PROGRESS REPORT: ALZHEIMER'S DETECTION WITH DEEP LEARNING

Muhammad Irfan
Student# 1009715612
m.irfan@mail.utoronto.ca

Hwang Wei Ju
Student# 1007874178
william.ju@mail.utoronto.ca

Hitansh Bhatt
Student# 1007931632
hitansh.bhatt@mail.utoronto.ca

Aryan Ghosh
Student# 1008838821
aryan.ghosh@mail.utoronto.ca

ABSTRACT

This proposal presents Group 27's APS360 Final Project, which focuses on developing a deep-learning model for the early detection of Alzheimer's disease using medical imaging. The project aims to classify different stages of Alzheimer's based on brain scan images, enabling early diagnosis and intervention. This document provides an overview of the project objectives, the proposed neural network architecture, relevant research, and additional key components as outlined in the Project Proposal Handout and Rubric. —-Total Pages: **??**

1 PROJECT DESCRIPTION

Alzheimer's disease is a progressive neurological disorder and the leading cause of dementia worldwide. Early and accurate detection is crucial for timely intervention, which can slow disease progression and improve patients' quality of life. However, traditional diagnosis methods rely on subjective clinical evaluations, which can lead to delayed or missed diagnoses. This project aims to develop a deep learning model for automated detection of Alzheimer's disease using magnetic resonance imaging (MRI) scans. The model will classify different stages of Alzheimer's, from mild cognitive impairment to severe dementia, providing a valuable tool to assist medical professionals in early diagnosis.

With the increasing prevalence of Alzheimer's disease, scalable and efficient diagnostic tools are needed to reduce the burden on healthcare systems. Automating the detection process enhances accessibility, reduces reliance on subjective evaluations, and improves diagnostic accuracy. The use of MRI scans ensures a non-invasive and widely available approach, making this project highly practical for real-world applications. Deep learning, particularly Convolutional Neural Networks (CNNs), is well-suited for medical imaging tasks due to its ability to extract complex spatial patterns from images. Unlike traditional machine learning methods, CNNs eliminate the need for extensive manual feature engineering and can learn relevant biomarkers directly from MRI data. Given its proven success in medical imaging applications, deep learning is a promising approach for improving Alzheimer's diagnosis.

The model will take MRI brain scans as input and output a classification of the Alzheimer's stage. The dataset will be preprocessed to ensure uniformity, and techniques such as data augmentation will be used to handle class imbalances. The CNN architecture will be designed to extract features from the images, followed by classification layers to determine the disease stage. The input consists of MRI brain scans, while the output classifies them into one of four categories: Non-Demented, Very Mild Dementia, Mild Dementia, and Moderate Dementia. The complexity of MRI images necessitates a model capable of recognizing intricate patterns. CNNs excel in feature extraction and classification, making them a robust choice for this medical imaging task.

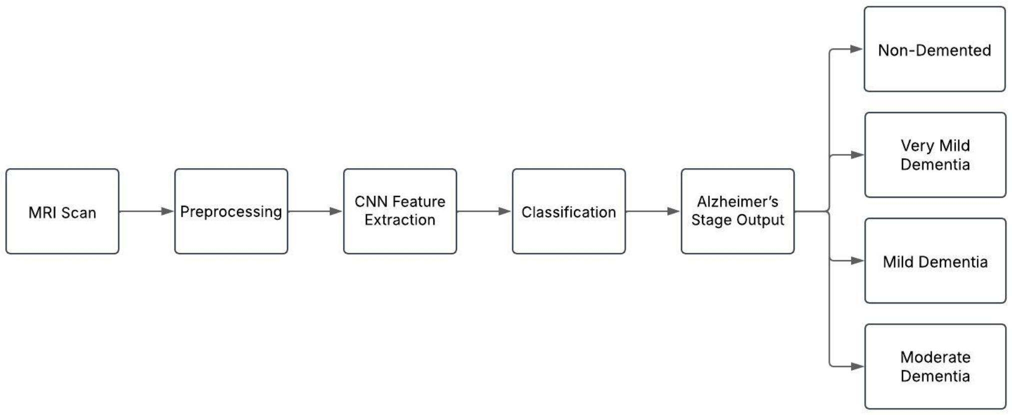


Figure 1: Illustration of Project Workflow.

