Project Phase - 02 Final Review

Detection of Alzheimer disease using Modified CNN Model

Group No : 16
Guide: Dr. K L Nisha,
Assistant Professor,
Department of Electronics and Communication Engineering,
Amrita School of Engineering, Amritapuri, India

Team Members	
B. Shahid	AM.EN.U4EAC21020
Janupally Akshaya Reddy	AM.EN.U4EAC21033
Malavika LS	AM.EN.U4EAC21047
P Nitin Balaram Varma	AM.EN.U4EAC21058
Sri Pranav P	AM.EN.U4ECE21069

Contents

Problem Statement

Existing methods for Alzheimer's Disease (AD) detection lack accuracy and efficiency, hindering timely intervention. This project addresses the need for a reliable early detection system using cognitive features extracted from brain imaging data. Leveraging Convolutional Neural Networks (CNNs), MobileNet as well as DenseNet alongside ensemble learning, the aim is to develop a robust classification model capable of accurately identifying individuals at risk of AD at earlier stages.

Project Objective

- To preprocess the brain imaging dataset, ensuring data quality by handling missing values, normalizing the dataset, and applying suitable transformations to prepare the data for effective model training.
- To design and develop multiple deep learning architectures, such as Convolutional Neural Networks (CNNs) along with other advanced neural network models and train them on the preprocessed dataset to extract key cognitive features for accurate Alzheimer's Disease detection.
- To combine the trained deep learning models into a modified ensemble model, leveraging the strengths of each individual architecture for enhanced prediction accuracy and robustness in classifying healthy, mild cognitive impairment (MCI), and Alzheimer's Disease (AD) cases.
- To analyze brain imaging data using the ensemble architecture, predicting whether each case falls into HC, MCI, or AD categories, and evaluating the model's performance in terms of accuracy, precision, recall, and other relevant metrics

Summary of Literature Review

S No	Paper	Methods Used	What is addressed
1	Early diagnosis of Alzheimer's disease using machine learning: a multi-diagnostic, generalizable approach	Traditional machine learning classifiers (SVM, Decision Trees, Random Forest) for classification, handcrafted features (hippocampal volume, cortical thickness, etc.) for training the model.	We replace these machine learning models with deep learning-based CNN architectures, which can automatically learn features from images,
2	A Review on Alzheimer's Disease Through Analysis of MRI Images Using Deep Learning Techniques	Convolutional Neural Networks (CNNs), pattern recognition techniques, and multi-modal learning approaches for MRI-based Alzheimer's disease classification and brain structure segmentation.	Deep learning techniques for analyzing MRI images to enhance the early detection and classification of Alzheimer's disease (AD). It discusses brain MRI segmentation methods, and the impact of CNNs and multi-modal learning on improving AD detection accuracy.
3	Computer-Aided Diagnosis System of Alzheimer's Disease Based on Multimodal Fusion: Tissue Quantification Based on the Hybrid Fuzzy-Genetic-Possibilistic Model and Discriminative Classification Based on the SVDD Model	The paper used SVDD (Support Vector Data Description) as the final classifier after multimodal fusion, focused more on clustering and rule-based segmentation methods, which can be slow.	Instead of SVDD, we are using a deep learning-based ensembling technique, where multiple models are trained and combined at the end, to help with accuracy as well as increase optimization .
4	Diagnosis and Detection of Alzheimer's Disease Using Learning Algorithm	Random Forest XG Boost Convolutional Neural Networks	Emphasizes preprocessing techniques to enhance classification performance, reducing training time and improving Alzheimer's detection accuracy.
5	A multimodal deep learning approach for the prediction of cognitive decline and its effectiveness in clinical trials for Alzheimer's disease	Methods used: DenseNet-121, Support Vector Regression (SVR) V-Net, Faster R-CNN	The biased allocation in Alzheimer's clinical trials by proposing an AI-based stratified randomization method that uses predicted cognitive decline to improve participant distribution

Existing Method

Traditional methods for diagnosing Alzheimer's Disease rely on clinical assessments, cognitive tests, and neuroimaging techniques such as MRI and PET scans. However, these methods often lack the sensitivity and specificity required for early detection. Additionally, they are subjective, costly, and not readily accessible to all individuals. This project aims to overcome these limitations by leveraging machine learning algorithms to develop a more accurate and efficient early detection system.

Disadvantages:

- Lack of Sensitivity: Traditional diagnostic methods for Alzheimer's Disease often lack the sensitivity to detect subtle cognitive changes in the early stages of the disease, leading to delayed diagnosis and intervention.
- Subjectivity: Clinical assessments and cognitive tests rely on subjective interpretation by healthcare professionals, introducing variability and potential biases in diagnosis.
- Cost and Accessibility: Neuroimaging techniques such as MRI and PET scans are costly and may not be readily accessible to all individuals, particularly in resource-limited settings, limiting widespread adoption for early detection purposes.

