Cognitive Care: Early Intervention For Alzheimer's Disease

Final Project Report

Team Leader = Varuni D

Team Member = Aklavya Bhagat

Team Member = Guna Shekhar

Team Member = Sanjana

1. Introduction

1.1 Project Overview

The objective of this project is to pioneer the application of deep learning methodologies in the early detection of Alzheimer's Disease through the analysis of MRI scans. By harnessing cutting-edge machine learning techniques, the project emphasizes the refinement of diagnostic accuracy and consistency in identifying various stages of Alzheimer's development from intricate medical imaging data. Through the integration of state-of-the-art models and meticulous pre-processing methods, our aim is to establish a dependable diagnostic framework that supports healthcare practitioners in prompt intervention and personalized treatment strategies.

1.2 Objective

The main objectives of the project include:

1. **Model Development:** Develop and optimize advanced deep learning models capable of accurately classifying Alzheimer's Disease progression from MRI scans with high precision and sensitivity.
2. **Data Enhancement:** Implement robust data preprocessing techniques such as normalization, augmentation, and effective handling of data imbalance to enhance model robustness and performance.
3. **Validation and Fine-tuning:** Conduct extensive validation and fine-tuning processes to ensure the reliability and generalizability of the models across diverse patient populations and imaging conditions.
4. **Deployment Preparation:** Prepare the finalized models for seamless integration into clinical workflows, ensuring compatibility with existing medical systems and adherence to regulatory standards.
5. **Impact Evaluation:** Assess the potential clinical impact and utility of the developed models in facilitating early diagnosis and personalized treatment strategies for Alzheimer's Disease patients, aiming to improve patient outcomes and healthcare delivery.

2. Project Initialization and Planning Phase

2.1 Define Problem Statement

Alzheimer's Disease (AD) poses a significant challenge, affecting millions worldwide, yet early symptoms are often unrecognized or misunderstood, leading to delays in critical intervention. This delay is particularly pronounced in underserved areas where resources and awareness for early detection are limited. Social stigma and misconceptions about AD further exacerbate these challenges, hindering timely diagnosis and intervention.

Problem Statement:

- **Customer:** As a 55-year-old female educator,
- **I am trying to:** Continue my career and remain mentally sharp,
- **But:** I struggle with recognizing early symptoms of Alzheimer's Disease,
- **Because:** There is a significant lack of accessible information and resources,
- **Which makes me feel:** Uncertain about my cognitive health and future abilities

2.2 Project Proposal (Proposed Solution)

Project Overview

Objectives:

1. **Early Diagnosis:** Develop tools and systems for early detection of Alzheimer's Disease.
2. **Awareness:** Educate the public and healthcare professionals about early signs and symptoms.
3. **Personalized Care Plans:** Create tailored care plans based on individual patient needs.
4. **Support Services:** Provide resources and services to support patients and caregivers.
5. **Monitoring and Feedback:** Implement systems to monitor disease progression and provide feedback to healthcare providers.

Scope: The project focuses on developing online tools, educational resources, and personalized care plans to facilitate early intervention for Alzheimer's Disease. It excludes direct medical treatment and in-person services.

Problem Statement: The project addresses the critical need for early detection and personalized intervention strategies for Alzheimer's Disease to improve patient outcomes and quality of life.

Impact: Successfully addressing this problem could significantly delay disease progression, preserve cognitive function, and enhance overall quality of life for individuals affected by Alzheimer's Disease.

Proposed Solution:

- **Approach:** Utilize agile development methodologies to iteratively refine online tools, educational materials, care plans, support service integrations, and monitoring systems.

Collaborate closely with stakeholders and healthcare experts for comprehensive input and feedback.

- **Key Features:** The proposed solution integrates advanced technology with personalized care plans and community engagement to enhance early intervention and support for Alzheimer's Disease.

Resource Requirements

Resource Type	Description	Specification/Allocation
Hardware	Computing Resources	CPU/GPU specifications, number of cores T4 GPU
Memory	RAM specifications	16 GB
Storage	Disk space for data, models, and logs	512 GB SSD
Software	Frameworks	Python frameworks Flask, Django, Pandas, Numpy
Libraries	Additional libraries	TensorFlow, Scikit-learn
Development Environment	IDE, version control	Jupyter Notebook, Git, Google Collab, Spyde

2.3 Initial Project Planning

Sprint-1: Data Collection

- **Functional Requirement (Epic)**: Data Collection
- **User Story Number**: USN-1
- **User Story / Task**: Collect images of brain MRI then organize into subdirectories based on their respective names as shown in the project structure. Create folders of types of Alzheimer.
- *Story Points****: 8***
- **Priority**: Medium
- **Team Members**: Varuni
- **Sprint Start Date**: 2024-07-10
- **Sprint End Date (Planned)**: 2024-07-10

Sprint-1: Image Pre-processing

- **Functional Requirement (Epic)**: Image Pre-processing
- **User Story Number**: USN-2
- **User Story / Task**: Importing the necessary libraries, Handling Imbalance DataStory **Points: 5**
- **Priority**: Medium
- **Team Members**: Varuni
- **Sprint Start Date**: 2024-07-10
- **Sprint End Date (Planned)**: 2024-07-10

Sprint-2: Model Development

- **Functional Requirement (Epic)**: Model Development
- **User Story Number**: USN-3
- **User Story / Task** Developing and training the model.Story
- **Points: 5**
- **Priority**: Medium
- **Team Members**: Varuni
- **Sprint Start Date**: 2024-07-12
- **Sprint End Date (Planned)**: 2024-07-12

Sprint-3: Model Tuning and Testing

- **Functional Requirement (Epic)**: Model Tuning and Testing
- **User Story Number**: USN-4
- **User Story / Task**: Testing the model with different datasets and checking for errors.
- **Story** Points**: 8**
- **Priority**: High
- **Team Members**: Aklavya Bhagat, Varuni
- **Sprint Start Date**: 2024-07-12
- **Sprint End Date (Planned)**: 2024-07-13

Sprint-4: Application Building

- **Functional Requirement (Epic)**: Application Building
- **User Story Number**: USN-5
- **User Story / Task** Building HTML pages, building Flask code and running the application
- **Story Points: 5**
- **Priority**: High
- **Team Members**: Aklavya
- **Sprint Start Date**: 2024-07-18
- **Sprint End Date (Planned)**: 2024-07-19

Sprint-5: Documentation and Report

- **Functional Requirement (Epic)**: Documentation and Report
- **User Story Number**: USN-6
- **User Story / Task**: Documenting necessary templates and making a report
- ***Story Points: 8***
- **Priority**: High
- **Team Members**: Varuni, Aklavya, Guna Shekhar, Sanjana
- **Sprint Start Date**: 2024-07-19
- **Sprint End Date (Planned)**: 2024-07-20

3. Data Collection and Preprocessing Phase

3.1 Data Collection Plan and Raw Data Sources Identified

Effective data training strategies rely on strategic planning to systematically identify and gather relevant data sources. By ensuring data integrity and comprehensive coverage, organizations can improve the accuracy and reliability of their training datasets, enhancing informed decision-making and robust model performance in data-driven applications.

Data Collection Plan

Project Overview

The project aims to develop advanced diagnostic models for Alzheimer's Disease using MRI scans. It focuses on meticulous data collection and rigorous training methods to elevate model accuracy and facilitate informed medical decisions.

Data Collection Plan

Data for this project was sourced from Kaggle, a renowned platform for datasets and machine learning competitions.

Raw Data Sources Identified

The raw data sources for this project consist of JPEG images sourced from Kaggle. These images are categorized into four classes, representing different stages of Alzheimer's Disease progression:

- Mild Cognitive Impairment
- Moderate Cognitive Impairment
- No Cognitive Impairment
- Very Mild Cognitive Impairment

Raw Data Sources Report

- **Source Name:** Kaggle Dataset
- **Description:** This dataset includes JPEG images sourced from Kaggle, depicting MRI scans categorized into four classes based on varying stages of cognitive impairment.
- **Location/URL:** Kaggle Alzheimer's Dataset
- **Format:** JPEG
- **Size:** 33.0 MB
- **Access Permissions:** Public

3.2 Data Quality Report

The Data Quality Report summarizes issues identified in the selected data source, detailing severity levels and plans for resolution to ensure accuracy and reliability in subsequent analyses.

Data Source: Kaggle Dataset

Data Quality Issues:

1. **Class imbalance among Alzheimer's Disease categories**
   - **Severity:** Moderate
   - **Resolution Plan:** Utilize SMOTE (Synthetic Minority Over-sampling Technique) to generate synthetic samples for underrepresented classes, ensuring balanced representation during model training.
2. **Inconsistencies in image quality and resolution across MRI scans**
   - **Severity:** High
   - **Resolution Plan:** Standardize preprocessing methods including resizing to a uniform resolution, application of denoising filters, and consistent contrast adjustment to enhance image clarity and uniformity.