2 Individual Contribution and Responsibilities

Our team has been working collaboratively to ensure timely and successful completion of the Alzheimer's Detection with Deep Learning project. We use Github for version control, Google Colab for model development and experimentation, Discord for communication, and Notion for project management, tracking tasks, deadlines, and progress updates. Regular weekly meetings are held to discuss individual contributions, review challenges, and ensure alignment with project milestones.

2.1 Team Members and responsibilities

Table 1: Team Responsibilities and Progress

Member	**Responsibilities**	**Progress**	**Next Steps**
William	Acquired dataset, assisted processing, and developed baseline model.	Data collected, baseline model implemented.	Hyperparameter tuning and explore transfer learning.
Hitansh	Designed CNN architecture and trained the model. Extracted model outputs (graphs, confusion matrix, etc.).	CNN architecture and preliminary training completed.	Fine-tune model and explore transfer learning.
Irfan	Processed MRI scans, implemented data augmentation, and tested baseline model.	Preprocessing and baseline model implemented.	Tune hyperparameters and fine-tune the model.
Aryan	Implemented Data Augmentation to increase dataset size and supported model development.	Data Augmentation API integrated.	Verify augmentation effectiveness and tune primary model hyperparameters.

2.2 Team Progress Summary

We are currently on track with our project deadlines. Data collection, preprocessing, and baseline model implementation have been completed. Model training was recently finished, and we are now transitioning into hyperparameter tuning and optimization. Weekly meetings help maintain strong

Table 2: Updated Project plan

Task	Description	Deadline	Members Assigned	Status
Data collection	Gathering relevant datasets from various sources for model training and evaluation.	Feb 1	William, Hitansh	Completed
Data handling	Processing and preparing the dataset for training.	Feb 15	Irfan, Aryan	Completed
Implementing baseline model	Developing a basic machine learning model for initial testing.	March 1	William, Irfan	Completed
Train the model	Running the training process on the dataset.	March 8	Hitansh, Aryan	Completed
Transfer Learning	Applying a pre-trained deep learning model to improve classification accuracy	March 22	William, Hitansh	Not started
Tune hyperparameters	Optimizing model parameters for better accuracy.	March 22	All team members	Not started
Final report and presentation	Preparing documentation and presentation for final submission.	April 1	All team members	Not started

communication, ensuring that challenges are addressed promptly and that the team remains aligned with the project timeline.

2.3 Redundancy Plan for Critical Tasks

To mitigate potential risks that may hinder project completion, we have identified key tasks that are essential to the project's success. Each of these tasks has a designated backup member to ensure continuity in case of unforeseen circumstances such as illness, scheduling conflicts, or technical issues. This redundancy plan ensures that critical aspects of our project remain on track.

Table 3: Redundancy Plan for Critical Tasks

Critical Task	**Primary Responsible Member**	**Backup Member**
Dataset Acquisition and Preprocessing	William	Aryan
Model Architecture Design	Hitansh	William
Model Training and Optimization	Aryan	Irfan

Redundancy Strategies:

- **Shared Access:** All essential datasets, scripts, and models are stored in a shared GitHub repository and accessible by all team members.

- **Knowledge Sharing:** Weekly meetings are conducted to ensure that each backup member understands the key processes.
- **Task Documentation:** A step-by-step guide for crucial tasks is maintained on Notion to enable quick onboarding if a backup member needs to take over.
- **Code Versioning:** GitHub is used to track changes in code and allow seamless transition between team members.

By implementing this redundancy plan, we minimize the risk of project delays and ensure smooth collaboration even if a primary team member is unavailable.

3 Notable Contribution

3.1 Data Processing

Proper data preparation is a crucial step in ensuring the success of a deep learning model. High-quality, well-preprocessed data can significantly enhance model performance, reduce biases, and prevent overfitting. This section outlines the essential steps taken to clean, preprocess, and balance the dataset to ensure robust and reliable learning. Effective data preparation not only improves accuracy but also ensures that the model generalizes well to unseen data. The following steps outline the data preparation process:

1. **Importing the Dataset:** The dataset will be taken from kaggle by using the import method directly from kaggle which is more effective and less need of storage.

```
    import kagglehub
    path = kagglehub.dataset_download("ninadaithal/imagesoasis")

```

2. **Data Management:** After importing, we need to manage the dataset and get the dataset file location into the array. In this case, it would help us to train the model by using this array to access each file in the folder.
   (a) Creates a 2D list (array) with 4 empty lists (4 because we have 4 categories) which will store file paths.
   (b) Loops through all folders inside a given `folder_path`. Walks through each folder (including subfolders) to find all files.
   (c) Stores the full path of each file in one of the 4 lists inside the array.
   (d) Increases `count` to move to the next list (but incorrectly, so it may cause an error).

