal clumps and tangled clumps of fiber known as amyloid plaques and neurofibrillary (tau) tangles respectively

(The Evolving Story of Alzheimer's Disease, 2023). Amyloid plaques accumulate between neurons and disrupt cell function, thus contributing towards cognitive decline. Additionally, the abnormal accumulation of tau protein inside neurons leads to the formation of neurofibrillary tangles which interfere with the neuron's transport system, leading to impeded communication between neurons. Furthermore, when beta-amyloid levels reach a critical threshold, tau protein spreads rapidly throughout the brain, hindering the brain regions involved with memory and eventually leading to significant loss of brain volume and function (*What Happens to the Brain in Alzheimer's Disease?*, 2017). These growths can be identified in fMRI and PET tomography scans, which are relatively less physically invasive methods.

Alzheimer's was first discovered in 1906 by Dr. Alois Alzheimer, who noticed these abnormal growths in the brain of a woman who had passed away from a mental illness. It was then subsequently categorized into several stages of severity: Mild, Moderate, and Severe (*Alzheimer's Disease Fact Sheet*, 2023).

---

In the mild stage of the disease, the patient develops greater short-term memory loss and similar cognitive issues. These symptoms can manifest themselves through the patient getting lost easily and being unable to handle mental calculations that they were previously able to do (*Alzheimer's Disease Fact Sheet*, 2023).

In the moderate stage, the brain starts accumulating damage in some vital areas required for functioning properly. The patient might be unable to execute long tasks, recall familiar sounds and smells, and start experiencing hallucinations. (*Alzheimer's Disease Fact Sheet*, 2023).

In the severe stage, the brain is almost completely taken over by tangles and plaques and atrophies significantly. The patient will only intermittently remember relatives and will be confined to the bed for the rest of their lives (*Alzheimer's Disease Fact Sheet*, 2023).

### **Attention Mechanisms**

Although the symptoms of Alzheimer's are discoverable in MRI scans, there is a critical flaw with using MRI scans of the brain as conclusive evidence for diagnosing Alzheimer's. Many other diseases such as brain stroke and multiple sclerosis also have similar effects on the brain so the model may misdiagnose a patient for Alzheimer's when they actually have some other disease, resulting in a false positive (*Cerebral Atrophy*, n.d.).

In order to minimize errors such as these, an extra layer known as the Attention Layer was incorporated. The Attention Layer can selectively choose which parts of the image the model should emphasize. This layer is significant to this scenario as it ensures that the regions of the brain that the model has largely based its decision on are known to be common regions affected by Alzheimer's Disease. (Hore & Chatterjee, 2023).

Attention mechanisms work by assigning weights to each part of the input, with larger weights for more significant areas. In addition, the model can be discouraged from using regions

not significant to Alzheimer’s Disease by assigning low weights. In Natural Language Processing, for example, words that provide more context should be given more attention. Prodip Hore illustrates this idea with a simple example, ‘Let  $\alpha$  is [0.2, 0.3, 0.3, 0.2] and the input sentence is “I am doing it”. Here, the context vector corresponding to it will be:  $C=0.2*I^{\text{“I”}} + 0.3*I^{\text{“am”}} + 0.3*I^{\text{“doing”}} + 0.3*I^{\text{“it”}}$ .’ The “I” next to every number represents the hidden state of that number (Hore et al., 2023, par. 30-31).

A similar approach was chosen by Ashurov et al. in their publication, *Improved Breast Cancer Classification through Combining Transfer Learning and Attention Mechanism*. They utilized the Convolutional Block Attention Module (CBAM) to improve their accuracy. The CBAM consists of the Channel Attention Module (CAM) and the Spatial Attention Module (SAM) (Figure 4) (Ashurov et al., 2023).

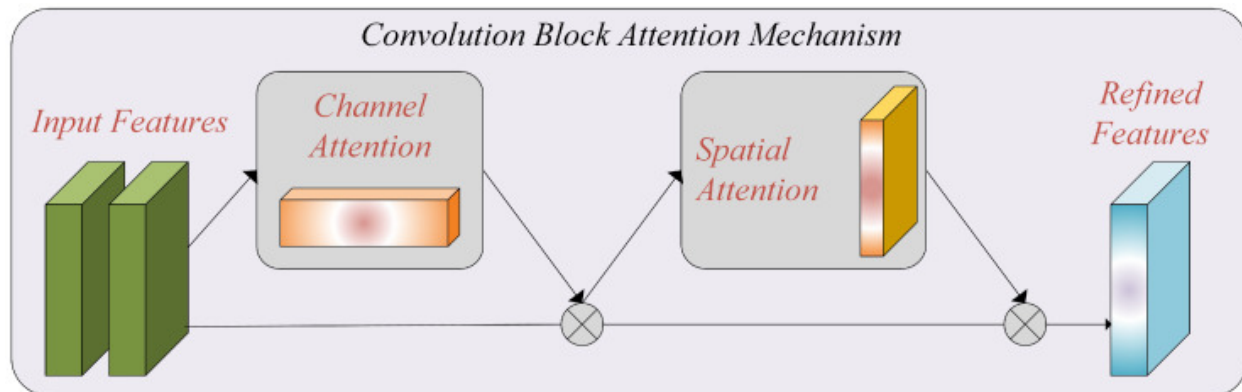


Figure 4. Convolutional Block Attention Module (CBAM) (Source: Ashurov, 2023).

The CAM uses a multilayer perceptron as well as global average and max pooling layers to generate a channel attention map (Figure 5). The maximum pooling layer takes an input array and shrinks the dimensions by selecting the highest values, for example, in every 4x4 square to output a smaller output array. The average pooling layer does the same except it takes the average of the 4x4 square instead (*What are Convolutional Neural Networks?*, n.d.). This part of



the CBAM decides what part of the input is meaningful (Woo et al., 2018).