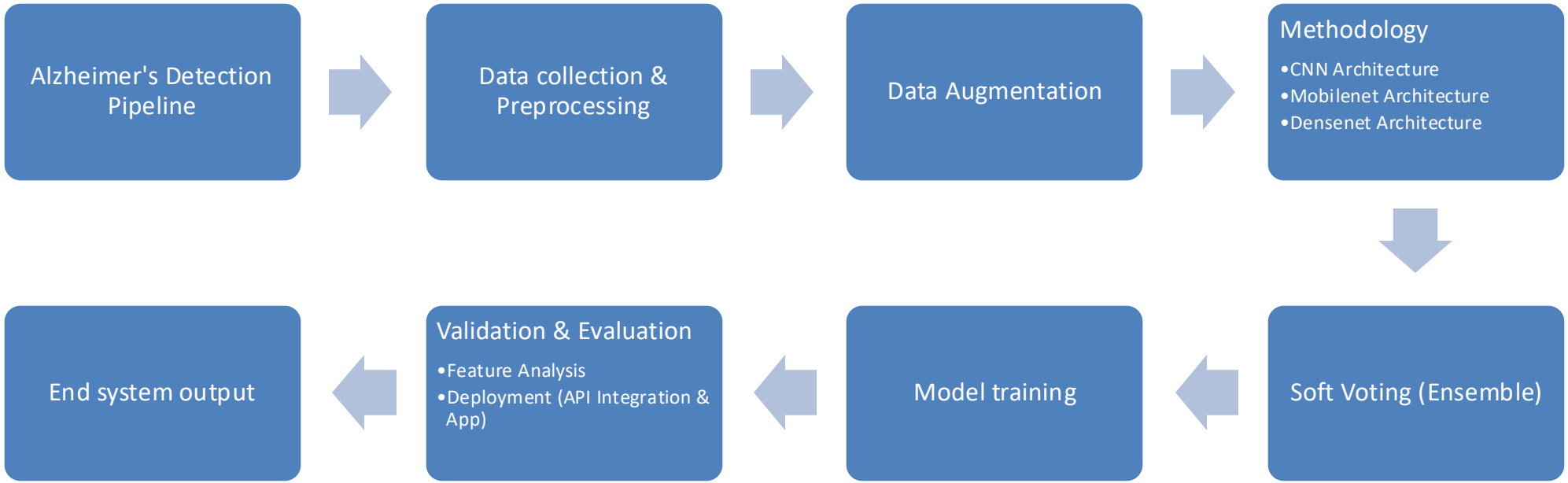
Proposed Method

The proposed system leverages Convolutional Neural Networks (CNNs) to extract cognitive features from brain imaging data for early detection of Alzheimer's Disease. A CNN model, MobileNet and an additional DenseNet are trained independently, and their predictions are combined using ensemble learning techniques(Soft Voting) for improved accuracy. By utilizing deep learning and ensemble methods, the system aims to overcome the limitations of existing diagnostic methods, offering a more efficient and reliable approach for early detection of AD with enhanced sensitivity and specificity.

Advantages:

- Enhanced Sensitivity: The proposed system leverages Convolutional Neural Networks (CNNs) to extract subtle cognitive features from brain imaging data, enhancing sensitivity for early detection of Alzheimer's Disease compared to traditional methods.
- Improved Accuracy: Ensemble learning techniques are employed to combine the predictions of multiple models, leading to improved classification accuracy and robustness in identifying individuals at risk of AD at earlier stages.
- Cost-Efficiency: By automating the analysis of brain imaging data using machine learning algorithms, the proposed system offers a cost-effective alternative to traditional neuroimaging techniques, making early detection more accessible to a broader population.

Block Diagram



CNN Architecture

- **Input Layer**:
  Takes an image as input.
- **Feature Extraction**:
  Convolution Layers: Apply filters to extract features like edges & textures,3 convolutional layers for our model,kernel size 3*3
  Pooling Layers: Reduce spatial dimensions while preserving important features,3 maxpooling layers, of size 2*2 and stride 2.
  Process repeats to extract deeper representations.
- **Classification**:
  Flattened features pass through **Fully Connected Layers,** where it has 1024 neurons now
- **Activation Functions** (ReLU, Softmax) aid decision-making.
  Final layer outputs class probabilities.

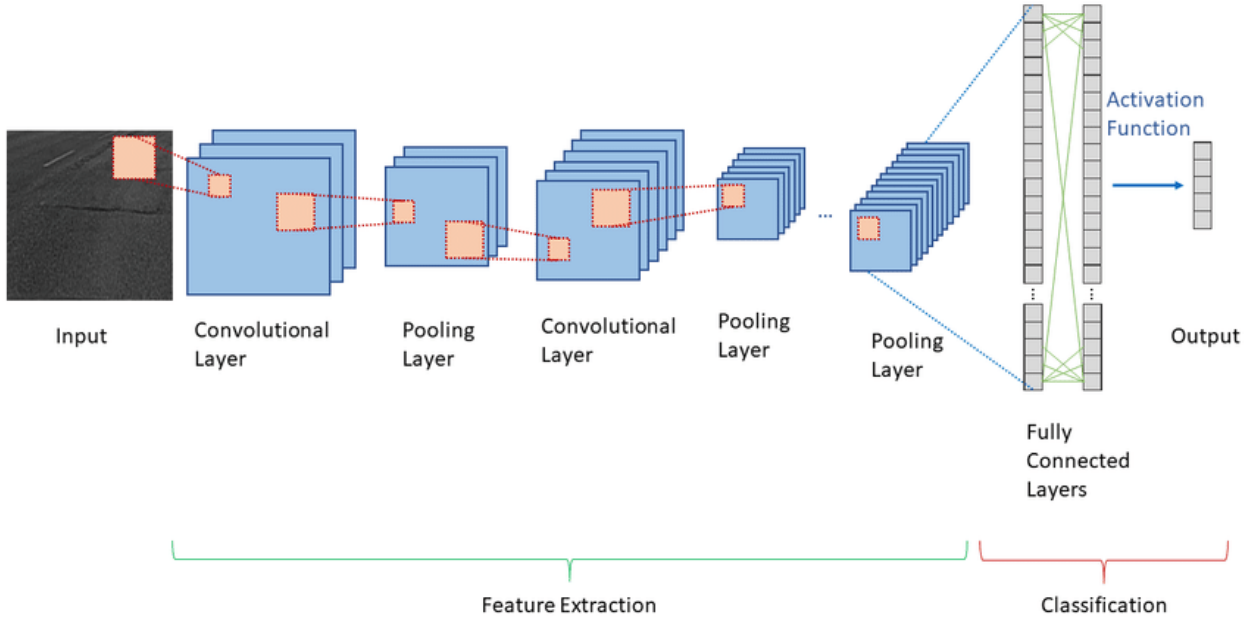


Fig.1 CNN Architecture

MobileNet Architecture

- **Feature Extraction (Depthwise Separable Convolutions + Pooling)**:
- Uses **Depthwise Separable Convolutions** instead of standard convolutions to reduce computation(1 Standard convolution layer(kernel size 3*3), 13 depthwise convolution layers(3 kernels of size 3*3,stride 1).
- ReLU activation used to introduce non linearity.
- **Global Average Pooling** replaces traditional fully connected layers, reducing the number of trainable parameters,
- **Fully Connected Layers + Softmax Classifier**:
- The extracted features are fed into a classifier that assigns probabilities to different classes.