3.3 Data Pre-processing

Data exploration and pre-processing for Alzheimer's Disease involve critical steps aimed at improving data quality, enhancing model generalization, and optimizing neural network training using advanced computer vision techniques.

Data Overview

The project utilizes MRI scans for Alzheimer's Disease sourced from Kaggle, categorized into multiple classes.

Resizing

All images are resized to a uniform 180x180 pixels to meet input requirements for the Xception model used in preprocessing.

Normalization

Normalization adjusts pixel values to a scale between 0 and 1, ensuring consistent data ranges across images for effective model training and convergence.

Data Augmentation

Techniques such as brightness adjustments, zooming, and horizontal flipping are employed for data augmentation, diversifying training data to enhance model robustness in recognizing patterns related to Alzheimer's Disease.

Data Balancing

Addressing class imbalance involves using SMOTE to generate synthetic samples, improving model accuracy across all Alzheimer's Disease categories.

Transfer Learning

Transfer learning incorporates a pre-trained Xception model with frozen layers, leveraging previously acquired knowledge to enhance classification performance for Alzheimer's Disease.

Batch Normalization

Stabilizing training through batch normalization normalizes input batches, promoting model convergence and generalization.

4. Model Development Phase

4.1 Model Selection Report

Xception

The Xception model, short for Extreme Inception, is a sophisticated architecture in deep convolutional neural networks that builds upon the Inception model. It replaces traditional Inception modules with depth-wise separable convolutions, drastically reducing parameters and computational complexity while maintaining high accuracy.

VGG19

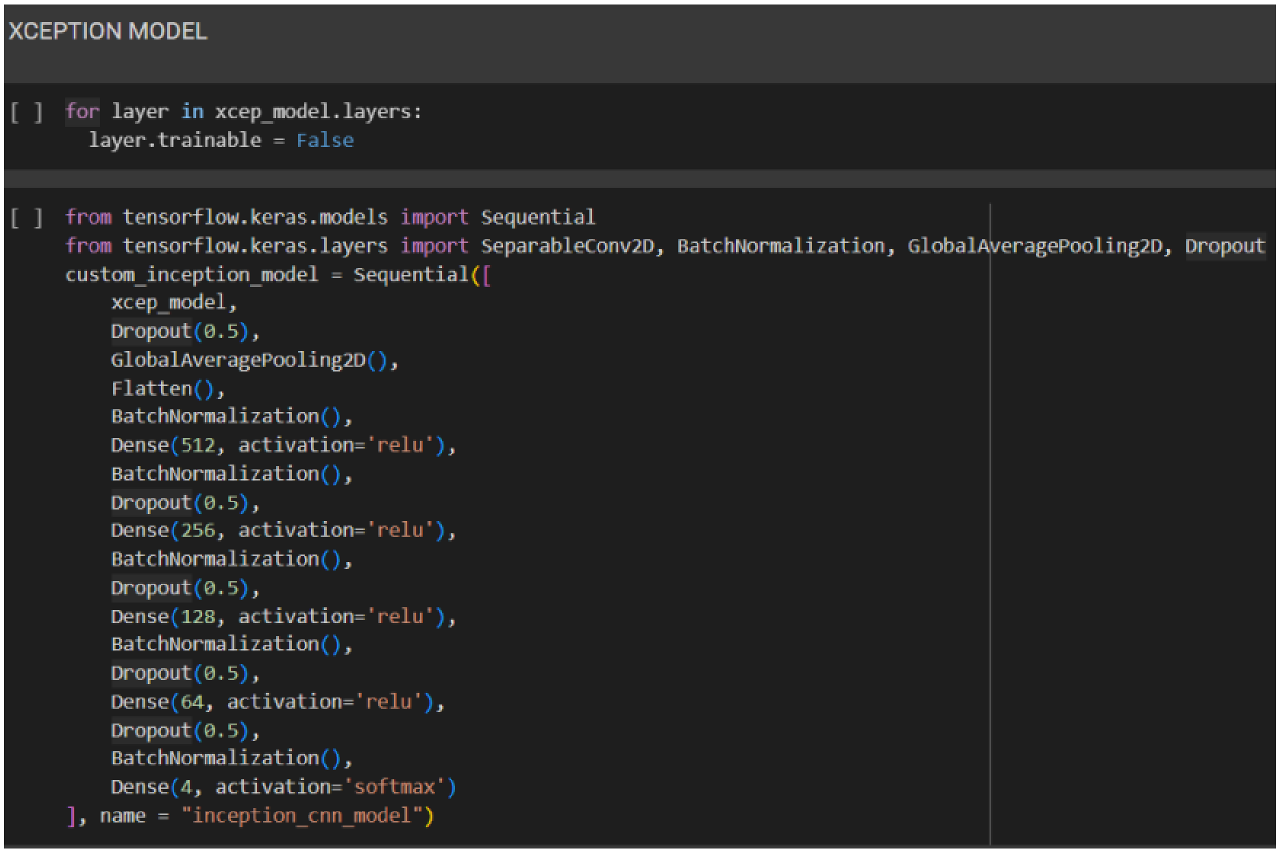
VGG19 is a deep convolutional neural network architecture comprising 19 layers, developed by the Visual Geometry Group at the University of Oxford. Renowned for its straightforwardness and depth, VGG19 utilizes small 3x3 convolution filters uniformly across the network, enhancing its capacity to learn intricate features.

Inception V3

Inception V3 is an advanced deep convolutional neural network architecture introduced by Google within the Inception series. It integrates sophisticated techniques such as factorized convolutions, robust regularization, and label smoothing to boost model efficacy and lower computational demands.

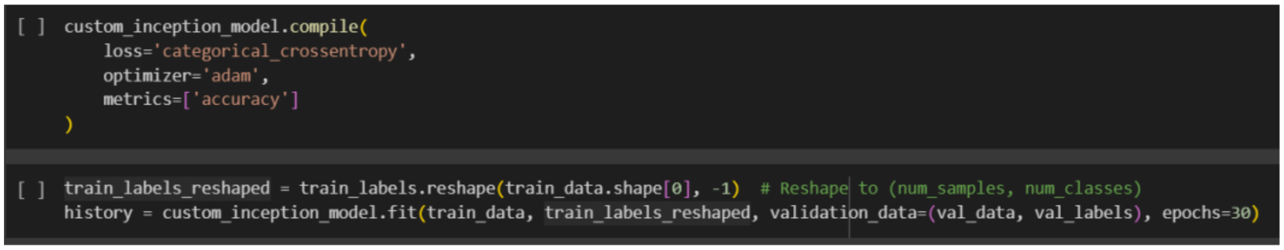
4.2 Initial Model Training Code, Model Validation and Evaluation Report

Xception Model


XCEPTION MODEL

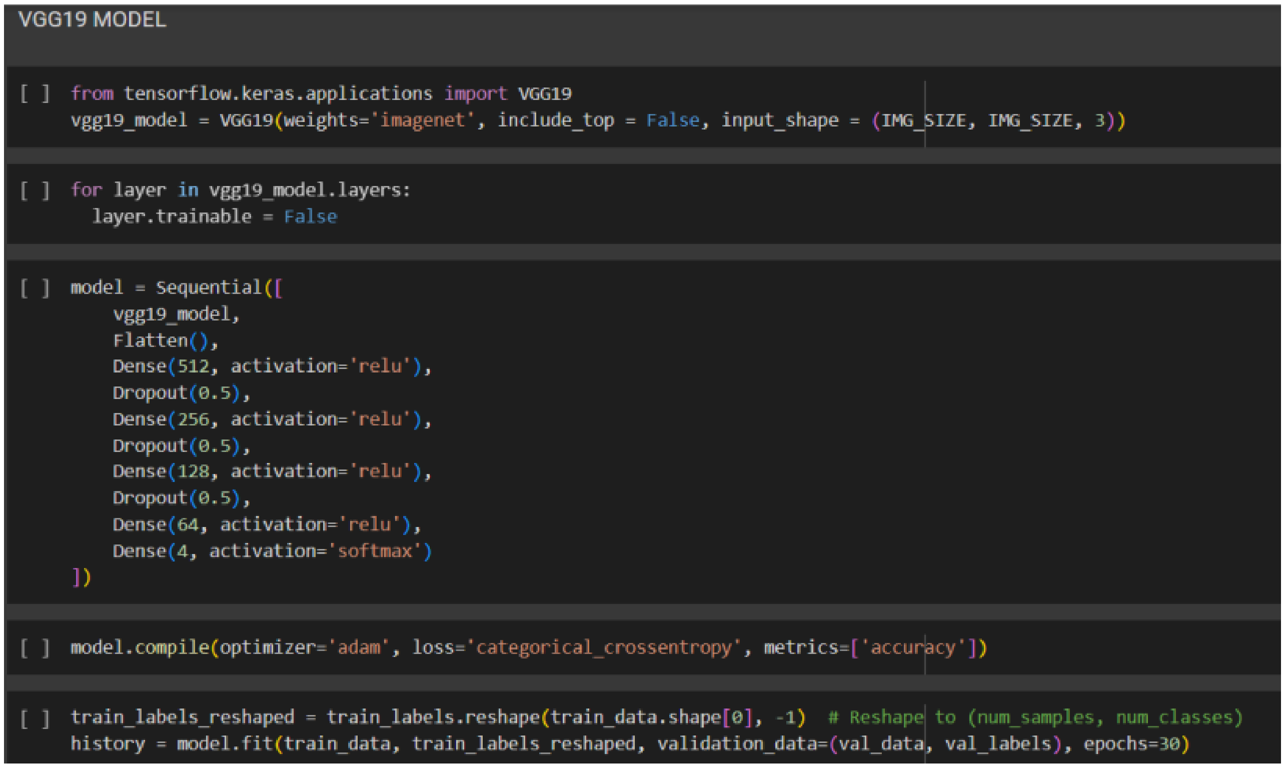
```
[ ] for layer in xcep_model.layers:
      layer.trainable = False
```

```
[ ] from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import SeparableConv2D, BatchNormalization, GlobalAveragePooling2D, Dropout
    custom_inception_model = Sequential([
        xcep_model,
        Dropout(0.5),
        GlobalAveragePooling2D(),
        Flatten(),
        BatchNormalization(),
        Dense(512, activation='relu'),
        BatchNormalization(),
        Dropout(0.5),
        Dense(256, activation='relu'),
        BatchNormalization(),
        Dropout(0.5),
        Dense(128, activation='relu'),
        BatchNormalization(),
        Dropout(0.5),
        Dense(64, activation='relu'),
        Dropout(0.5),
        BatchNormalization(),
        Dense(4, activation='softmax')
    ], name = "inception_cnn_model")
```


```
[ ] custom_inception_model.compile(
        loss='categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy']
    )
```

```
[ ] train_labels_reshaped = train_labels.reshape(train_data.shape[0], -1)  # Reshape to (num_samples, num_classes)
    history = custom_inception_model.fit(train_data, train_labels_reshaped, validation_data=(val_data, val_labels), epochs=30)
```