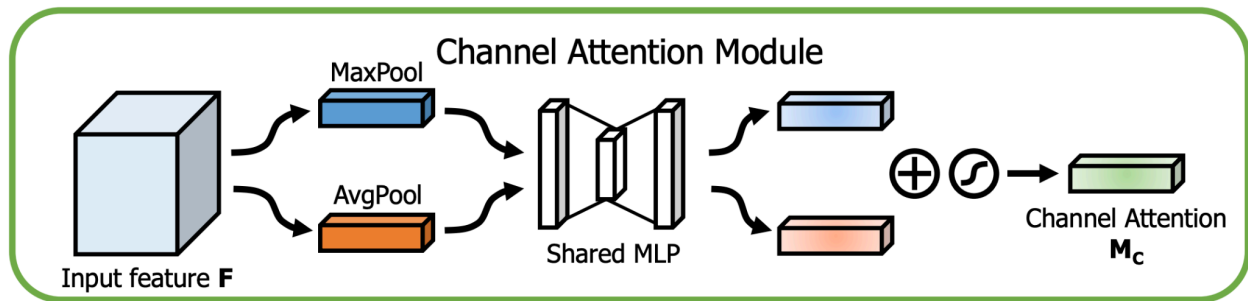


Figure 5. Channel Attention Module (CAM) (Source: Woo et al., 2018).

The SAM generates two feature maps by using global average and max pooling layers and then concatenates, or combines, the feature maps into a spatial attention map (Figure 6). This part of the CBAM helps identify where the input is relevant by processing the results from the CAM. (Woo et al., 2018).

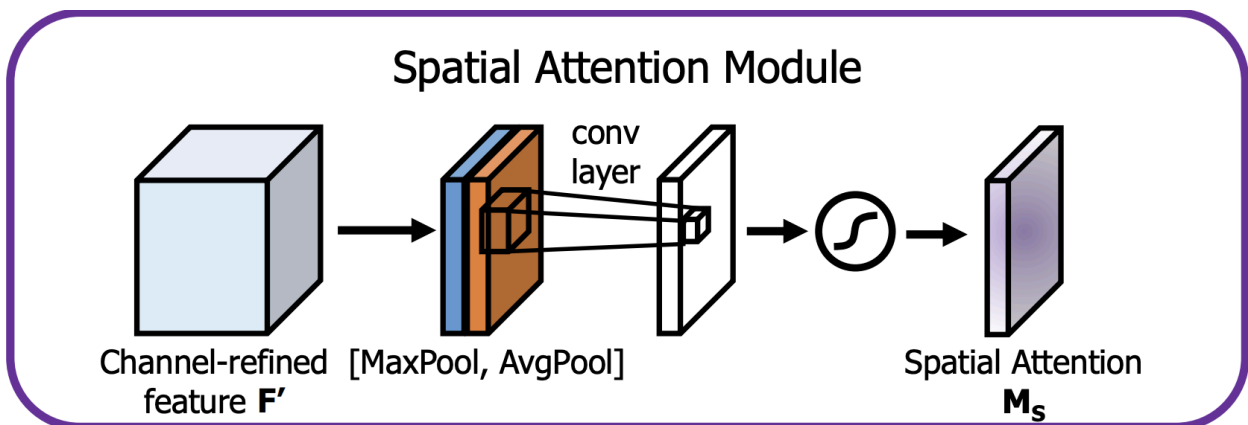


Figure 6. Spatial Attention Module (SAM) (Source: Woo et al., 2018).

To detect the different stages of Alzheimer's disease using MRI images, various CNNs will be tested in order to evaluate their performance. Each CNN model will be trained on a preprocessed dataset of MRI images labeled with the corresponding Alzheimer's disease stage. In addition, to increase the reliability of the results, each CNN will have attention mechanisms incorporated into them. The attention mechanism will assign weights to different parts of the

---

MRI image, emphasizing areas that are more likely to be associated with Alzheimer's disease pathology. The performance of each CNN model will then be assessed using percent accuracy on a test dataset and the results will be compared to identify the most effective CNN for Alzheimer's disease classification.


## **Materials**


- 32-bit/64-bit PC (running macOS Sonoma)
- Access to the internet
- Alzheimer MRI Preprocessed Dataset (Retrieved September 9, 2023).
- A web browser such as Google Chrome (Version 120.0.6099.199) that is compatible with Kaggle
- Visual Studio Code Version 1.85
- ResNet50 Model from TensorFlow Version 2.14.0
- VGG16 Model from TensorFlow Version 2.14.0
- Xception Model from TensorFlow Version 2.14.0
- DenseNet169 Model from TensorFlow Version 2.14.0
- A Kaggle Account with a verified phone number

## **Procedure**

### **Balancing Data:**

- 1) Download Visual Studio Code from <https://code.visualstudio.com/download>
- 2) Press Shift + Command + X to open the extension sidebar
- 3) Search python
- 4) Click on the first extension “Python”

- 
- 5) Click on Install to download the python language 
  - 6) Go to [https://drive.google.com/file/d/1XDxp6Bw320rT2epj6xX\\_NTrcRyILmFya/view?usp=sharing](https://drive.google.com/file/d/1XDxp6Bw320rT2epj6xX_NTrcRyILmFya/view?usp=sharing)
  - 7) Download the code to preprocess the dataset by clicking the download button
  - 8) Open Finder
  - 9) Go to your downloads
  - 10) Right click on the preprocess.py file
  - 11) Hover over the “open with” option
  - 12) Choose Visual Studio Code
  - 13) If Visual Studio Code asks permission to access files in your downloads, click allow
  - 14) Go to <https://www.kaggle.com/datasets/ninadaithal/imagesoasis>
  - 15) Click the download button in the top right side of screen
  - 16) In Finder, open Downloads
  - 17) Double click the “.zip” file named “archive.zip”
  - 18) Press Option + Command + P to show the path bar on the bottom of the finder window
  - 19) With the Data folder selected (highlighted in blue), right click the “Data” button on the path bar
  - 20) Click “Copy Data as Pathname”
  - 21) Go back to Visual Studio Code
  - 22) Press Command + Shift + P
  - 23) Type “terminal”
  - 24) Select the “Python: Create Terminal” option



- 
- 25) In the terminal that appears on the bottom, type “pip install opencv-python”
  - 26) After the command executes, type “pip install albumentations”
  - 27) Click the run button  at the top right of the VS Code Window
  - 28) When prompted in the terminal to enter the path for the dataset, press Command + V
  - 29) Press Enter to make the dataset balanced by augmenting images for the Moderate Dementia class (488 Images → 4880 Images) and removing excess images for the Non Demented (67,222 Images → 5000 Images), Mild Dementia (5002 Images → 5000 Images), and Very Mild Dementia (13,725 Images → 5000 Images) classes.
  - 30) In Finder, open Downloads
  - 31) Right click the “Data” folder that has just been preprocessed
  - 32) Click ‘Compress “Data”’ to make a zipped version of the balanced dataset