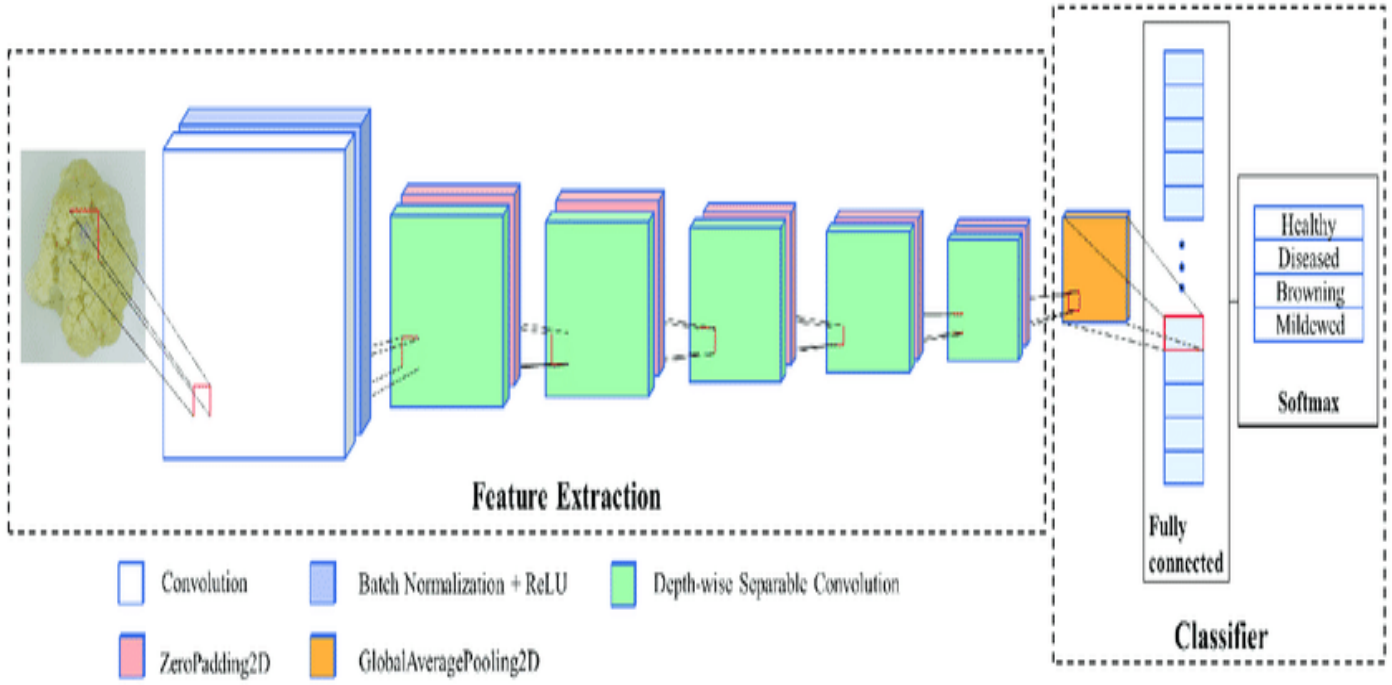


Fig.2 Mobilenet Architecture

DenseNet Architecture

- **Input Layer**:
  Takes an image of size **176×176×3** as input.
- **Feature Extraction**:
- **Base Model:** DenseNet121 (Pretrained on ImageNet)

DenseNet uses *dense connections* where each layer receives input from all previous layers, improving feature reuse and gradient flow.

↳ **Frozen parameters:** 6,956,979 (non-trainable in this setup)

↳ 4 dense blocks with transition layers (convolution + pooling).

GlobalAveragePooling2D: Converts spatial feature maps into a 1D vector (output shape: (1024,)).

- **Classification:** Flattened features pass through a Dense layer with 1024 neurons (ReLU, 1,049,600 parameters), followed by a Dropout layer (rate 0.3), and a final Dense layer with 4 output classes (Softmax, 4,100 parameters) to output class probabilities.
- **Activation Functions:**
  ↳ ReLU used after Dense layer for non-linearity.
  ↳ `Softmax` in the final layer for multi-class classification.