VGG19 Model



VGG19 MODEL

```
[ ] from tensorflow.keras.applications import VGG19
    vgg19_model = VGG19(weights='imagenet', include_top = False, input_shape = (IMG_SIZE, IMG_SIZE, 3))
```

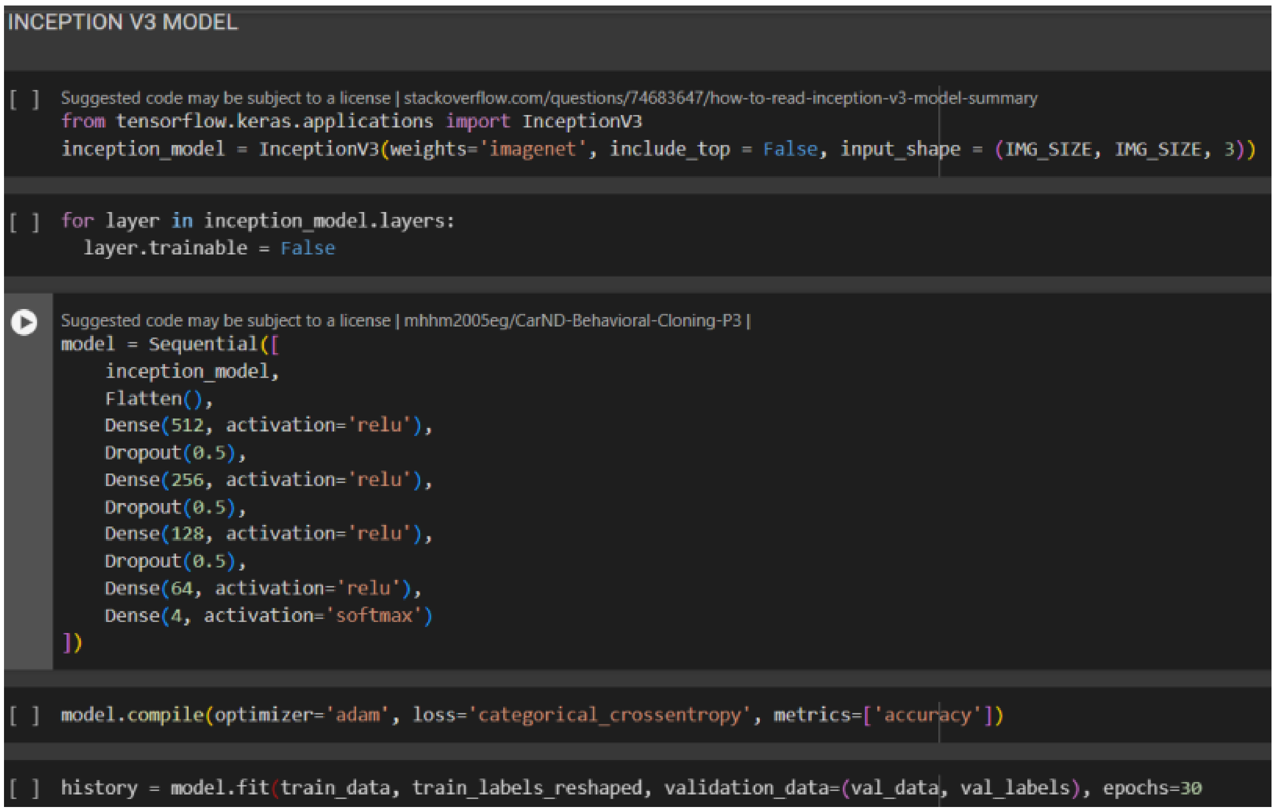
```
[ ] for layer in vgg19_model.layers:
      layer.trainable = False
```

```
[ ] model = Sequential([
        vgg19_model,
        Flatten(),
        Dense(512, activation='relu'),
        Dropout(0.5),
        Dense(256, activation='relu'),
        Dropout(0.5),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(64, activation='relu'),
        Dense(4, activation='softmax')
    ])
```

```
[ ] model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
[ ] train_labels_reshaped = train_labels.reshape(train_data.shape[0], -1)  # Reshape to (num_samples, num_classes)
    history = model.fit(train_data, train_labels_reshaped, validation_data=(val_data, val_labels), epochs=30)
```

Inception V3 Model



INCEPTION V3 MODEL

```
[ ] Suggested code may be subject to a license | stackoverflow.com/questions/74683647/how-to-read-inception-v3-model-summary
    from tensorflow.keras.applications import InceptionV3
    inception_model = InceptionV3(weights='imagenet', include_top = False, input_shape = (IMG_SIZE, IMG_SIZE, 3))
```

```
[ ] for layer in inception_model.layers:
      layer.trainable = False
```

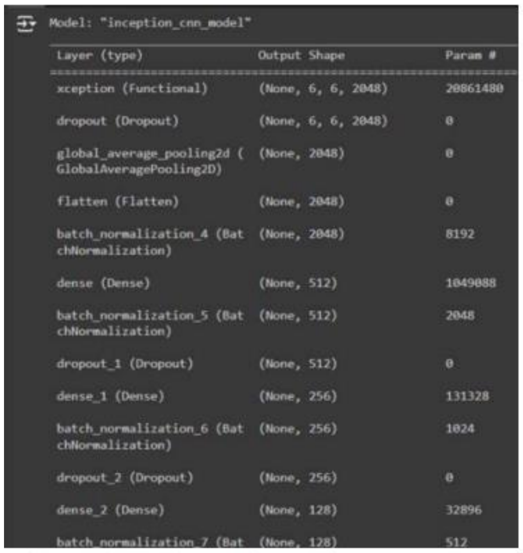
```
    Suggested code may be subject to a license | mhhm2005eg/CarND-Behavioral-Cloning-P3 |
    model = Sequential([
        inception_model,
        Flatten(),
        Dense(512, activation='relu'),
        Dropout(0.5),
        Dense(256, activation='relu'),
        Dropout(0.5),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(64, activation='relu'),
        Dense(4, activation='softmax')
    ])
```

```
[ ] model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

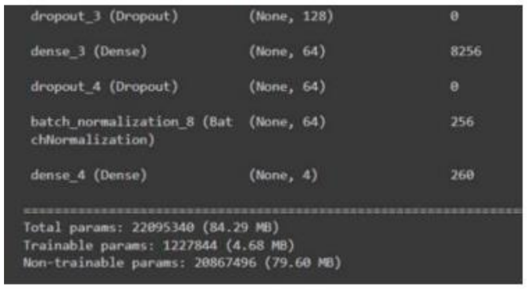
```
[ ] history = model.fit(train_data, train_labels_reshaped, validation_data=(val_data, val_labels), epochs=30
```

Model Validation and Evaluation Report:

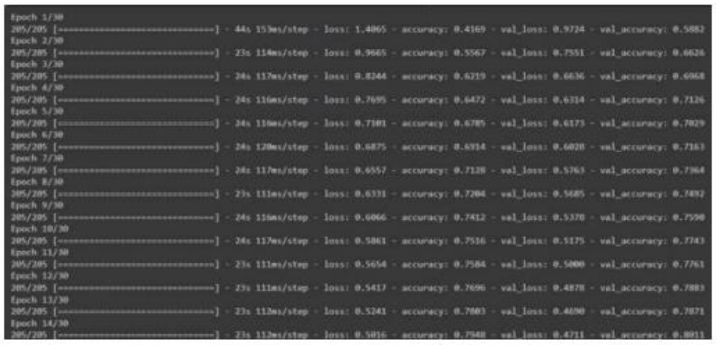
Xception Model