### **Kaggle Setup:**

- 33) Register an account on Kaggle:  
<https://www.kaggle.com/account/login?phase=startRegisterTab&returnUrl=%2F>
- 34) Click on your profile picture in the top right corner
- 35) Click on “Settings”
- 36) Using the “Phone Verify” button, verify your phone number to be able to use an accelerator
- 37) Click the following link to open the notebook:  
<https://www.kaggle.com/code/debarhyaroy11707/roy-tummala-23-24-alzheimer-s-classification?scriptVersionId=158232378>
- 38) Click the “Copy and Edit” button in the top right corner of the screen


- 
- 39) Scroll down in the “Notebook” tab on the right side until you find a “Notebook Options” dropdown
  - 40) Open the “Notebook Options” dropdown
  - 41) Choose the GPU T4 x2 accelerator in the “Accelerator” dropdown

**Loading the Dataset:**

- 42) In the “Notebook” tab, click the upload data button next to the + Add Data button 
- 43) Click Browse Files
- 44) Go to your Downloads folder
- 45) Select the zipped Data.zip file
- 46) Click Open
- 47) Fill in a title for the dataset
- 48) Click Create
- 49) Once the data has been uploaded, click on the dataset in the sidebar to show the “Data” folder
- 50) Hover over the the Data folder to see a copy button  to the right of it
- 51) Click the copy button to copy the path for that directory
- 52) Go to cell 2 in Kaggle
- 53) On the second line, select the text inside the quotes
- 54) Press Command + V

**Running the code:**

- 55) Click on cell 1

- 
- 56) Click the Run button  to import the matplotlib library for creating graphs, and several modules in the TensorFlow library including the CNN models (ResNet50, VGG16, Xception, DenseNet169)
  - 57) Click on cell 2
  - 58) Click the Run button to load the dataset and split it into 70% for training, 20% for testing, and 10% for validation
  - 59) Click on cell 3
  - 60) Click the Run button to create several functions for the CBAM attention mechanism
  - 61) Click on cell 4
  - 62) Click the Run button to merge all the layers to be added after the pre trained model
  - 63) Click on cell 5
  - 64) Click the Run button to create a function that can graph the training and validation accuracies and losses
  - 65) Click on cell 6
  - 66) Click the Run button to create a function that will test the model and output a test accuracy
  - 67) Click on cell 7
  - 68) Click the Run button to create the early stopping mechanism
  - 69) Click on cell 8
  - 70) Click the Run button to download the weights for the ResNet50 model and train it
  - 71) Click on cell 9
  - 72) Click the Run button to test the ResNet50 model on the testing dataset
  - 73) Record the test accuracy for the ResNet50 model

---

74) Click on cell 10

75) Click the Run button to produce the graphs for the training and validation accuracies and losses for the ResNet50 model

76) Repeat steps 68-74 for cells 11-13, 14-16, and 17-19 to train and evaluate the DenseNet169, VGG16, and Xception models respectively

## **Results**

Each CNN model underwent iterative epochs, each epoch signifying complete runs through the training dataset. Additionally, all CNN models were evaluated on the same ratio of data splits in a dataset of 19880 images: 3976 samples for testing, 1988 samples for validation, and 13916 samples for training. The validation dataset was used while training the model after every epoch to evaluate its performance on images that it had not previously seen before. The testing dataset was used for gathering the final data points.

The line graphs below reveal the growth of the model as it trains on the training dataset. The line graphs on the left side consist of the accuracies of each of the models at each epoch, while the graphs on the right side demonstrate the loss value of each model at each epoch. The loss value measures how much a model's predictions differ from the correct classifications. A lower loss indicates the model is making accurate predictions more often.