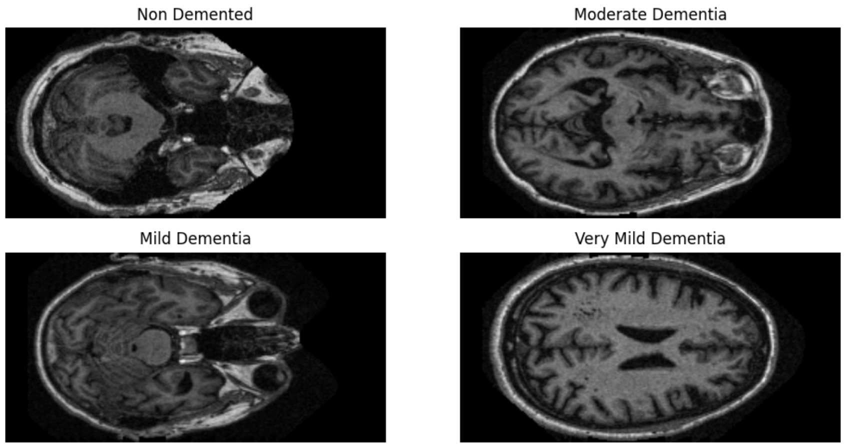
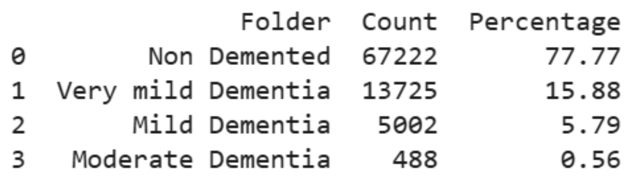


Figure 2: Alzheimer's dataset for each category

3. **Data Overview:** Check the number of dataset for each category to get the information about the distribution and if either it's balanced or not.



```
                Folder  Count  Percentage
0         Non Demented  67222       77.77
1  Very mild Dementia   13725       15.88
2        Mild Dementia   5002        5.79
3    Moderate Dementia    488        0.56
```

Figure 3: Dataset distribution for each category

From the Figure 2, it seems the data is imbalanced since the gap between each category especially Moderate Dementia and Non Demented is very big.

To solve that case, we actually have two options:

(a) Reduce the number of dataset to be 488 for each category (since the smalles category is Moderate Dementia and only has 488 dataset)

(b) Increase number of dataset by using data augmentation.

We successfully used the first options. However, we also try the second options and see if it improves the performance.

4. **Data Augmentation:** Due to the issue of imbalance, data augmentation would be crucial to help the model get the knowledge of each category equally. Minority category such as Moderate Dementia need to be augmented by using some method, such as:

(a) Random cropping
   i. Crops a random 250x250 region from the original image.
   ii. Helps in augmenting images by providing different perspectives and removing unnecessary parts.

(b) Horizontal Flipping
   i. Flips the image horizontally with a 50% probability (p=0.5).
   ii. Helps the model learn invariance to left-right orientation.

(c) Random Brightness contrast
   i. Randomly adjusts the brightness and contrast of the image with a 20% probability (p=0.2).
   ii. Helps the model generalize to different lighting conditions.

(d) Image rotation
   i. Rotates the image randomly within a ±20-degree range with a 50% probability (p=0.5).
   ii. Helps the model learn rotation-invariant features.

(e) Normalization
   i. Normalizes pixel values using the mean and standard deviation from the ImageNet dataset.
   ii. Ensures consistency in input data and improves model training.

Using this method, we want to increase the number of dataset for Moderate Dementia to be 1000 (which is 2 times higher than our first method and could be increased to be 5000 if we want as well). Then, we would see if there is an improvement in the performance.