```
Model: "inception_cnn_model"

 Layer (type)                Output Shape              Param #
=================================================================
 xception (Functional)       (None, 6, 6, 2048)        20861480

 dropout (Dropout)           (None, 6, 6, 2048)        0

 global_average_pooling2d (  (None, 2048)              0
 GlobalAveragePooling2D)

 flatten (Flatten)           (None, 2048)              0

 batch_normalization_4 (Bat  (None, 2048)              8192
 chNormalization)

 dense (Dense)               (None, 512)               1049088

 batch_normalization_5 (Bat  (None, 512)               2048
 chNormalization)

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 batch_normalization_6 (Bat  (None, 256)               1024
 chNormalization)

 dropout_2 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 128)               32896

 batch_normalization_7 (Bat  (None, 128)               512
```

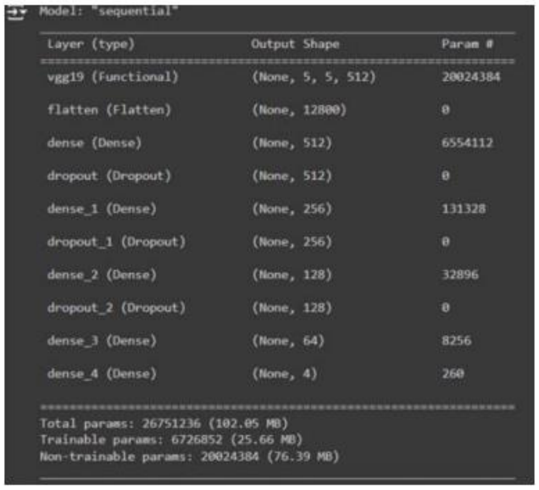


```
 dropout_3 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 64)                8256

 dropout_4 (Dropout)         (None, 64)                0

 batch_normalization_8 (Bat  (None, 64)                256
 chNormalization)

 dense_4 (Dense)             (None, 4)                 260

=================================================================
Total params: 22095340 (84.29 MB)
Trainable params: 1227844 (4.68 MB)
Non-trainable params: 20867496 (79.60 MB)
```

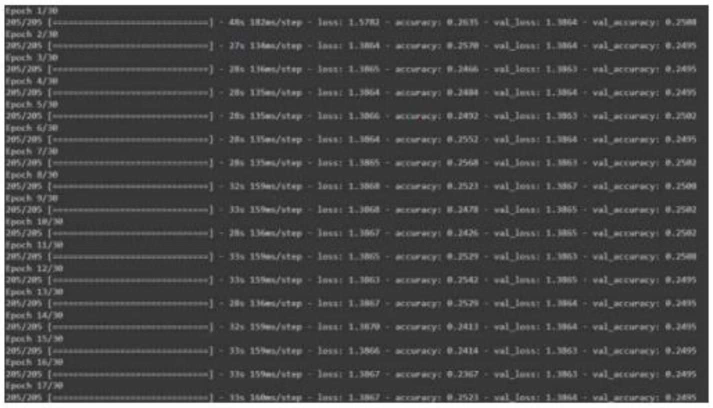


```
Epoch 1/30
205/205 [==============================] - 44s 153ms/step - loss: 1.4065 - accuracy: 0.4169 - val_loss: 0.9724 - val_accuracy: 0.5882
Epoch 2/30
205/205 [==============================] - 23s 114ms/step - loss: 0.9665 - accuracy: 0.5567 - val_loss: 0.7551 - val_accuracy: 0.6626
Epoch 3/30
205/205 [==============================] - 24s 117ms/step - loss: 0.8244 - accuracy: 0.6219 - val_loss: 0.6636 - val_accuracy: 0.6968
Epoch 4/30
205/205 [==============================] - 24s 116ms/step - loss: 0.7695 - accuracy: 0.6472 - val_loss: 0.6314 - val_accuracy: 0.7126
Epoch 5/30
205/205 [==============================] - 24s 116ms/step - loss: 0.7101 - accuracy: 0.6785 - val_loss: 0.6173 - val_accuracy: 0.7029
Epoch 6/30
205/205 [==============================] - 24s 120ms/step - loss: 0.6875 - accuracy: 0.6914 - val_loss: 0.6028 - val_accuracy: 0.7163
Epoch 7/30
205/205 [==============================] - 24s 117ms/step - loss: 0.6557 - accuracy: 0.7128 - val_loss: 0.5763 - val_accuracy: 0.7364
Epoch 8/30
205/205 [==============================] - 23s 111ms/step - loss: 0.6331 - accuracy: 0.7204 - val_loss: 0.5685 - val_accuracy: 0.7492
Epoch 9/30
205/205 [==============================] - 24s 116ms/step - loss: 0.6066 - accuracy: 0.7412 - val_loss: 0.5370 - val_accuracy: 0.7590
Epoch 10/30
205/205 [==============================] - 24s 117ms/step - loss: 0.5861 - accuracy: 0.7516 - val_loss: 0.5175 - val_accuracy: 0.7743
Epoch 11/30
205/205 [==============================] - 23s 111ms/step - loss: 0.5654 - accuracy: 0.7584 - val_loss: 0.5000 - val_accuracy: 0.7761
Epoch 12/30
205/205 [==============================] - 23s 111ms/step - loss: 0.5417 - accuracy: 0.7696 - val_loss: 0.4878 - val_accuracy: 0.7883
Epoch 13/30
205/205 [==============================] - 23s 112ms/step - loss: 0.5241 - accuracy: 0.7803 - val_loss: 0.4690 - val_accuracy: 0.7871
Epoch 14/30
205/205 [==============================] - 23s 112ms/step - loss: 0.5016 - accuracy: 0.7948 - val_loss: 0.4711 - val_accuracy: 0.8011
```

VGG19 Model

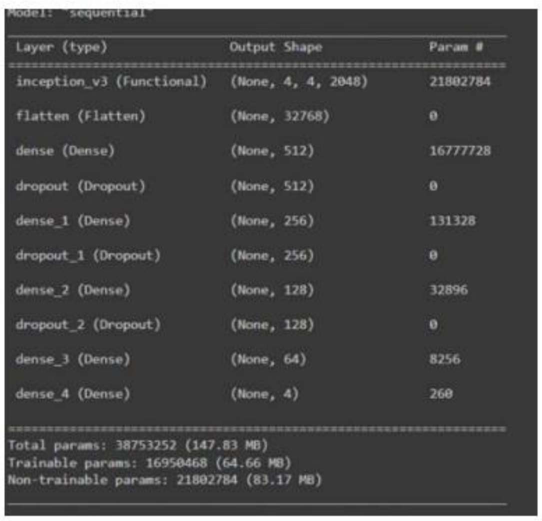


```
Model: "sequential"

 Layer (type)                Output Shape              Param #
=================================================================
 vgg19 (Functional)          (None, 5, 5, 512)         20024384

 flatten (Flatten)           (None, 12800)             0

 dense (Dense)               (None, 512)               6554112

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 128)               32896

 dropout_2 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 64)                8256

 dense_4 (Dense)             (None, 4)                 260

=================================================================
Total params: 26751236 (102.05 MB)
Trainable params: 6726852 (25.66 MB)
Non-trainable params: 20024384 (76.39 MB)
```

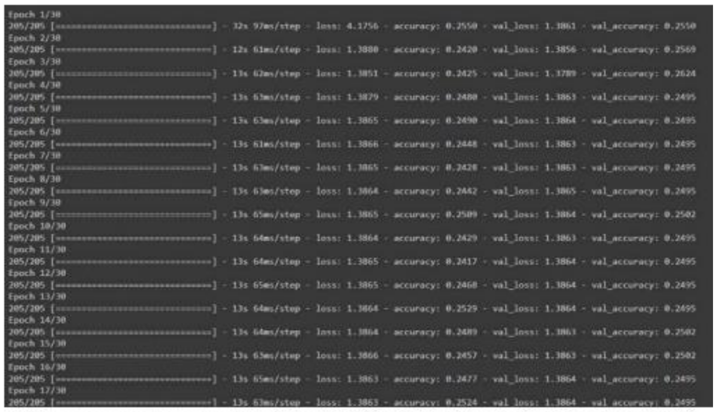