Figure 7: Accuracy vs. Epochs for ResNet50

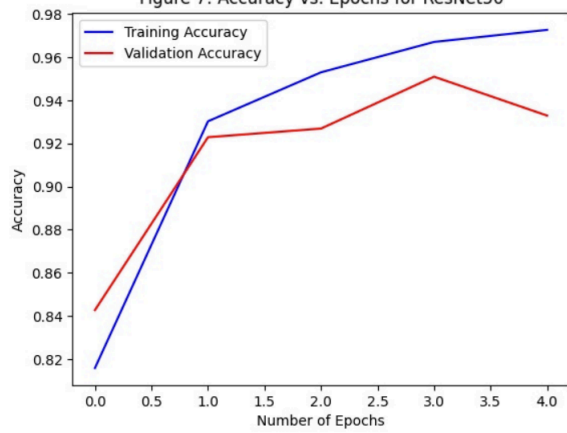


Figure 8: Loss vs. Epochs for ResNet50

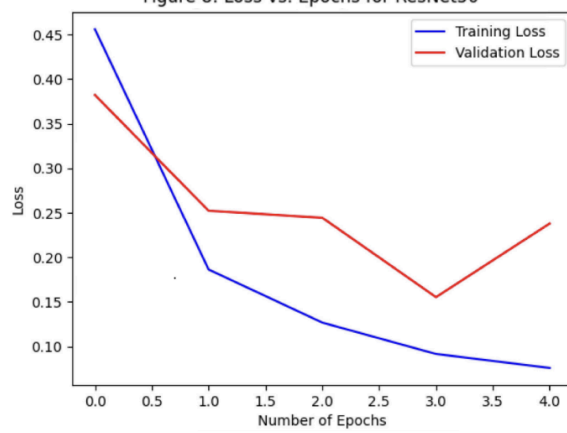


Figure 9: Accuracy vs. Epochs for DenseNet169

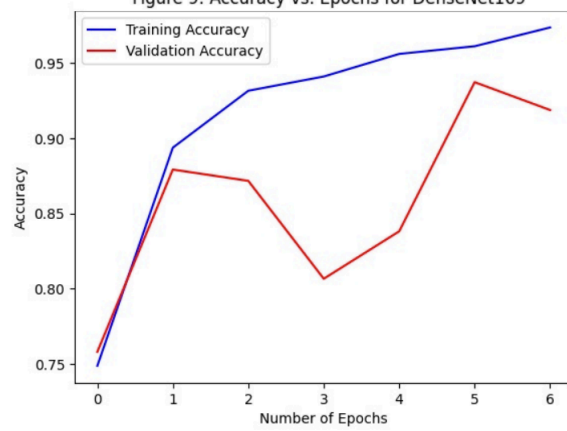


Figure 10: Loss vs. Epochs for DenseNet169

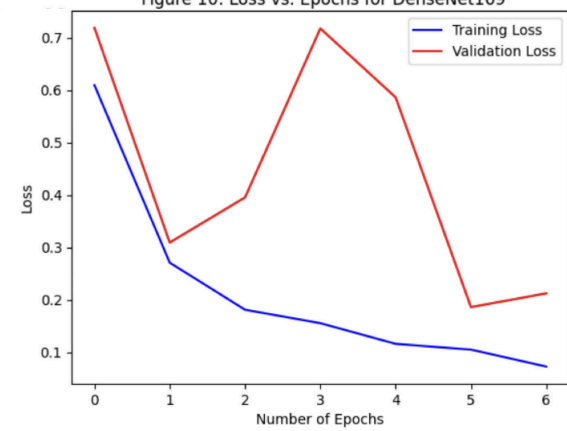


Figure 11: Accuracy vs. Epochs for VGG16

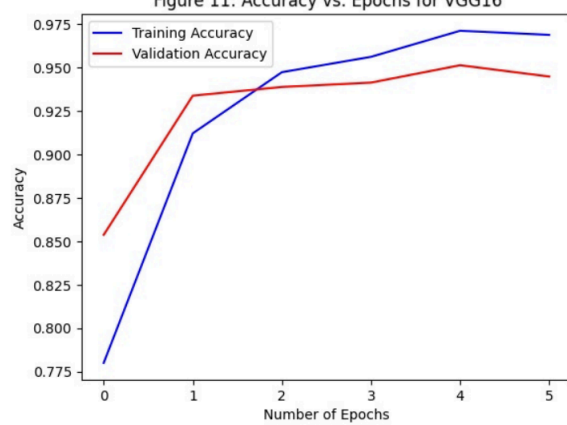
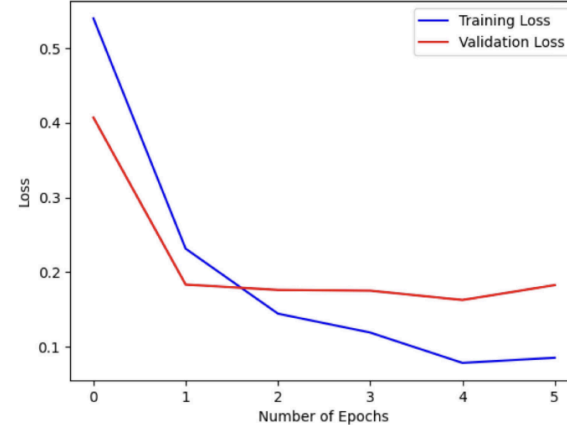
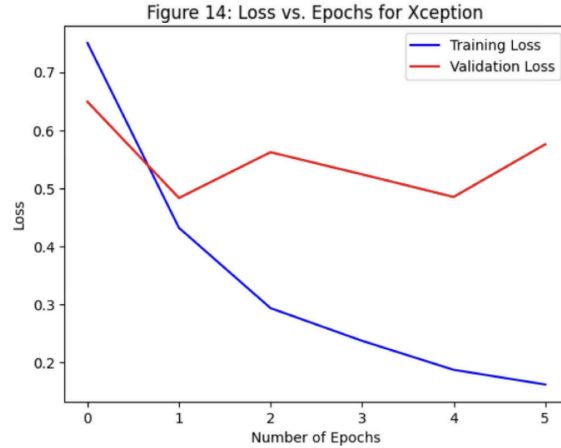
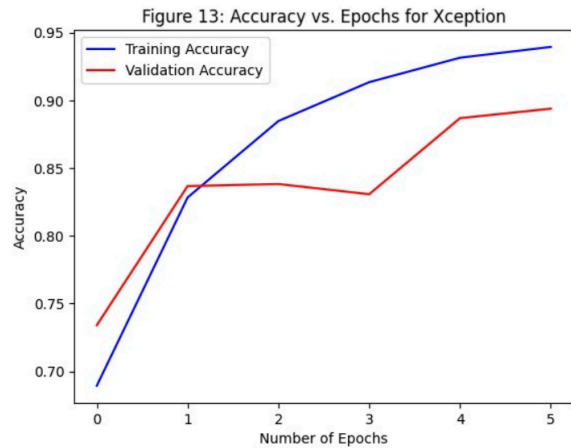


Figure 12: Loss vs. Epochs for VGG16







All the models achieved significant progress in the first epoch. Generally, the training accuracy appears to be higher than the validation accuracy and the training loss appears to be lower than the validation loss, especially after the first epoch. The training and validation lines being relatively similar indicates that the model is not overfitting too much. Overfitting occurs when a model increases its accuracy through memorization of the training data or luck rather than recognizing important patterns for differentiating the different stages of Alzheimer's. One technique for preventing overfitting includes augmenting the data, which is further explained in Discussion. The validation data is therefore more accurate as it is only used for evaluating the models performance and is new to the model.

Figure 15: Optimal Number of Epochs vs. Type of Convolutional Neural Network				
	Convolutional Neural Network			
	VGG16	ResNet50	DenseNet169	Xception
Optimal # of Epochs	5	4	6	5

Another precaution taken to prevent overfitting includes the implementation of an early stopping mechanism into each model which tracks the validation loss throughout the training

process, and forces the model to stop training as soon as it detects a spike in the validation loss when compared to the previous epoch. As previously mentioned, the validation loss is a reliable metric to track since its increase helps indicate that the CNN is learning too much from the training data to the point where it loses its ability to adapt to new data—necessitating a stop.

Figure 16: Accuracy in Identifying Alzheimer's vs. Type of Convolutional Neural Network				
	Convolutional Neural Network			
	VGG16	ResNet50	DenseNet169	Xception
Accuracy (Percent)	94.83	93.60	92.72	90.45

Figure 16 shows the CNN's accuracy on the testing dataset following the completion of training and validation. The testing phase only represents a singular trial as it is a culmination of the network's learning from the training process when it went through the training and validation datasets. Similar to the validation dataset, this phase assesses CNN's ability to handle unseen data.