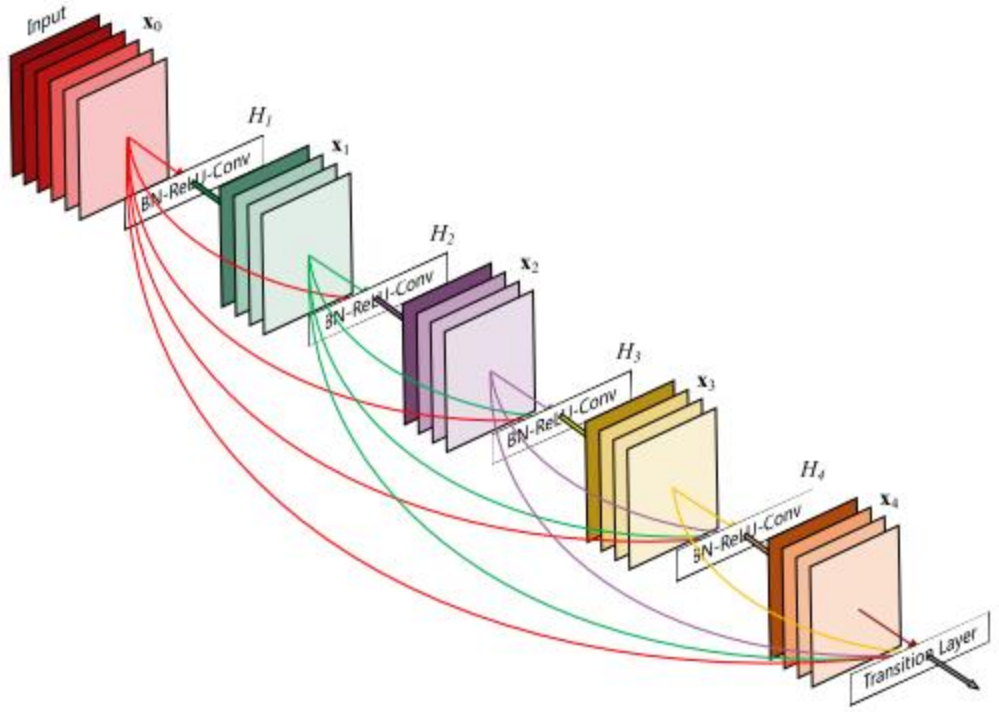


Fig.1 Densenet Architecture

Dataset

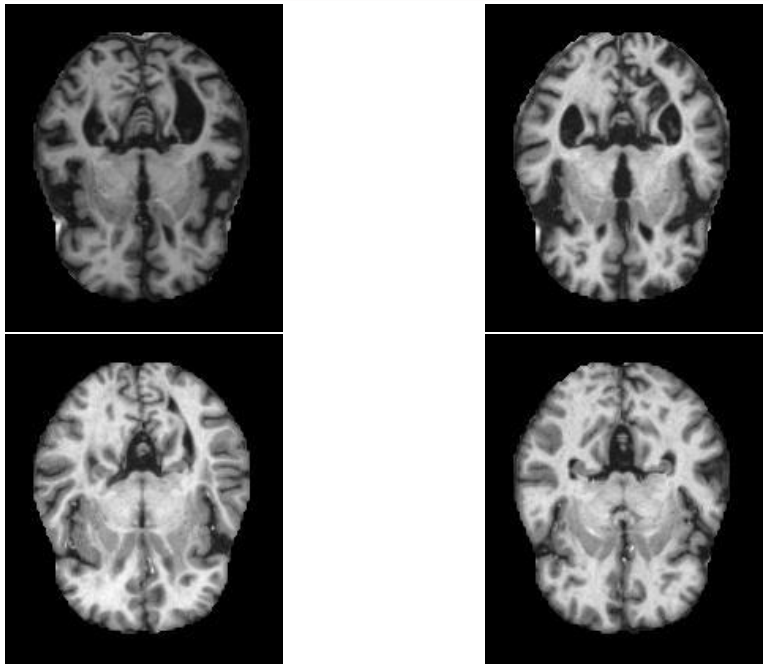


Fig.3 Training dataset (a) Mild Demented (b) Moderate Demented (c) Non Demented (d) Very Mild Demented.
Total Mild Demented images:1700
Total Moderate Demented images: 50
Total Non Demented images: 2500
Total Very Mild Demented images: 1000

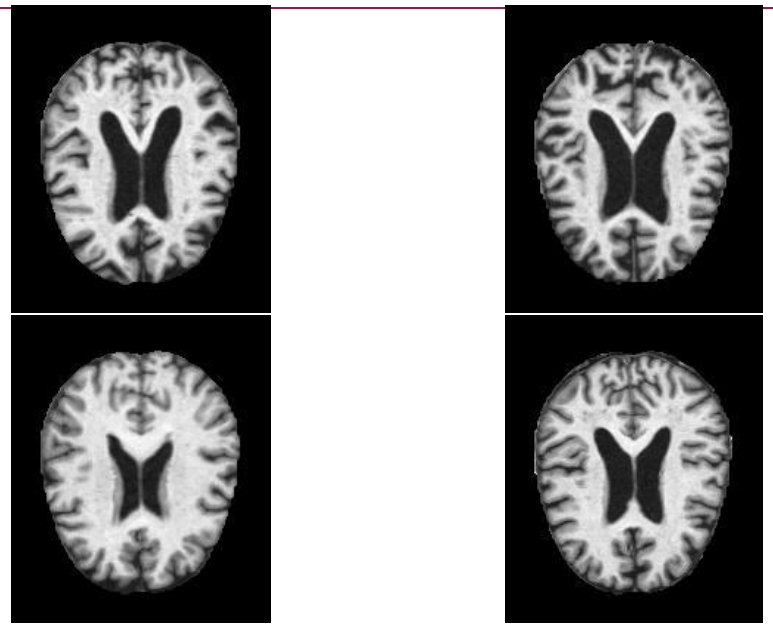


Fig.4 Test dataset (a) Mild Demented (b) Moderate Demented (c) Non Demented (d) Very Mild Demented
Total Mild Demented images:70
Total Moderate Demented images: 11
Total Non Demented images: 100
Total Very Mild Demented images: 100

Preprocessing

- **Image Resizing:** All images are resized to **(176 × 176)** pixels to maintain uniform input shape using the flow-from-data frame function. It loads images in batches and automatically resizes images.

•**Rescaling:** Images are rescaled using Rescale=1/255, which will normalize the scale to [0,1] range, helping in faster training by preventing larger weights.

- **SMOTE (Synthetic Minority Over-sampling Technique):** Used for imbalanced datasets. Finds the class with the least number of samples and using k nearest Neighbour, creates synthetic samples, and repeats until class is balanced.

- **Flattening and Shuffling :** Using `batch_size=6500`, loading all ~6500 images in one big batch. Shuffle=False set to keep the original order of the dataset to preserve the order of labels and images to facilitate SMOTE, after which Shuffle is set to TRUE before training.

Implementation Algorithm

- **Input:** Raw MRI images labeled as Non-Demented, Very Mild Demented, Mild Demented, or Moderate Demented.

- **Dataset Preparation:** Load the structural MRI image dataset containing the four classes for training, Non Demented(2500 images),Very Mild Demented (1000 images),Mild Demented (1700 images),Moderate Demented (50 images);

- **Data Preprocessing:** Resize all images to a fixed size of 176 × 176 pixels, Normalize image pixel values by applying `Rescale = 1/255,` and set `shuffle = False` to maintain the order for label association before SMOTE; apply SMOTE to the flattened dataset using **k-nearest neighbors** to generate synthetic examples for underrepresented classes

- **Model Training:**

1. CNN model: Designed to extract low- to mid-level spatial features from MRI images.

- Layers act as feature extractors; identifying edges, textures, and other patterns relevant to cognitive degeneration.
- Output from the final convolutional block is flattened and passed to a fully connected Dense layer with 1024 neurons.
- ReLU (Rectified Linear Unit) activations are used in all hidden layers to introduce non-linearity.
- SoftMax function in the output layer provides probability distributions over the four Alzheimer's classes.

Implementation Algorithm

2. MobileNet model: employs Depthwise Separable Convolutions

- Instead of applying convolution filters across all input channels, MobileNet performs depthwise convolutions
- Then performs pointwise convolutions (1×1 convolutions) to combine the outputs.
- Architecture includes 13 depthwise convolution layers, all with 3×3 kernels and stride 1
- A **Global Average Pooling layer** replaces the traditional dense layers to average all spatial information across each feature map
- Final **Softmax classifier** provides the predicted probability for each class.
- **ReLU** activation is used throughout the model.

3. DenseNet Model: DenseNet121 is a deep convolutional neural network known for its **dense connectivity pattern**, where each layer receives input from all preceding layers within a dense block.