```
Epoch 1/30
205/205 [==============================] - 48s 182ms/step - loss: 1.5782 - accuracy: 0.2635 - val_loss: 1.3864 - val_accuracy: 0.2508
Epoch 2/30
205/205 [==============================] - 27s 134ms/step - loss: 1.3864 - accuracy: 0.2570 - val_loss: 1.3864 - val_accuracy: 0.2495
Epoch 3/30
205/205 [==============================] - 28s 136ms/step - loss: 1.3865 - accuracy: 0.2466 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 4/30
205/205 [==============================] - 28s 135ms/step - loss: 1.3864 - accuracy: 0.2484 - val_loss: 1.3864 - val_accuracy: 0.2495
Epoch 5/30
205/205 [==============================] - 28s 135ms/step - loss: 1.3866 - accuracy: 0.2492 - val_loss: 1.3863 - val_accuracy: 0.2502
Epoch 6/30
205/205 [==============================] - 28s 135ms/step - loss: 1.3864 - accuracy: 0.2552 - val_loss: 1.3864 - val_accuracy: 0.2495
Epoch 7/30
205/205 [==============================] - 28s 135ms/step - loss: 1.3865 - accuracy: 0.2568 - val_loss: 1.3863 - val_accuracy: 0.2502
Epoch 8/30
205/205 [==============================] - 32s 159ms/step - loss: 1.3868 - accuracy: 0.2523 - val_loss: 1.3867 - val_accuracy: 0.2508
Epoch 9/30
205/205 [==============================] - 33s 159ms/step - loss: 1.3868 - accuracy: 0.2478 - val_loss: 1.3865 - val_accuracy: 0.2502
Epoch 10/30
205/205 [==============================] - 28s 136ms/step - loss: 1.3867 - accuracy: 0.2426 - val_loss: 1.3865 - val_accuracy: 0.2502
Epoch 11/30
205/205 [==============================] - 33s 159ms/step - loss: 1.3865 - accuracy: 0.2529 - val_loss: 1.3863 - val_accuracy: 0.2508
Epoch 12/30
205/205 [==============================] - 33s 159ms/step - loss: 1.3863 - accuracy: 0.2542 - val_loss: 1.3865 - val_accuracy: 0.2495
Epoch 13/30
205/205 [==============================] - 28s 136ms/step - loss: 1.3867 - accuracy: 0.2529 - val_loss: 1.3864 - val_accuracy: 0.2495
Epoch 14/30
205/205 [==============================] - 32s 159ms/step - loss: 1.3870 - accuracy: 0.2413 - val_loss: 1.3864 - val_accuracy: 0.2495
Epoch 15/30
205/205 [==============================] - 33s 159ms/step - loss: 1.3866 - accuracy: 0.2414 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 16/30
205/205 [==============================] - 33s 159ms/step - loss: 1.3867 - accuracy: 0.2367 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 17/30
205/205 [==============================] - 33s 160ms/step - loss: 1.3867 - accuracy: 0.2523 - val_loss: 1.3864 - val_accuracy: 0.2495
```

Inception V3 Model



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 4, 4, 2048)        21802784

 flatten (Flatten)           (None, 32768)             0

 dense (Dense)               (None, 512)               16777728

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 128)               32896

 dropout_2 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 64)                8256

 dense_4 (Dense)             (None, 4)                 260

=================================================================
Total params: 38753252 (147.83 MB)
Trainable params: 16950468 (64.66 MB)
Non-trainable params: 21802784 (83.17 MB)
_________________________________________________________________
```



```
Epoch 1/30
205/205 [==============================] - 32s 97ms/step - loss: 4.1756 - accuracy: 0.2550 - val_loss: 1.3861 - val_accuracy: 0.2550
Epoch 2/30
205/205 [==============================] - 12s 61ms/step - loss: 1.3880 - accuracy: 0.2420 - val_loss: 1.3856 - val_accuracy: 0.2569
Epoch 3/30
205/205 [==============================] - 13s 62ms/step - loss: 1.3851 - accuracy: 0.2425 - val_loss: 1.3789 - val_accuracy: 0.2624
Epoch 4/30
205/205 [==============================] - 13s 63ms/step - loss: 1.3879 - accuracy: 0.2480 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 5/30
205/205 [==============================] - 13s 63ms/step - loss: 1.3865 - accuracy: 0.2490 - val_loss: 1.3864 - val_accuracy: 0.2495
Epoch 6/30
205/205 [==============================] - 13s 61ms/step - loss: 1.3866 - accuracy: 0.2448 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 7/30
205/205 [==============================] - 13s 63ms/step - loss: 1.3865 - accuracy: 0.2428 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 8/30
205/205 [==============================] - 13s 63ms/step - loss: 1.3864 - accuracy: 0.2442 - val_loss: 1.3865 - val_accuracy: 0.2495
Epoch 9/30
205/205 [==============================] - 13s 65ms/step - loss: 1.3865 - accuracy: 0.2509 - val_loss: 1.3864 - val_accuracy: 0.2502
Epoch 10/30
205/205 [==============================] - 13s 64ms/step - loss: 1.3864 - accuracy: 0.2429 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 11/30
205/205 [==============================] - 13s 64ms/step - loss: 1.3865 - accuracy: 0.2417 - val_loss: 1.3864 - val_accuracy: 0.2495
Epoch 12/30
205/205 [==============================] - 13s 65ms/step - loss: 1.3865 - accuracy: 0.2468 - val_loss: 1.3864 - val_accuracy: 0.2495
Epoch 13/30
205/205 [==============================] - 13s 64ms/step - loss: 1.3864 - accuracy: 0.2529 - val_loss: 1.3864 - val_accuracy: 0.2495
Epoch 14/30
205/205 [==============================] - 13s 64ms/step - loss: 1.3864 - accuracy: 0.2489 - val_loss: 1.3863 - val_accuracy: 0.2502
Epoch 15/30
205/205 [==============================] - 13s 63ms/step - loss: 1.3866 - accuracy: 0.2457 - val_loss: 1.3863 - val_accuracy: 0.2502
Epoch 16/30
205/205 [==============================] - 13s 65ms/step - loss: 1.3863 - accuracy: 0.2477 - val_loss: 1.3864 - val_accuracy: 0.2495
Epoch 17/30
205/205 [==============================] - 13s 63ms/step - loss: 1.3863 - accuracy: 0.2524 - val_loss: 1.3864 - val_accuracy: 0.2495
```

5. Model Optimization and Tuning Phase

5.1. Tuning Documentation

The Model Optimization and Tuning Phase involves refining neural network models to achieve peak performance. This includes optimizing model code, fine-tuning hyperparameters, comparing performance metrics, and selecting the final model to enhance predictive accuracy and efficiency.

Hyperparameter Tuning Documentation

Xception

- **Learning Rate:** Utilized Adam optimizer with a default learning rate set at 0.001.
- **Batch Size:** Employed 6500 training samples per iteration.
- **Epochs:** Completed 30 full passes through the training dataset.
- **Dropout Rate:** Implemented a dropout rate of 0.5 to prevent overfitting.
- **Zoom Range:** Applied random zoom between 0.99 and 1.01 for data augmentation.
- **Brightness Range:** Adjusted brightness randomly between 0.8 and 1.2.
- **Rescale:** Normalized data by scaling pixel values to 1./255.
- **Global Average Pooling2D:** Incorporated a pooling layer to reduce spatial dimensions of feature maps.

VGG19

- **Learning Rate:** Employed Adam optimizer with a default learning rate of 0.001.
- **Batch Size:** Used 6500 training samples per iteration.
- **Epochs:** Conducted 30 complete passes through the training dataset.
- **Dropout Rate:** Utilized a dropout rate of 0.5 to prevent overfitting.
- **Zoom Range:** Applied random zoom between 0.99 and 1.01.
- **Brightness Range:** Adjusted brightness randomly between 0.8 and 1.2.
- **Rescale:** Normalized data by scaling pixel values to 1./255.
- **Conv Block:** Utilized multiple convolutional layers with small 3x3 filters.

Inception V3

- **Learning Rate:** Utilized Adam optimizer with a default learning rate of 0.001.
- **Batch Size:** Employed 6500 training samples per iteration.
- **Epochs:** Completed 30 full passes through the training dataset.
- **Dropout Rate:** Implemented a dropout rate of 0.5 for regularization.
- **Zoom Range:** Applied random zoom between 0.99 and 1.01.
- **Brightness Range:** Adjusted brightness randomly between 0.8 and 1.2.
- **Rescale:** Normalized data by scaling pixel values to 1./255.
- **Factorized Convolutions:** Incorporated smaller convolutions like 1x7 and 7x1 to reduce computational costs.