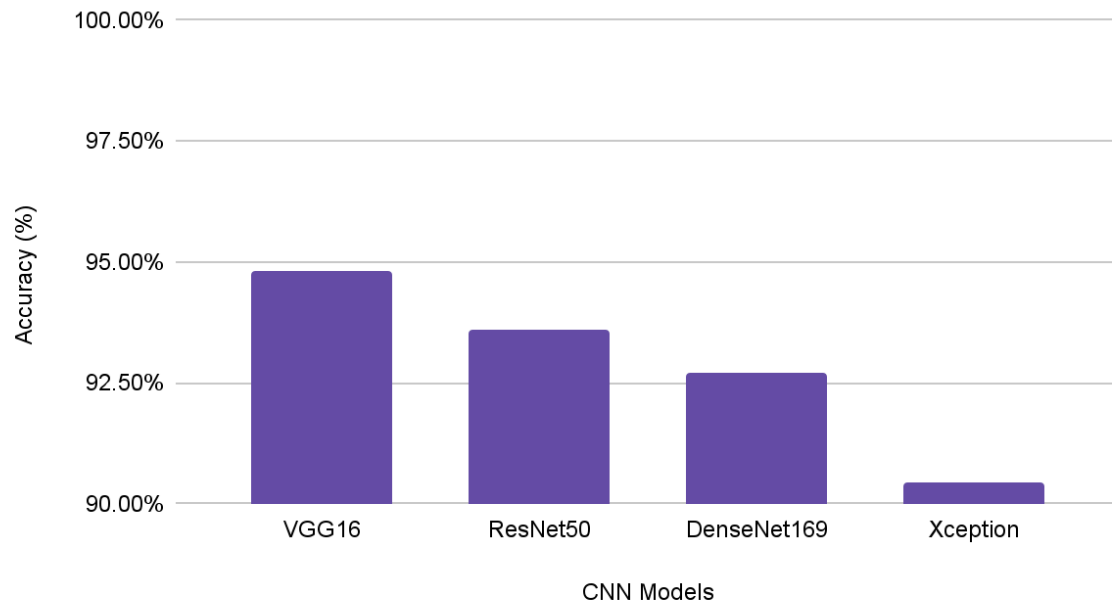
**Figure 17: Accuracy vs. CNN Models**

Figure 17 visualizes the data in Figure 16 as a bar graph. It can be seen that the CNN models exhibited a high level of accuracy overall, with all the models achieving an accuracy above 90%. However, there was some variation between the models due to their different architectures. VGG16 demonstrated the highest accuracy, reaching 94.83%. It was closely followed by ResNet50 and DenseNet169, which achieved accuracies of 93.60% and 92.72% respectively. All three of the models mentioned above had their accuracies differ only by about 1%, suggesting their proficiency in detecting the stages of Alzheimer's. Meanwhile, Xception, while still exhibiting strong performance, still fell behind with an accuracy of 90.45%, a difference of more than 2% in comparison to the previous model.

---

## **Discussion**

In some cases, although there was a sudden spike in the validation loss in the first few epochs, it continued to decrease soon after, showing that the model still had potential to improve. The phenomenon described above was especially prominent in Figure 10 and Figure 14 which correspond to the DenseNet169 and Xception models. To counter this, the “start\_from\_epoch” parameter was set to 3 to guarantee that all the models would pass 3 epochs before the early stopping mechanism activated.

The original dataset from Kaggle consisted of a heavy imbalance, with the largest class containing 77.77% of the images while the smallest class only contained 0.56% of the images. The initial presence of dataset imbalance posed challenges to the models where there were some classes that were far more represented than others. This could lead to the model favoring the majority class, which can result in the model being less effective in recognizing and predicting the underrepresented class, therefore impacting its overall performance and its reliability. If the data was left in the aforementioned imbalance state, and predicted all values to be off the majority class, its accuracy would've still been 77.77%. This outcome highlights a critical issue surrounding accuracy as a metric in cases of severe class imbalance. To combat the issue, all the classes were standardized to a size of 5000. For the large classes, 5000 images were chosen at random, and the small class was increased through data augmentation techniques. The 488 images were randomly rotated, flipped, and other augmentations 9 times each leading to a class of size 4880.

---

## **Conclusion**

The purpose of this experiment was to identify which CNN model would perform the best in classifying the different stages of Alzheimer's based on an MRI. As the life expectancy of humans increases, there has also been a corresponding increase in seniors aged 65+. The Alzheimer's Disease International Organization predicts that the number of people worldwide suffering from dementia is expected to approximately double every 20 years, going from 55 million in 2020 to 139 million in 2050 (*Alzheimer's Disease International - dementia statistics*, n.d.). As such, due to the detrimental impact Alzheimer's poses to the geriatric population, identifying the disease accurately has become a more pressing priority. The early detection of Alzheimer's is also crucial in prompting intervention that can slow the disease from progressing.

The hypothesis was that the VGG16 would perform with the highest accuracy out of the four CNNs due to its complex layers and extensive feature extraction which allows the network to discern complex patterns while mitigating overfitting through regularization techniques. Although this prediction is supported by Figure 16, as VGG16 achieved the highest accuracy of 94.83%, it is important to acknowledge that every single algorithm achieved accuracy above 90%, reinforcing their reliability in identifying Alzheimer's. Additionally, in Figure 12, the training and validation loss curves for VGG16 are closer together than those for ResNet50 in Figure 8. This suggests that VGG16 may be less prone to overfitting, as a large gap between the training and validation loss curves is generally an indicator of overfitting.

An experimental error that could potentially impact the data involves the approach of replicating the Moderate Dementia class several times with minor modifications to augment the data. While the technique helped artificially increase the size of the class in order to balance it with the other classes, it raises the possibility that the model could struggle to recognize

---

complexities in more diverse datasets in the real world due to the limited variety of images it was trained on. In future experiments, this source of error could be eliminated by more thorough data augmentation techniques or through another source of MRI images for Moderate Dementia that could be added onto the original dataset.