- Encourages **feature reuse**, improves gradient flow during backpropagation, and alleviates the vanishing gradient problem.
- Takes an input of size **176×176×3** and uses a **pre-trained DenseNet121 backbone**
- Comprises of **4 dense blocks** with layers that include convolution and pooling.
- These layers gradually refine features by combining low- and high-level information.
- Uses **GlobalAveragePooling2D**, followed by a fully connected **Dense layer with 1024 neurons** and **ReLU activation** for non-linearity and a **Dropout layer with a rate of 0.3**
- Final **Dense layer with 4 neurons** and **Softmax activation** generates the class probabilities.

Implementation Algorithm

- **Model Evaluation:** Evaluate each trained model using the **validation and test sets**.

- **Obtain Softmax outputs from Each Model:** Each model (CNN, MobileNet, DenseNet) outputs a **Softmax probability vector of length 4**, representing predicted probabilities for the four classes

•Ensemble Prediction:

•Apply **Soft Voting Ensemble**:

- Predict class probabilities for the test image using CNN, MobileNet, and DenseNet models:
- `P_ensemble = (P_cnn + P_mobilenet + P_densenet) / 3;`
- `Class = argmax(P_ensemble)`

- **Map index to class label:** The predicted index is mapped to the actual disease label using the predefined label mapping

Code Implementation

```python
def create_cnn_model():
    model = keras.models.Sequential([
        keras.layers.Conv2D(32, kernel_size=(3,3), strides=2, padding='same', activation='relu', input_shape=(176,176,3)),
        keras.layers.MaxPooling2D(pool_size=(2,2), strides=2, padding='same'),
        keras.layers.Conv2D(64, kernel_size=(3,3), strides=2, activation='relu', padding='same'),
        keras.layers.MaxPooling2D(pool_size=(2,2), strides=2, padding='same'),
        keras.layers.Conv2D(128, kernel_size=(3,3), strides=2, activation='relu', padding='same'),
        keras.layers.MaxPooling2D(pool_size=(2,2), strides=2, padding='same'),
        keras.layers.Flatten(),
        keras.layers.Dense(1024, activation='relu'),
        keras.layers.Dropout(0.3),
        keras.layers.Dense(4, activation='softmax')
    ])
    return model
```

```python
# Instantiate multiple instances of the CNN model
num_models = 4
cnn_models = [create_cnn_model() for _ in range(num_models)]

checkpoint_cb = checkpoint_cb = ModelCheckpoint("model_{epoch:02d}.keras", save_best_only=True)
early_stopping_cb = EarlyStopping(patience=10, restore_best_weights=True)
```

```
C:\Users\Lenovo\anaconda3\anaconda_new\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`i
nput_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

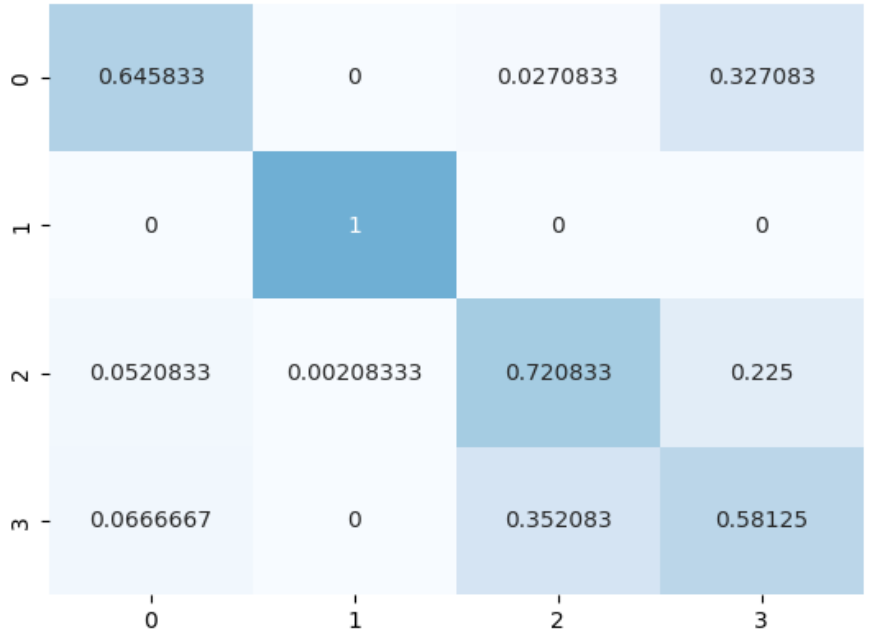
```python
# Train each CNN model and save them
for i, model in enumerate(cnn_models):
    print(f"Training and saving CNN Model {i+1}/{num_models}")
    # Train the model...
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    hist = model.fit(X_train,y_train, epochs=30, validation_data=(X_val,y_val), callbacks=[checkpoint_cb, early_stopping_cb])
    # Save the model
    model.save(f"model_{i+1}.h5")
```

Fig.5 Creating and training CNN model

Results

```
CM = confusion_matrix(y_test_,y_pred)
CM_percent = CM.astype('float') / CM.sum(axis=1)[:, np.newaxis]
sns.heatmap(CM_percent,fmt='g',center = True,cbar=False,annot=True,cmap='Blues')
CM
```

```
array([[310,   0,  13, 157],
       [  0, 480,   0,   0],
       [ 25,   1, 346, 108],
       [ 32,   0, 169, 279]], dtype=int64)
```




```
ClassificationReport = classification_report(y_test_,y_pred)
print('Classification Report is : ', ClassificationReport )
```

```
Classification Report is :                precision    recall  f1-score   support