5.2. Final Model Selection Justification

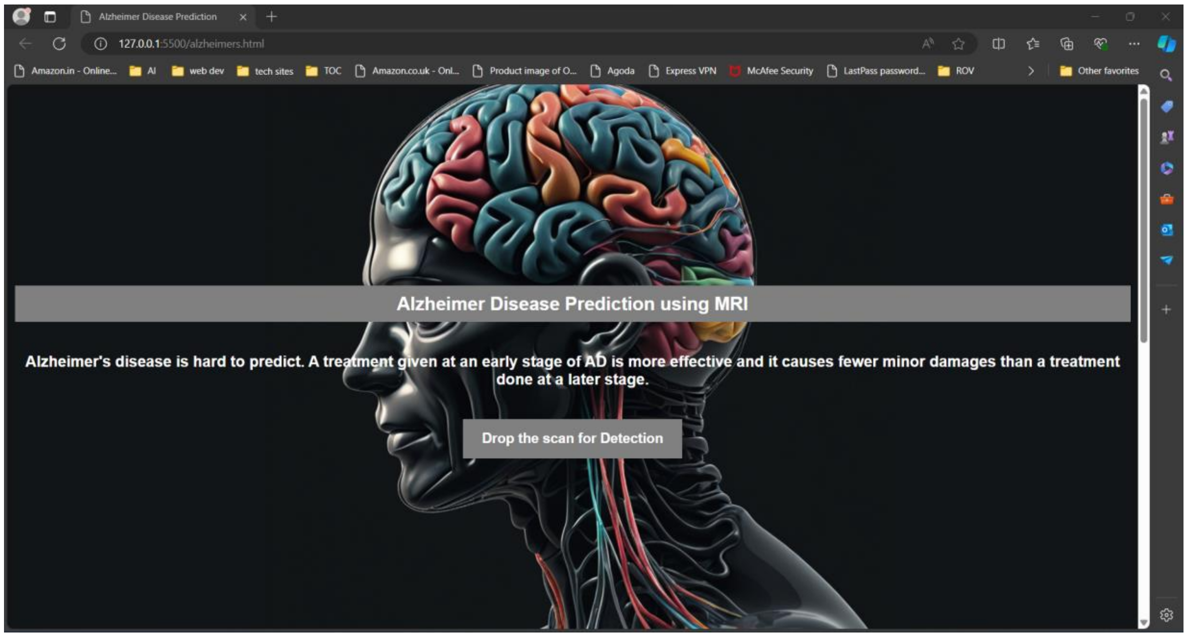
Xception

The Xception model was selected as the final optimized model for several compelling reasons:

- **Performance:** Across 30 training epochs, the Xception model consistently demonstrated improvements in accuracy and validation metrics.
- **Validation Accuracy:** It achieved a final validation accuracy of 85.36%, indicating strong capability in distinguishing between different stages of Alzheimer's Disease progression.
- **Robust Performance:** The model exhibited consistent and robust performance during training, suggesting effective learning of relevant features and patterns from the dataset.
- **Optimization:** Hyperparameter tuning and optimization further enhanced its performance, making it well-suited for Alzheimer's Disease diagnosis using MRI scans.