DenseNet169's success was quite unexpected as it is a fairly simple and straightforward model because its architecture only combines features from all preceding layers using dense blocks to find patterns. This type of architecture results in it having comparatively fewer parameters, with 14.3 million parameters total. When compared with Xception, it has 8.6 million less parameters, which signifies less weights that can be learnt during training. Thus, it was surprising to see that it outperformed Xception by more than 2%. Moreover, although VGG16 (138.4 million parameters) had more than 5 times the parameters than ResNet50 (25.6 million parameters) they both performed nearly the same. The comparisons above indicate parameters are not proportional to accuracy which is contrary to logic as generally, more parameters means more weights where it can learn patterns and store them in the convolutional layers.

This experiment's real-world application involves the potential for enhancement on Alzheimer's diagnosis accuracy. There have already been various studies, among which include one from Massachusetts General Hospital, where researchers attempted to develop an accurate method of detecting Alzheimer's through deep learning. The increased relevancy of AI in the real world has led to an explosive growth in AI based studies in almost every single field that is compatible with it. While each individual study may contribute to notable improvements, the cumulative effect of refining the diagnostic approaches taken to identify Alzheimer's through AI has the potential to save countless lives. Although the 1% or so differences in accuracies between

---

the models may seem small, when scaled to the over 55 million people suffering from Alzheimer's in the world, it could result in over half a million misclassifications.

Further research can be conducted to this experiment by integrating all of our 4 CNNs into what is known as an ensemble model. This technique has the potential of improving the accuracy further by combining the four CNNs into one model in the prediction stage for the testing datasets. Ensemble models can yield improved results as this approach capitalizes on the strengths of individual algorithms to enhance the collective performance. For instance, combining parts of a CNN adept at identifying early-stage Alzheimer's symptoms with another that is more proficient at detecting more moderate and later stages could lead to the algorithms complementing each other's weaknesses to create a more accurate diagnosis. In addition, a larger and more diverse dataset that includes statistics like the age of the patient would improve the accuracy of the model in the real world. Finally, an attempt could be made to visualize the areas of the MRI that the CNN focuses on through the implementation of attention masks. This visualization could help doctors learn new patterns for identifying Alzheimer's and also aid in directing the CNN to focus on specific regions of the brain associated with Alzheimer's.

---

## Reference List

- Alzheimer's Disease International - dementia statistics*. Alzheimer's Disease International (ADI). (n.d.). <https://www.alzint.org/about/dementia-facts-figures/dementia-statistics/>
- Ashurov, A., Chelloug, S. A., & Tselykh, A. (2023, September 21). *Improved Breast Cancer Classification through Combining Transfer Learning and Attention Mechanism*. NCBI. Retrieved October 8, 2023, from <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10532552/#B29-life-13-01945>
- Bondarenko, V. E. (2022). *Artificial Neural Networks*. EBSCO. Retrieved October 7, 2023, from <https://research.ebsco.com/c/aovkm6/viewer/html/gnm2wr3x3v>
- Chase, B. (2023, March 3). *Using AI to target Alzheimer's — Harvard Gazette*. Harvard Gazette. Retrieved January 14, 2024, from <https://news.harvard.edu/gazette/story/2023/03/using-ai-to-target-alzheimers/>
- Chase, R. J., Harrison, D. R., Lackmann, G. M., & McGovern, A. (2023, August). *A Machine Learning Tutorial for Operational Meteorology. Part II: Neural Networks and Deep Learning*. EBSCO. Retrieved October 8, 2023, from <https://research.ebsco.com/c/aovkm6/viewer/pdf/iu647jnekr>
- Dunn, T. (2023). *Deep Learning*. EBSCO. Retrieved October 7, 2023, from <https://research.ebsco.com/c/aovkm6/viewer/html/2is7nzmt6z>
- Fabien, M. (n.d.). XCEPTION model and depthwise separable convolutions. <https://maelfabien.github.io/deeplearning/xception/>
- Hore, P., & Chatterjee, S. (2023, July 11). *A comprehensive guide to attention mechanism in deep learning for everyone*. Analytics Vidhya. Retrieved October 7, 2023. <https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanis>



---

[m-deep-learning/](#)

Kundu, N. (2023, January 23). Exploring Resnet50: An in-depth look at the model architecture and code implementation. Medium. <https://medium.com/@nitishkundu1993/exploring-resnet50-an-in-depth-look-at-the-model-architecture-and-code-implementation-d8d8fa67e46f>

Larry Hardesty. (2017, April 14). *Explained: Neural networks*. MIT News | Massachusetts Institute of Technology. Retrieved October 7, 2023.

<https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414>

*Machine learning: What it is and why it matters*. SAS. (n.d.). Retrieved October 7, 2023.

[https://www.sas.com/en\\_us/insights/analytics/machine-learning.html#machine-learning-Workings](https://www.sas.com/en_us/insights/analytics/machine-learning.html#machine-learning-Workings)

Mukherjee, S. (2022, August 18). The annotated resnet-50. Medium.

<https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758>

Patil, S. (2023, January 30). *Loss functions and optimizers in ML models*. Medium.

<https://medium.com/geekculture/loss-functions-and-optimizers-in-ml-models-b125871ff0dc>

Ruiz, P. (2018, October 10). *Understanding and visualizing DenseNets* | by Pablo Ruiz. Towards Data Science. Retrieved November 25, 2023, from

<https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a>

Shah, D. (2023, January 26). *Cross entropy loss: Intro, applications, code*. V7.

<https://www.v7labs.com/blog/cross-entropy-loss-guide>

Tantawi, R. (2023). *Machine learning*. Retrieved October 7, 2023.

<https://research.ebsco.com/c/aovkm6/viewer/html/isujuwgvbr>

---

*The evolving story of Alzheimer's disease.* (2023, July). Retrieved October 7, 2023.

<https://research.ebsco.com/c/aovkm6/viewer/pdf/lwooh3kuub>

Tomar, N. (2021, January). *Attention Mechanism Implementation*. Github. Retrieved January 4, 2024, from

<https://github.com/nikhilroxtomar/Attention-Mechanism-Implementation/blob/main/TensorFlow/cbam.py>

Tsang, S.-H. (2018, September 25). Review: Xception - with depthwise separable convolution, better than inception-V3. Medium. <https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568>

U.S. Department of Health and Human Services. (2023, April 5). *Alzheimer's disease fact sheet*. National Institute on Aging. Retrieved October 7, 2023.