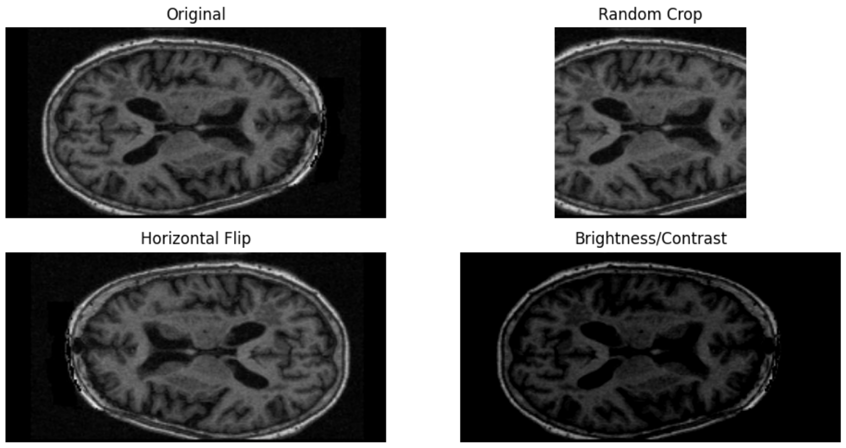


Figure 4: Augmentation of Alzheimer's dataset

5. **One-Hot Encoding:** The dataset is divided into four categories: *Non-Demented*, *Mild Dementia*, *Moderate Dementia*, and *Very Mild Dementia*. These string labels will be converted into integer labels using one-hot encoding for compatibility with the deep learning model.
   - **Categories:** {Non-Demented, Mild Dementia, Moderate Dementia, Very Mild Dementia}
   - **Integer Encoding:** {0, 1, 2, 3}

6. **Image Resizing:** Each image in the dataset will be resized to `(128, 128, 3)` dimensions to ensure uniformity. This step is essential as the deep learning model requires input images to have consistent dimensions.

7. **Data Splitting:** The dataset will be split into training, validation, and testing sets. Specifically, 70% of the data will be used for training, 15% for validation, and 15% for testing. This ensures that the model is trained effectively while also having separate datasets for tuning hyperparameters and evaluating final performance.

3.1.1 Handling Never-Before-Seen Data:

The dataset is split into three subsets: training, validation, and testing. The testing dataset is used only after selecting the best model from the training process and fine-tuning the hyperparameters through validation. Since the testing dataset consists of entirely new data that has not been used before, it serves as an effective measure of the model's performance on unseen data.

3.1.2 Challenges:

One of the main challenges in this process is the imbalance in the dataset, as described in the data overview section. To address this issue, we will apply data augmentation techniques to balance the dataset and improve model generalization.

3.2 Baseline Model

The baseline model chosen for comparison is a Random Forest classifier due to its simplicity, interpretability, and widespread use as a reference in classification tasks. It provides a minimum performance benchmark, allowing us to assess whether more complex models, such as neural networks, are necessary. In this case, we use Random Forest because it offers a balance between accuracy and efficiency while being less prone to overfitting compared to more complex models.

3.2.1 Implementation Details

1. **Input Format**
   - Images are resized to $128 \times 128 \times 3$ for consistency.
   - Flattened into one-dimensional vectors.
2. **Preprocessing**
   - Pixel values are normalized to $[0, 1]$ for better convergence.
   - No additional feature extraction is applied.
3. **Model Architecture and Regularization**
   - Random forest with softmax activation for multi-class classification.
   - L_2 regularization ($\lambda = 1.0$) to prevent overfitting.
4. **Training and Evaluation**
   - Trained using cross-entropy loss and optimized via LBFGS solver (100 iterations).
   - Performance evaluated using a confusion matrix and standard metrics: accuracy, precision, recall, and F1-score.

3.2.2 Results:

The baseline model demonstrates strong overall performance, achieving balanced scores for accuracy (81.25%), recall (81.25%), and precision (81.36%). However, since the primary objective of this project is to detect Alzheimer's disease, recall is the most critical metric to minimize false negatives.