           0       0.84      0.65      0.73       480
           1       1.00      1.00      1.00       480
           2       0.66      0.72      0.69       480
           3       0.51      0.58      0.54       480

    accuracy                           0.74      1920
   macro avg       0.75      0.74      0.74      1920
weighted avg       0.75      0.74      0.74      1920
```

Fig.6 (a) Confusion Matrix of CNN model (b) Classification Report of CNN model

Code Implementation

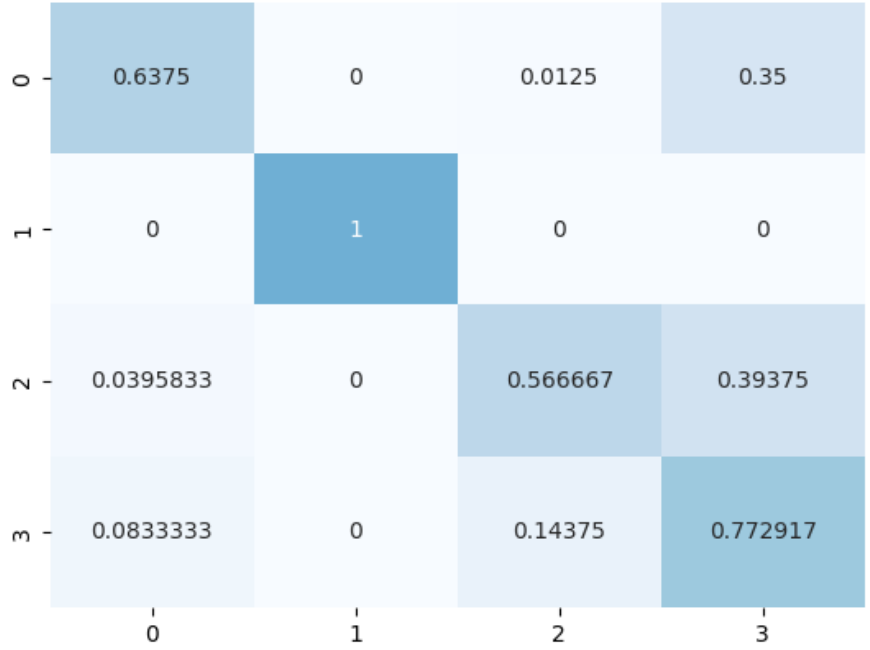
```python
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

# Define the function to create a MobileNet model
def create_mobilenet_model():
    base_model = MobileNet(weights='imagenet', include_top=False, input_shape=(176, 176, 3))

    # Freeze the base model
    base_model.trainable = False

    # Add custom layers on top of MobileNet
    model = models.Sequential([
        base_model,
        layers.GlobalAveragePooling2D(),
        layers.Dense(1024, activation='relu'),
        layers.Dropout(0.3),
        layers.Dense(4, activation='softmax')
    ])

    return model

# Instantiate multiple instances of the MobileNet model
num_models = 4
mobilenet_models = [create_mobilenet_model() for _ in range(num_models)]

checkpoint_cb = ModelCheckpoint("model_mobilenet{epoch:02d}.keras", save_best_only=True)
early_stopping_cb = EarlyStopping(patience=10, restore_best_weights=True)

# Train each MobileNet model and save them
for i, model in enumerate(mobilenet_models):
    print(f"Training and saving MobileNet Model {i+1}/{num_models}")
    # Compile the model
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    # Train the model
    hist = model.fit(X_train, y_train, epochs=30, validation_data=(X_val, y_val), callbacks=[checkpoint_cb, early_stopping_cb])
    # Save the model
    model.save(f"mobilenet_model_{i+1}.h5")
```

Fig.7 Creating and training Mobilenet model

Results

```
CM = confusion_matrix(y_test_,y_pred)
CM_percent = CM.astype('float') / CM.sum(axis=1)[:, np.newaxis]
sns.heatmap(CM_percent,fmt='g',center = True,cbar=False,annot=True,cmap='Blues')
CM
```

```
array([[306,   0,   6, 168],
       [  0, 480,   0,   0],
       [ 19,   0, 272, 189],
       [ 40,   0,  69, 371]], dtype=int64)
```




```
ClassificationReport = classification_report(y_test_,y_pred)
print('Classification Report is : ', ClassificationReport )
```

```
Classification Report is :                precision    recall  f1-score   support

           0       0.84      0.64      0.72       480
           1       1.00      1.00      1.00       480
           2       0.78      0.57      0.66       480
           3       0.51      0.77      0.61       480

    accuracy                           0.74      1920
   macro avg       0.78      0.74      0.75      1920
weighted avg       0.78      0.74      0.75      1920
```

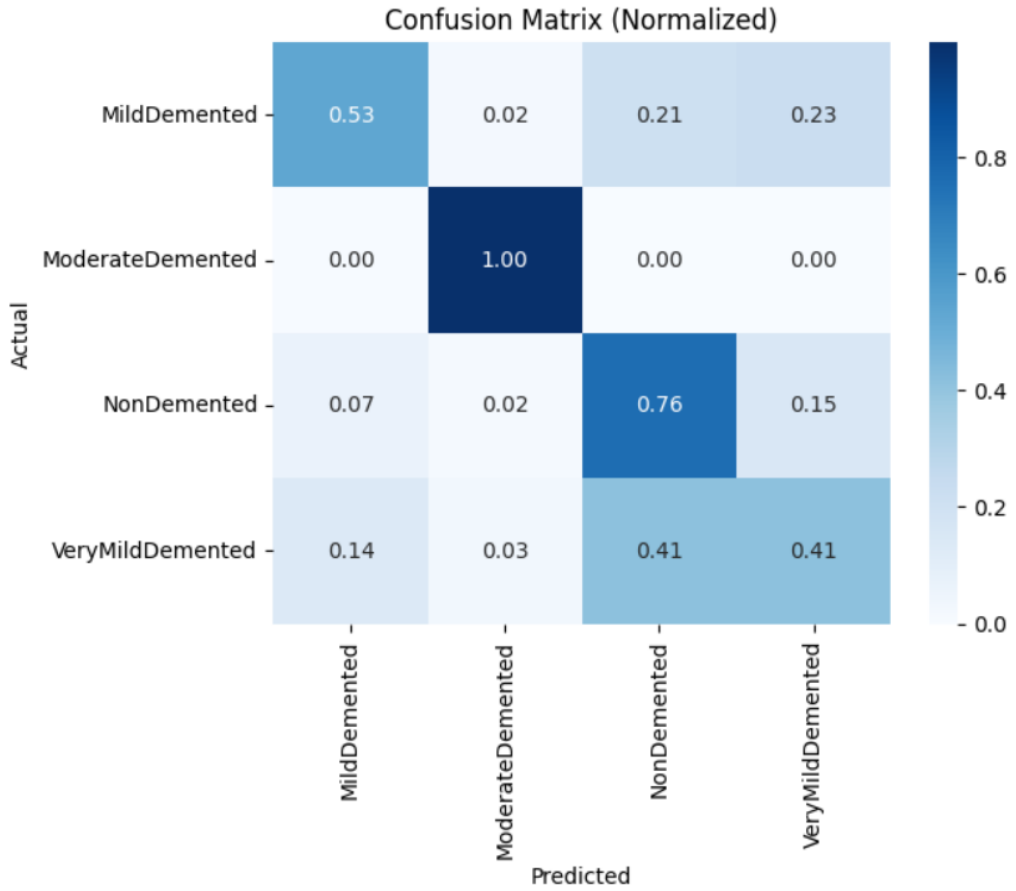
Fig.8 (a) Confusion Matrix of Mobilenet model (b) Classification Report of Mobilenet model

Code Implementation