<https://www.nia.nih.gov/health/alzheimers-disease-fact-sheet>

U.S. Department of Health and Human Services. (n.d.). *Cerebral atrophy*. National Institute of Neurological Disorders and Stroke. Retrieved October 8, 2023.

<https://www.ninds.nih.gov/health-information/disorders/cerebral-atrophy>

U.S. Department of Health and Human Services. (2017, May 16). *What happens to the brain in Alzheimer's disease?* National Institute on Aging. [https://www.nia.nih.gov/health/](https://www.nia.nih.gov/health/what-happens-brain-alzheimers-disease)

[what-happens-brain-alzheimers-disease](https://www.nia.nih.gov/health/what-happens-brain-alzheimers-disease)

*Understanding VGG16: Concepts, Architecture, and Performance.* (n.d.). Datagen. Retrieved November 25, 2023, from <https://datagen.tech/guides/computer-vision/vgg16/>

Vrahatis, A. G., Skolariki, K., Krokidis, M. G., Lazaros, K., Exarchos, T. P., & Vlamos, P. (2023, April 22). *Revolutionizing the early detection of Alzheimer's disease through*

---

*non-invasive biomarkers: The role of Artificial Intelligence and deep learning.*

*Sensors (Basel, Switzerland).* <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10180573/>

*What are convolutional neural networks?* IBM. (n.d.) Retrieved October 8, 2023.

<https://www.ibm.com/topics/convolutional-neural-networks>

*What is deep learning?* IBM. (n.d.). Retrieved October 7, 2023.

<https://www.ibm.com/topics/deep-learning>

*What are neural networks?* IBM. (n.d.) Retrieved October 7, 2023.

<https://www.ibm.com/topics/neural-networks>

*What is the difference between dementia and Alzheimer's disease?* Alzheimer's Society. (2023, February 21). Retrieved October 7, 2023. [https://www.alzheimers.org.uk/blog/](https://www.alzheimers.org.uk/blog/difference-between-dementia-alzheimers-disease)

[difference-between-dementia-alzheimers-disease](https://www.alzheimers.org.uk/blog/difference-between-dementia-alzheimers-disease)

Woo, S., Park, J., Lee, J.-Y., & Kweon, I. S. (1970a, January 1). *CBAM: Convolutional Block Attention Module*. CVF Open Access. Retrieved October 7, 2023. [https://openaccess](https://openaccess.thecvf.com/content_ECCV_2018/html/Sanghyun_Woo_Convolutional_Block_Attention_ECCV_2018_paper.html)

[.thecvf.com/content\\_ECCV\\_2018/html/Sanghyun\\_Woo\\_Convolutional\\_Block\\_Attention\\_ECCV\\_2018\\_paper.html](https://openaccess.thecvf.com/content_ECCV_2018/html/Sanghyun_Woo_Convolutional_Block_Attention_ECCV_2018_paper.html)

Wood, T. (2019, May 17). *Convolutional Neural Network*. DeepAI. Retrieved October 7, 2023.

<https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network>

## Appendix A: Code

### Balancing Data Code

Python

```
import os
import albumentations as A
import cv2

def augmentImage(path):
    image = cv2.imread(path)
    transform = A.Compose([
        A.CLAHE(),
        A.Transpose(),
        A.HorizontalFlip(p=.5),
        A.Rotate(limit=30, p=.25),
        A.RandomBrightnessContrast(p=.5),
        A.RandomGamma(p=.5),
    ])
    augmented_image = transform(image=image)['image']
    return augmented_image

origPath = input("Enter the path for the dataset folder containing the 4 folders: ")
for path, dirs, files in os.walk(origPath + "/Moderate Dementia"):
    i=0
    for file in files:
        i+=1
        if i < 5000:
            filePath = os.path.join(path, file)
            for j in range(9):
                cv2.imwrite(os.path.join(origPath, 'aug_img' + str(i) + " " + str(j) + ".jpg"),
                    augmentImage(filePath))
for path, dirs, files in os.walk(origPath + "/Non Demented"):
    i=0
    for file in files:
        i+=1
        if i > 5000:
            filePath = os.path.join(path, file)
            os.remove(filePath)
for path, dirs, files in os.walk(origPath + "/Mild Dementia"):
    i=0
    for file in files:
        i+=1
        if i < 5000:
            filePath = os.path.join(path, file)
            os.remove(filePath)
for path, dirs, files in os.walk(origPath + "/Very mild Dementia"):
    i=0
    for file in files:
        i+=1
        if i < 5000:
            filePath = os.path.join(path, file)
            os.remove(filePath)
```