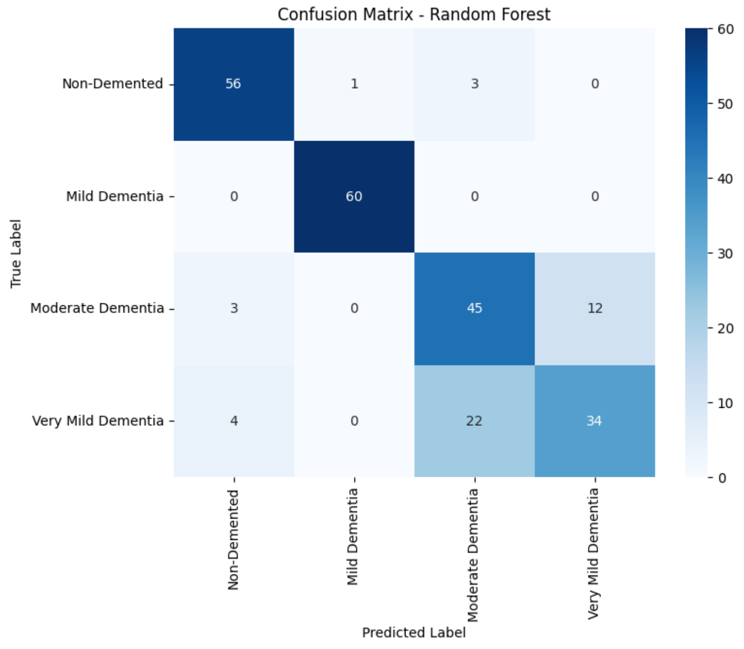


Figure 5: Baseline Model's Confusion Matrix Result

As shown in Figure 7, the confusion matrix reflects strong classification performance across categories. However, most misclassifications occur in the Moderate Dementia and Very Mild Dementia categories which seems maybe looks similar so there are some errors.

3.2.3 Challenges:

Fortunately, we did not encounter any significant challenges while building the baseline model. The implementation was straightforward, requiring minimal tuning. However, minor adjustments were needed to ensure proper data preprocessing specifically for baseline model such as reshape the dataset to be 2D vectors.

3.3 Primary Model

The AlzheimerNet model developed for classifying MRI images into four categories of Alzheimer's disease (Non-Demented, Very Mild Dementia, Mild Dementia, and Moderate Dementia) is a deep convolutional neural network (CNN) implemented using PyTorch. The architecture consists of six convolutional layers, each followed by batch normalization, ReLU activation, and max pooling to reduce spatial dimensions and improve feature extraction.

A fully connected (FC) layer with dropout regularization is used after flattening the extracted features, leading to a final output layer with four neurons (one for each class), activated using Softmax for multi-class classification.