```python
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.applications import DenseNet121  # Import DenseNet model
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

# Define the function to create a DenseNet model
def create_densenet_model():
    base_model = DenseNet121(weights='imagenet', include_top=False, input_shape=(176, 176, 3))

    # Freeze the base model
    base_model.trainable = False

    # Add custom layers on top of DenseNet
    model = models.Sequential([
        base_model,
        layers.GlobalAveragePooling2D(),
        layers.Dense(1024, activation='relu'),
        layers.Dropout(0.3),
        layers.Dense(4, activation='softmax')
    ])

    return model

# Instantiate multiple instances of the DenseNet model
num_models = 4
densenet_models = [create_densenet_model() for _ in range(num_models)]

checkpoint_cb = ModelCheckpoint("model_densenet{epoch:02d}.keras", save_best_only=True)
early_stopping_cb = EarlyStopping(patience=10, restore_best_weights=True)

# Train each DenseNet model and save them
for i, model in enumerate(densenet_models):
    print(f"Training and saving DenseNet Model {i+1}/{num_models}")
    # Compile the model
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    # Train the model
    hist = model.fit(X_train, y_train, epochs=30, validation_data=(X_val, y_val), callbacks=[checkpoint_cb, early_stopping_cb])
    # Save the model
    model.save(f"densenet_model_{i+1}.h5")
```

Fig.9 Creating and training DenseNet Model

Results




```
ClassificationReport = classification_report(y_test_,y_pred)
print('Classification Report is : ', ClassificationReport )
```

```
Classification Report is :                precision    recall  f1-score   support

           0       0.64      0.92      0.76       480
           1       0.99      1.00      1.00       480
           2       0.73      0.69      0.71       480
           3       0.67      0.42      0.52       480

    accuracy                           0.76      1920
   macro avg       0.76      0.76      0.74      1920
weighted avg       0.76      0.76      0.74      1920
```

Fig.10 (a) Confusion Matrix of Densenet model (b) Classification Report of Densenet model

Code Implementation

```
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix

from tensorflow.keras.models import load_model

cnn_model = load_model("model_1.h5")
mobilenet_model = load_model("mobilenet_model.h5")
densenet_model = load_model("densenet_model.h5")

cnn_preds = cnn_model.predict(X_test)
mobilenet_preds = mobilenet_model.predict(X_test)
densenet_preds = densenet_model.predict(X_test)

ensemble_preds_prob = (cnn_preds + mobilenet_preds + densenet_preds) / 3.0


ensemble_preds = np.argmax(ensemble_preds_prob, axis=1)

true_labels = np.argmax(y_test, axis=1)

print("Confusion Matrix:\n", confusion_matrix(true_labels, ensemble_preds))
print("\nClassification Report:\n", classification_report(true_labels, ensemble_preds))
```

Fig.11 Ensembling of models using Soft Voting

Final Ensembled Results

```
Confusion Matrix:
 [[476   0   2   2]
 [  0 480   0   0]
 [  5   0 451  24]
 [ 15   0  50 415]]

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.98       480
           1       1.00      1.00      1.00       480
           2       0.90      0.94      0.92       480
           3       0.94      0.86      0.90       480