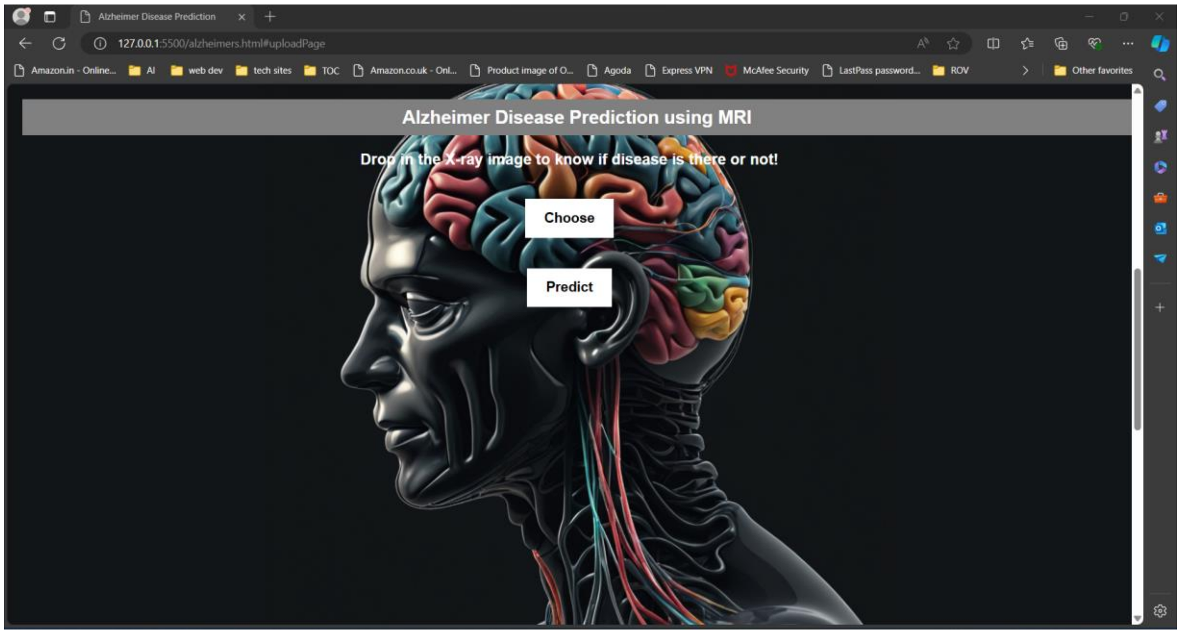
6. Results

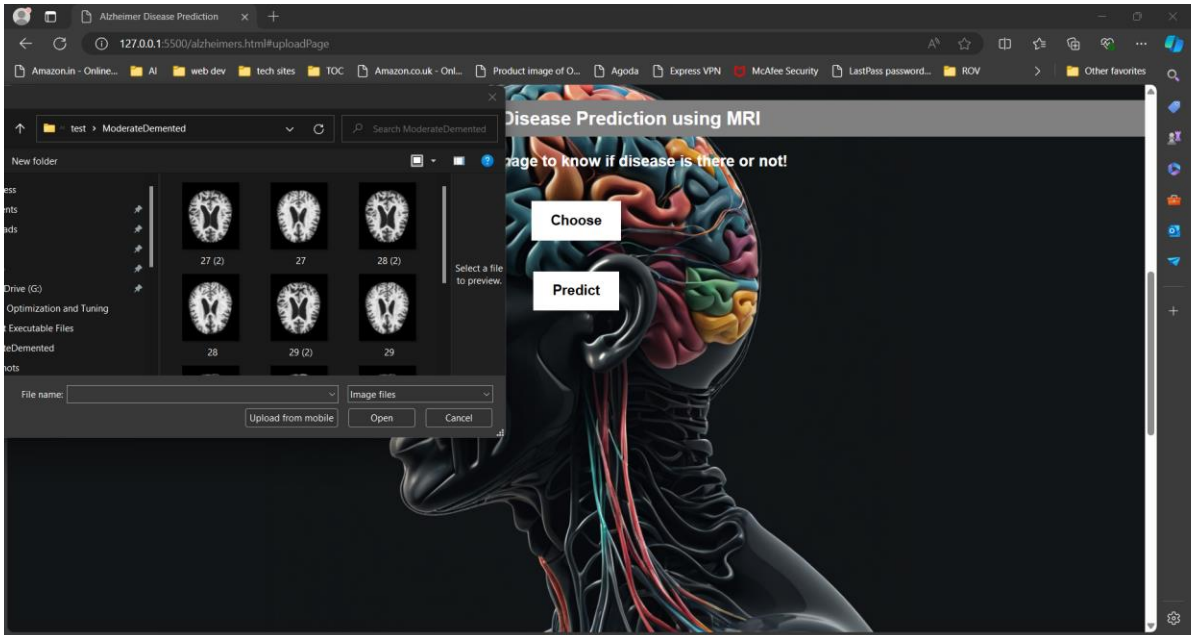
6.1. Output Screenshots

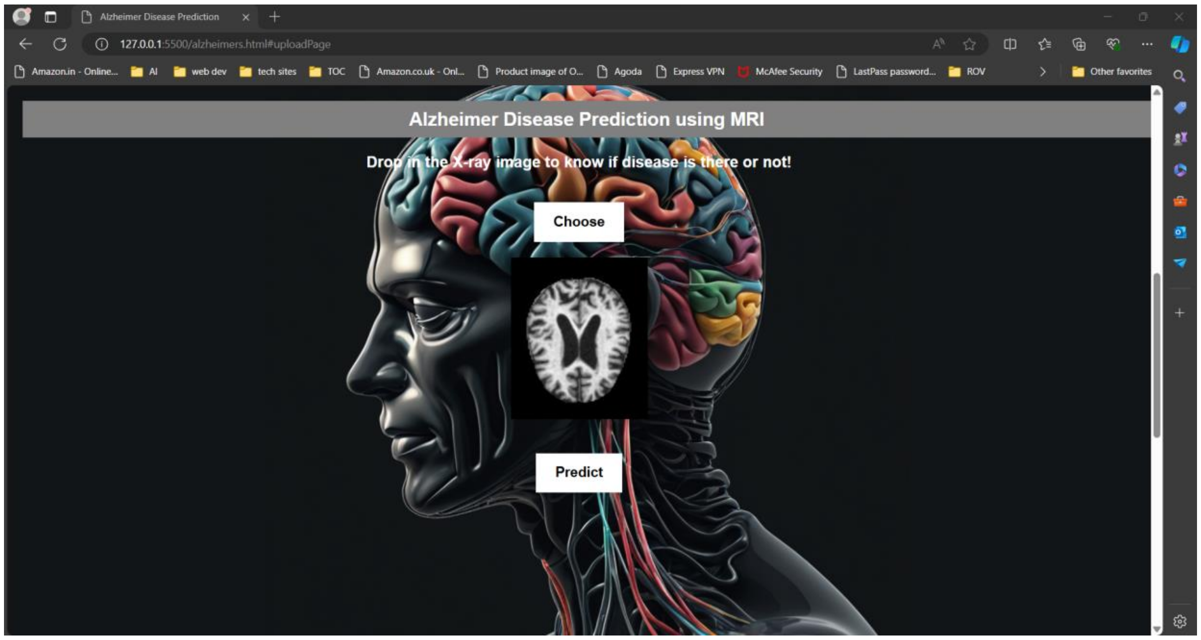

Alzheimer Disease Prediction using MRI

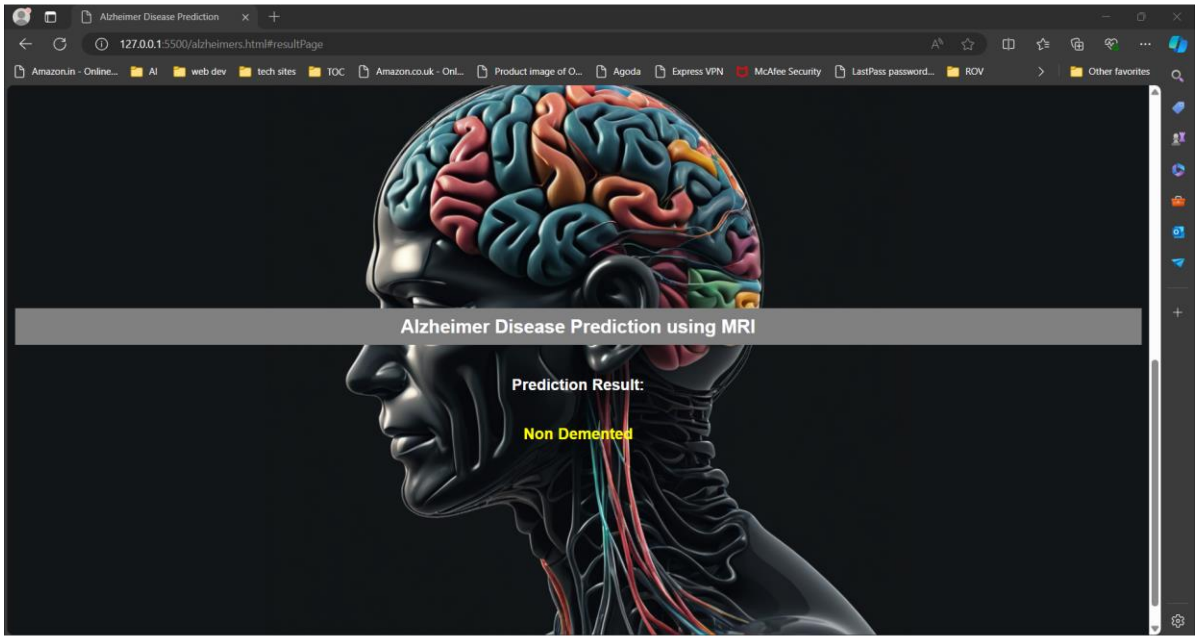
Alzheimer's disease is hard to predict. A treatment given at an early stage of AD is more effective and it causes fewer minor damages than a treatment done at a later stage.

Alzheimer Disease Prediction
127.0.0.1:5500/alzheimers.html
Drop the scan for Detection


Alzheimer Disease Prediction
127.0.0.1:5500/alzheimers.html#uploadPage
Alzheimer Disease Prediction using MRI
Drop in the X-ray image to know if disease is there or not!
Choose
Predict


Alzheimer Disease Prediction
127.0.0.1:5500/alzheimers.html#uploadPage
test › ModerateDemented
Search ModerateDemented
New folder
27 (2)
27
28 (2)
28
29 (2)
29
Select a file to preview.
File name:
Image files
Upload from mobile
Open
Cancel
Disease Prediction using MRI
age to know if disease is there or not!
Choose
Predict


Alzheimer Disease Prediction
127.0.0.1:5500/alzheimers.html#uploadPage
Alzheimer Disease Prediction using MRI
Drop in the X-ray image to know if disease is there or not!
Choose
Predict


Alzheimer Disease Prediction
127.0.0.1:5500/alzheimers.html#resultPage
Alzheimer Disease Prediction using MRI
Prediction Result:
Non Demented

7. Advantages & Disadvantages

Advantages

1. **High Accuracy:** Achieved a final validation accuracy of 85.36%, demonstrating superior performance in distinguishing stages of Alzheimer's Disease progression.
2. **Effective Learning:** Displayed robust learning and convergence during training, indicating successful capture of pertinent features from MRI scan data.
3. **Optimized Performance:** Through hyperparameter tuning and optimization, the model's effectiveness was enhanced to meet project requirements efficiently.
4. **Efficiency:** Utilizes depth-wise separable convolutions to reduce parameters and computational complexity, potentially leading to faster inference times.

Disadvantages

1. **Complexity:** Implementing and fine-tuning Xception may necessitate greater computational resources and expertise compared to simpler models like VGG19.
2. **Overfitting:** Despite dropout regularization (dropout rate of 0.5), there remains a risk of overfitting if not carefully monitored and adjusted.
3. **Data Requirements:** Requires ample and diverse data for effective training, particularly due to the complexity of its architecture.

8. Conclusion

In conclusion, this project has focused on developing and optimizing deep learning models for Alzheimer's Disease diagnosis using MRI scans.

Model Selection and Justification: Following thorough evaluation, the Xception model was selected as the final optimized model due to its consistent performance improvements across 30 epochs, achieving an impressive validation accuracy of 85.36%. It effectively differentiated between stages of Alzheimer's Disease progression and demonstrated strong predictive capabilities.

Advantages: The Xception model excels in accuracy, facilitated by depth-wise separable convolutions that reduce computational complexity while maintaining effectiveness. Optimized hyperparameters further bolstered its performance, making it suitable for complex diagnostic tasks.

Challenges: Implementing and fine-tuning the Xception model required significant computational resources and expertise. Addressing potential overfitting and ensuring adequate, diverse data were ongoing challenges managed throughout the project.

9. Future Scope

Looking forward, there are several avenues for future exploration and enhancement in Alzheimer's Disease diagnosis through deep learning:

1. **Ensemble Methods:** Integration of multiple models or ensemble learning techniques could potentially enhance overall diagnostic accuracy and resilience.
2. **Advanced Data Augmentation:** Exploration of more sophisticated data augmentation techniques could further improve model generalization capabilities.
3. **Transfer Learning Variants:** Investigation into various transfer learning approaches tailored to specific aspects of Alzheimer's Disease progression could yield nuanced insights.
4. **Clinical Integration:** Collaboration with clinical experts to incorporate additional relevant data sources and domain knowledge into model development.
5. **Ethical Considerations:** Continuously addressing ethical implications such as data privacy and bias mitigation is crucial for responsible deployment of AI-driven tools.