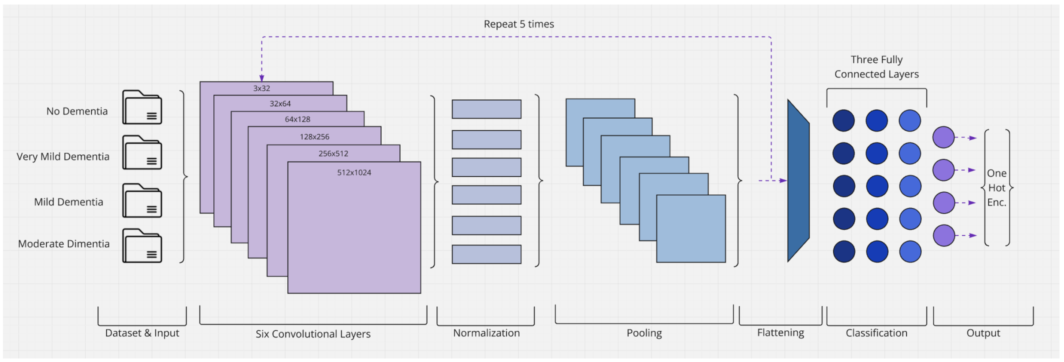


Figure 6: Primary Model Architecture

3.3.1 Model Architecture Overview:

- **Input Shape:** Standardized grayscale MRI images of size (128, 128). The model processes MRI scans as grayscale images with a fixed resolution of 128×128 pixels, ensuring uniformity and compatibility with the convolutional layers.
- **Convolutional Layers:** 5 convolutional layers with increasing filter sizes (32, 64, 128, 256, 512). These layers progressively extract hierarchical features, capturing both low-level details (such as edges and textures) and high-level patterns crucial for Alzheimer's classification.
- **Kernel Size & Stride:** 3×3 kernels with stride 1 for detailed feature extraction. The small 3×3 kernel size enables the model to detect intricate patterns while maintaining computational efficiency, and a stride of 1 ensures fine-grained feature extraction.
- **Activation Function:** ReLU activation applied after each convolutional layer. The ReLU (Rectified Linear Unit) activation function introduces non-linearity, helping the model learn complex representations while mitigating the vanishing gradient problem.
- **Batch Normalization:** Applied after each convolutional block to stabilize training. Batch normalization improves training efficiency by normalizing feature distributions, reducing internal covariate shifts, and allowing for faster convergence.
- **Pooling Layers:** Max pooling (2×2) after each convolutional block to reduce spatial dimensions. Max pooling layers downsample feature maps, reducing computational cost and overfitting while retaining the most relevant features for classification.
- **Dropout Regularization:** Applied in fully connected layers to prevent overfitting (dropout rate: 0.5). Dropout (50%) forces the model to randomly deactivate neurons during training, preventing reliance on specific neurons and improving generalization to unseen data.
- **Fully Connected Layers:** After feature extraction, the model flattens feature maps and processes them through dense layers to make the final classification.

- **Flattening Layer:** Converts feature maps into a one-dimensional vector to feed into dense layers.
- **Dense Layers:** Three fully connected layers with 512, 256, and 128 neurons, each using ReLU activation to refine the learned representations.
- **Output Layer:** The final layer consists of 4 neurons, each representing an Alzheimer's stage, using Softmax activation for multi-class classification.

- **Total number of parameters:** Currently, the module produces **eleven million one hundred forty-four thousand seven hundred eight (11,144,708)** trainable parameters. This parameter count reflects the depth and complexity of the architecture, making it a powerful yet computationally feasible model for MRI-based Alzheimer's detection.

3.3.2 MODEL'S PERFORMANCE:

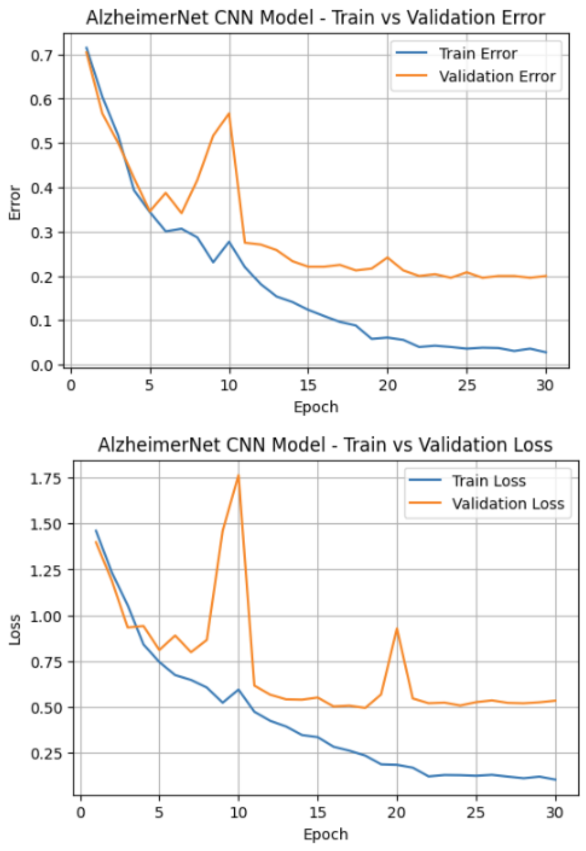


Figure 7: Primary Model Performance

The AlzheimerNet CNN model showed effective learning over 30 epochs, with a steady decrease in training error and loss. However, validation error and loss fluctuations indicate overfitting and training instability, likely due to class imbalance or high data variance. To improve generalization, we propose increasing dropout, applying L2 regularization, and expanding data augmentation. Additionally, lowering the learning rate and implementing early stopping could stabilize training. Future work will focus on hyperparameter tuning and transfer learning to enhance performance and ensure reliable Alzheimer's detection using MRI scans.