---

## Kaggle Code

Python

```
# import necessary libraries
import matplotlib.pyplot as plt

# import CNNs from tensorflow
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.applications.densenet import DenseNet169

# import other tensorflow modules
from tensorflow.keras.layers import Dense, Flatten, InputLayer, MaxPool2D, Conv2D,
BatchNormalization, ReLU
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import image_dataset_from_directory

# import modules for CBAM
import tensorflow as tf
from tensorflow.keras.layers import GlobalAveragePooling2D, GlobalMaxPooling2D, Reshape,
Input
from tensorflow.keras.layers import Activation, Concatenate, Multiply

# Split the data for the 4 classes with 70% going into training, 20% going into testing,
and 10% for validation
dataset_path = "/kaggle/input/data01/Data copy"
training = image_dataset_from_directory(dataset_path,
    labels='inferred',
    label_mode='categorical',
    class_names=['Very mild Dementia', 'Non Demented', 'Moderate Dementia', 'Mild Dementia'],
    color_mode='rgb',
    batch_size=16,
    image_size=(224, 224),
    shuffle=True,
    seed=42,
    validation_split=0.3,
    subset='training',
    interpolation='bilinear',
    follow_links=False,
    crop_to_aspect_ratio=False,
)
validation = image_dataset_from_directory(dataset_path,
    labels='inferred',
    label_mode='categorical',
    class_names=['Very mild Dementia', 'Non Demented', 'Moderate Dementia', 'Mild Dementia'],
    color_mode='rgb',
    batch_size=16,
    image_size=(224, 224),
    shuffle=True,
    seed=42,
    validation_split=0.3,
    subset='validation',
    interpolation='bilinear',
    follow_links=False,
    crop_to_aspect_ratio=False,
```

```

)
size = tf.data.experimental.cardinality(validation)
testing = validation.take((2*size) // 3)
validation = validation.skip((2*size) // 3)

"""
Implementation of CBAM: Convolutional Block Attention Module in the TensorFlow 2.5. by
nikhilroxtomar
Github:
https://github.com/nikhilroxtomar/Attention-Mechanism-Implementation/blob/main/TensorFlow/
cbam.py
Paper: https://arxiv.org/pdf/1807.06521
"""

def channel_attention_module(x, ratio=8):
    batch, _, _, channel = x.shape

    ## Shared layers
    l1 = Dense(channel//ratio, activation="relu", use_bias=False)
    l2 = Dense(channel, use_bias=False)

    ## Global Average Pooling
    x1 = GlobalAveragePooling2D()(x)
    x1 = l1(x1)
    x1 = l2(x1)

    ## Global Max Pooling
    x2 = GlobalMaxPooling2D()(x)
    x2 = l1(x2)
    x2 = l2(x2)

    ## Add both the features and pass through sigmoid
    feats = x1 + x2
    feats = Activation("sigmoid")(feats)
    feats = Multiply()([x, feats])

    return feats

def spatial_attention_module(x):
    ## Average Pooling
    x1 = tf.reduce_mean(x, axis=-1)
    x1 = tf.expand_dims(x1, axis=-1)

    ## Max Pooling
    x2 = tf.reduce_max(x, axis=-1)
    x2 = tf.expand_dims(x2, axis=-1)

    ## Concatenat both the features
    feats = Concatenate()([x1, x2])

    ## Conv layer
    feats = Conv2D(1, kernel_size=7, padding="same", activation="sigmoid")(feats)
    feats = Multiply()([x, feats])

    return feats

def cbam(x):
    x = channel_attention_module(x)
    x = spatial_attention_module(x)

```

```

    return x

# Concatenates several layers onto the pre-trained model including the attention mechanism
def seqLayers(x):
    x = cbam(x)
    x = Conv2D(filters=224, kernel_size=2, input_shape=(224, 224, 3))(x)
    x = ReLU()(x)
    x = MaxPool2D(pool_size=(2, 2), padding='same')(x)

    x = cbam(x)
    x = Conv2D(filters=224, kernel_size=2)(x)
    x = ReLU()(x)
    x = MaxPool2D(pool_size=(2, 2), padding='same')(x)

    x = Flatten()(x)
    x = BatchNormalization()(x)
    x = Dense(224, activation='relu')(x)
    x = Dense(4, activation='softmax')(x)
    return x

# Plots two graphs for the model given, one for the Validation and Training Accuracies,
# and one for the Validation and Training Losses
def graphEpochs(results, model_name, fig_num):
    plt.plot(results.history['accuracy'], color='b', label="Training Accuracy")
    plt.plot(results.history['val_accuracy'], color='r', label="Validation Accuracy")
    plt.title("Figure " + fig_num + ": Accuracy vs. Epochs for " + model_name)
    plt.xlabel("Number of Epochs")
    plt.ylabel("Accuracy")
    plt.legend()
    plt.show()
    plt.plot(results.history['loss'], color='b', label="Training Loss")
    plt.plot(results.history['val_loss'], color='r', label="Validation Loss")
    plt.title("Figure " + str(int(fig_num) + 1) + ": Loss vs. Epochs for " + model_name + "
model")
    plt.xlabel("Number of Epochs")
    plt.ylabel("Loss")
    plt.legend()
    plt.show()

# Tests the model given on the training data and prints the Accuracy and F1_Score
def testModel(model):
    test_results = model.evaluate(testing)
    print("\nAccuracy: {:.4f}".format(test_results[1]))

# Initialize an early stopping function to stop training once val_loss stops decreasing
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, start_from_epoch=3)

# Setting up ResNet50 Model
resNet = ResNet50(input_shape=(224, 224, 3), weights = "imagenet", include_top = False)
for layers in resNet.layers:
    layers.trainable=False
input_tensor = Input(shape = (224, 224, 3))
x = resNet(input_tensor)
output = seqLayers(x)
model = Model(inputs = input_tensor, outputs = output)

# Training ResNet50

```

```
model.compile(optimizer=Adam(lr=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
resNetResults = model.fit(training, validation_data = validation, epochs=1000, callbacks=[es])
testModel(model)
graphEpochs(resNetResults, "ResNet50", "8")

# Setting up DenseNet169 Model
denseNet = DenseNet169(
    include_top=False,
    weights='imagenet',
    input_shape=(224, 224, 3)
)
for layers in denseNet.layers:
    layers.trainable=False
input_tensor = Input(shape = (224, 224, 3))
x = denseNet(input_tensor)
output = seqLayers(x)
model = Model(inputs = input_tensor, outputs = output)

# Training DenseNet169
model.compile(optimizer=Adam(lr=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
denseNetResults = model.fit(training, validation_data = validation, epochs=1000,
callbacks=[es])
testModel(model)
graphEpochs(denseNetResults, "DenseNet169", "10")

# Setting up VGG16 Model
vgg = VGG16(
    include_top=False,
    weights='imagenet',
    input_shape=(224, 224, 3),
)
for layers in vgg.layers:
    layers.trainable=False
input_tensor = Input(shape = (224, 224, 3))
x = vgg(input_tensor)
output = seqLayers(x)
model = Model(inputs = input_tensor, outputs = output)

# Training VGG16
model.compile(optimizer=Adam(lr=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
vgg_results = model.fit(training, validation_data = validation, epochs=1000, callbacks=[es])
testModel(model)
graphEpochs(vgg_results, "VGG16", "12")

# Setting up Xception Model
xception = Xception(
    include_top=False,
    weights='imagenet',
    input_shape=(224, 224, 3),
)
for layers in xception.layers:
    layers.trainable=False
input_tensor = Input(shape = (224, 224, 3))
x = xception(input_tensor)
```



---

```
output = seqLayers(x)
model = Model(inputs = input_tensor, outputs = output)

# Training Xception
model.compile(optimizer=Adam(lr=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
xceptionResults = model.fit(training, validation_data = validation, epochs=1000,
callbacks=[es])
testModel(model)
graphEpochs(xceptionResults, "Xception", "14")
```