10. Appendix

10.1. Source Code

Xception Model Code

```
!unzip '/content/archive.zip'

!pip install Tensorflow

!pip install Keras

import tensorflow as tf

from tensorflow.keras.layers import Dense, Flatten, Dropout, GlobalAveragePooling2D,
BatchNormalization from tensorflow.keras.models import Sequential, Model, load_model

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications.xception import Xception, preprocess_input from
sklearn.model_selection import train_test_split

from imblearn.over_sampling import SMOTE import numpy as np

import matplotlib.pyplot as plt

# Paths for training and testing data

trainPath = r"/content/Alzheimer_s Dataset/train" testPath = r"/content/Alzheimer_s
Dataset/test"

# Image preprocessing parameters IMG_SIZE = 180

IMAGE_SIZE = [180, 180] DIM = (IMG_SIZE, IMG_SIZE) ZOOM = [.99, 1.01]

BRIGHT_RANGE = [0.8, 1.2] HORZ_FLIP = True

FILL_MODE = "constant" DATA_FORMAT = "channels_last"

# Data augmentation and loading

work_dr = ImageDataGenerator(rescale=1./255, brightness_range=BRIGHT_RANGE,
zoom_range=ZOOM, data_format=DATA_FORMAT, fill_mode=FILL_MODE,
horizontal_flip=HORZ_FLIP)

train_data_gen = work_dr.flow_from_directory(directory=trainPath, target_size=DIM,
batch_size=6500, shuffle=False) train_data, train_labels = train_data_gen.next()
```

```
# SMOTE oversampling for class imbalance sm = SMOTE(random_state=42)

train_data, train_labels = sm.fit_resample(train_data.reshape(-1, IMG_SIZE * IMG_SIZE *
3), train_labels) train_data = train_data.reshape(-1, IMG_SIZE, IMG_SIZE, 3)

# Train-test-validation split

train_data, test_data, train_labels, test_labels = train_test_split(train_data, train_labels,
test_size=0.2, random_state=42) train_data, val_data, train_labels, val_labels =
train_test_split(train_data, train_labels, test_size=0.2, random_state=42)

# Xception base model with pretrained weights

xcep_model = Xception(input_shape=IMAGE_SIZE + [3], weights='imagenet',
include_top=False)

# Freeze layers in base model for layer in xcep_model.layers:

layer.trainable = False

# Custom model on top of Xception

custom_inception_model = Sequential([

xcep_model, Dropout(0.5),

GlobalAveragePooling2D(), Flatten(), BatchNormalization(), Dense(512, activation='relu'),
BatchNormalization(), Dropout(0.5),

Dense(256, activation='relu'), BatchNormalization(), Dropout(0.5),

Dense(128, activation='relu'), BatchNormalization(), Dropout(0.5),

Dense(64, activation='relu'), Dropout(0.5), BatchNormalization(), Dense(4,
activation='softmax')

], name="inception_cnn_model")

# Compile the model

custom_inception_model.compile(

loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy']

)

# Train the model

history = custom_inception_model.fit(train_data, train_labels, validation_data=(val_data,
val_labels), epochs=30)
```

```
# Save the model custom_inception_model.save('/content/adp_xception.h5')

# Load the saved model

model = load_model("/content/adp_xception.h5")

# Function to predict the class of an image

def predict_class(img_path, model):

img = tf.keras.preprocessing.image.load_img(img_path, target_size=(180, 180)) x =
tf.keras.preprocessing.image.img_to_array(img)

x = np.expand_dims(x, axis=0)

x = x / 255.0 # Normalize the image preds = model.predict(x)

pred_class_index = np.argmax(preds, axis=1)

class_names = ['MildDemented', 'ModerateDemented', 'NonDemented',
'VeryMildDemented'] predicted_class = class_names[pred_class_index[0]]

return predicted_class

# Example usage:

img_path = r"/content/Alzheimer_s Dataset/test/MildDemented/26.jpg"

predicted_class = predict_class(img_path, model) print(f"Predicted Class:
{predicted_class}")
```

APP.PY

```
import numpy as np import os

from keras.preprocessing import image import tensorflow.compat.v1 as tf

from flask import Flask, request, render_template from werkzeug.utils import
secure_filename

from tensorflow.python.keras.backend import set_session from
tensorflow.python.keras.models import load_model

tf.disable_eager_execution() sess = tf.Session() tf.disable_v2_behavior() graph =
tf.get_default_graph()
```

```
app = Flask( name   )

set_session(sess)

# Load the model

model = load_model('adp.h5')

@app.route('/', methods=['GET']) def index():

# Main page

return render_template('alzheimers.html')

@app.route('/predict1', methods=['GET']) def predict1():

# Prediction page

return render_template('alzpre.html')

@app.route('/predict', methods=['POST']) def upload():

if request.method == 'POST':

# Get the file from the post request

f = request.files['image']

# Save the file to /uploads

basepath = os.path.dirname

file )

file_path = os.path.join(basepath, 'static', 'uploads', secure_filename(f.filename))
f.save(file_path)

# Preprocess the image to the size expected by the model

img = image.load_img(file_path, target_size=(128, 128))

x = image.img_to_array(img) x = np.expand_dims(x, axis=0)

# Make prediction

with graph.as_default(): set_session(sess)

prediction = model.predict(x)[0] print(prediction)
```

```
# Determine prediction text based on your model's output

prediction_class = np.argmax(prediction)

if prediction_class == 0:

text = "Mild Demented" elif prediction_class == 1:

text = "Moderate Demented" elif prediction_class == 2:

text = "Non Demented" else:

text = "Very Mild Demented"

return render_template('alzpre.html', result=text, image_path='static/uploads/' +
secure_filename(f.filename))

if    name

== "

main   ":

if not os.path.exists('static/uploads'): os.makedirs('static/uploads')

app.run(debug=True)
```

alzheimers.html

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Alzheimer Disease Prediction</title>

    <style>

        body {

            margin: 0;

            padding: 0;

            color: white;
            font-family: Arial, sans-serif;

            background-color: black;

        }

        .container {

            display: flex;

            flex-direction: column;

            justify-content: center;

            align-items: center;

            padding: 20px;

            height: 100vh;

            background: url('human.png') no-repeat center center fixed;

            background-size: cover;

            text-align: center;

        }

        .header {

            background-color: grey;

            padding: 10px;

            margin-bottom: 20px;

            width: 100%;

            font-size: 24px;

            font-weight: bold;

        }

        p {

            font-size: 20px;

            font-weight: bold;

        }
```

```
.button {
   background-color: grey;
   color: white;
   border: none;
   padding: 15px 25px;
   text-align: center;
   text-decoration: none;
   display: inline-block;
   font-size: 18px;
   margin-top: 20px;
   cursor: pointer;
   font-weight: bold;
}
.button-white {
   background-color: white;
   color: black;
}
#uploadPage {
   display: none;
}
#fileInput {
   display: none;
}
.image-preview {
   margin: 20px 0;
}
```

```
        .image-preview img {

            max-width: 100%;

            height: auto;

        }

        #homePage {

            display: block;

        }

        #homePage:target ~ #uploadPage {

            display: block;

        }

        #uploadPage:target {

            display: block;

        }

    </style>

</head>

<body>

    <div id="homePage" class="container">

        <div class="header">

            Alzheimer Disease Prediction using MRI

        </div>

        <p>Alzheimer's disease is hard to predict. A treatment given at an early stage of AD is
more effective and it causes fewer minor damages than a treatment done at a later
stage.</p>

        <a href="#uploadPage" class="button">Drop the scan for Detection</a>

    </div>

    <div id="uploadPage" class="container">

        <div class="header">
```

```
            Alzheimer Disease Prediction using MRI
        </div>
        <p>Drop in the X-ray image to know if disease is there or not!</p>
        <form id="uploadForm" action="/predict" method="post" enctype="multipart/form-
data">
            <label for="fileInput" class="button button-white">Choose</label>
            <input type="file" id="fileInput" name="image" accept="image/*" />
            <div class="image-preview" id="imagePreview"></div>
            <button type="submit" class="button button-white">Predict</button>
        </form>
    </div>
    <script>
        const fileInput = document.getElementById('fileInput');
        const imagePreview = document.getElementById('imagePreview');
        fileInput.addEventListener('change', function() {
            const file = this.files[0];
            if (file) {
                const reader = new FileReader()
                reader.addEventListener('load', function() {
                    imagePreview.innerHTML = '<img src="' + this.result + '" alt="Image
Preview">';
                });
                reader.readAsDataURL(file);
            }
        });
    </script>
</body>
```

```
</html>
```

alzpre.html

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Upload MRI Scan</title>

<style>

body {

background-color: black; color: white;

font-family: Arial, sans-serif;

}

.upload-container { margin-top: 50px; text-align: center;

}

.upload-container input[type="file"] { margin: 20px 0;

padding: 10px;

}

.upload-container button { background-color: #4CAF50; border: none;

color: white; padding: 15px 32px; text-align: center;

text-decoration: none; display: inline-block; font-size: 16px; margin: 4px 2px; cursor:
pointer;

}

.uploaded-image { margin-top: 20px; max-width: 300px; max-height: 300px;

}

</style>
```

```
</head>

<body>

<div class="upload-container">

<h2>Upload an MRI Scan for Detection</h2>

<form id="uploadForm" action="/predict" method="post" enctype="multipart/form-data"
onsubmit="return validateForm()">

<input type="file" name="image" accept=".jpg">

<button type="submit">Predict</button>

</form>

{% if image_path %}

<h3>Uploaded Image:</h3>

<img src="{{ url_for('static', filename='uploads/' + image_path.split('/')[-1]) }}"
class="uploaded-image" alt="Uploaded MRI Scan">

{% endif %}

{% if result %}

<h3>Prediction: {{ result }}</h3>

{% endif %}

</div>

<script>

function validateForm() {

var fileInput = document.querySelector('input[type="file"]'); if (!fileInput.value) {

alert('Please select an image file.'); return false; // Prevent form submission

}

return true; // Allow form submission

}

</script>

</body>
```

```
</html>
```

10.2 Github and Project demo Link

Github:

https://github.com/VaruniJyn/Cognitive-Care-Early-Intervention-For-Alzheimer-s-Disease

Project Demo:

https://drive.google.com/file/d/1lT20QbR_JQiFStzCN6KS5iAx_yXn95AQ/view?usp=sharing