    accuracy                           0.95      1920
   macro avg       0.95      0.95      0.95      1920
weighted avg       0.95      0.95      0.95      1920
```

Fig.12 Classification Report of ensemble model

Webpage

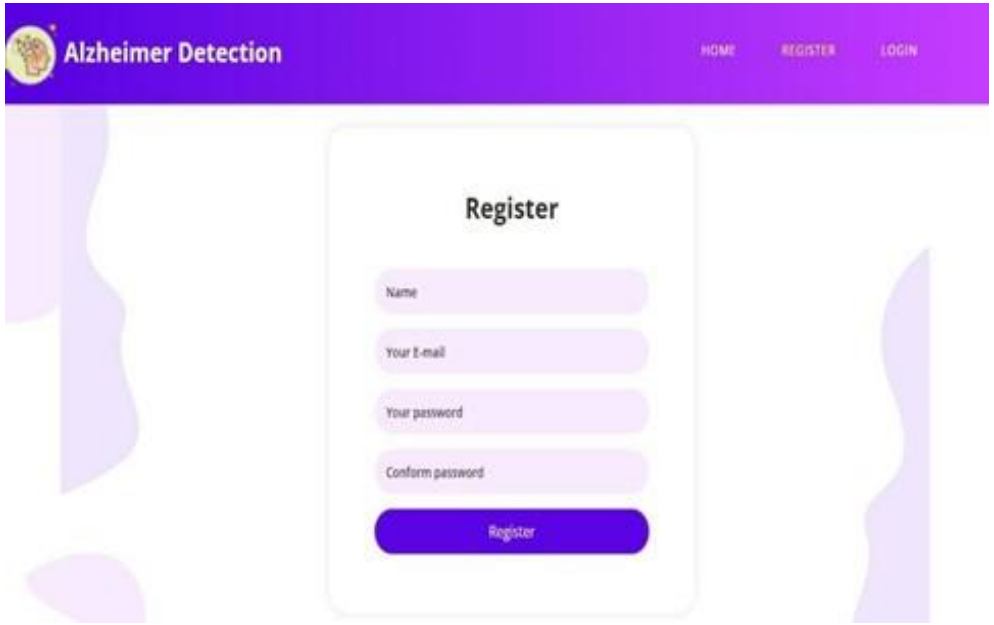


Fig.13 Register page

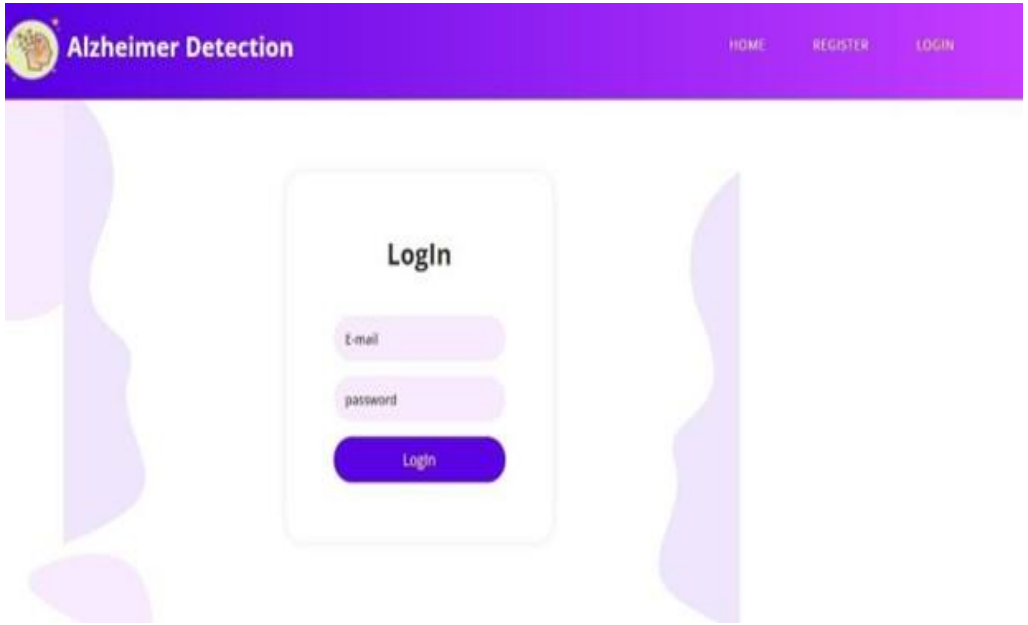


Fig.14 Login page

Webpage

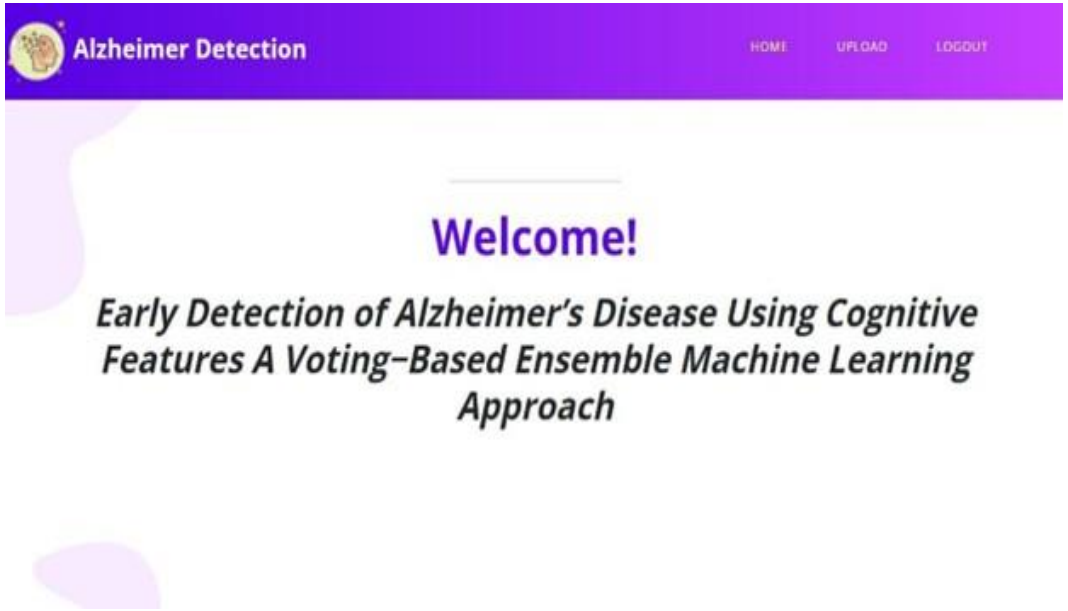


Fig.15 Home page

Fig.16 Upload page

Webpage

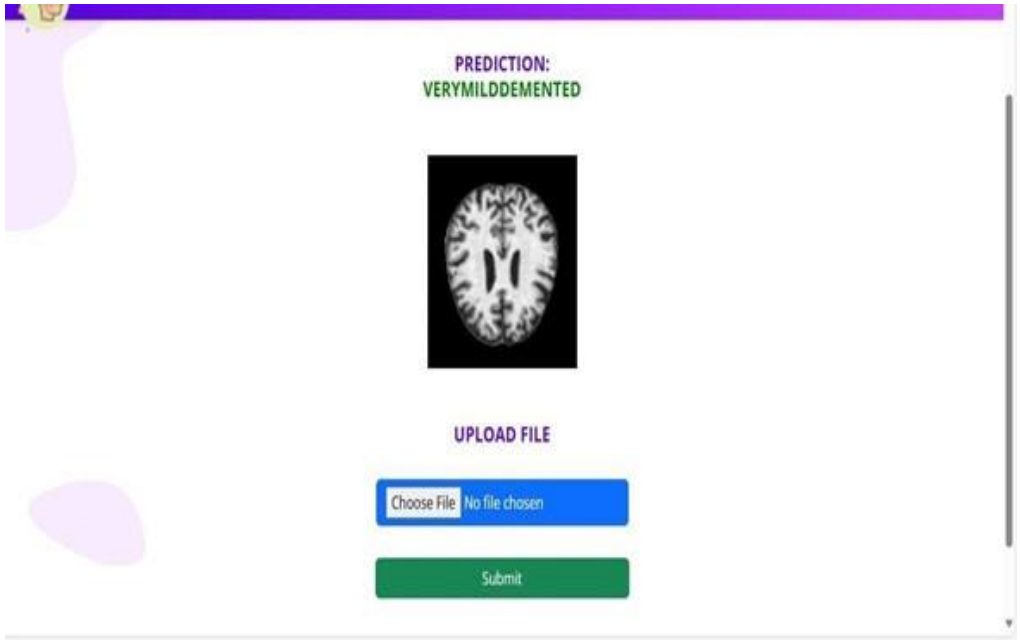


Fig.17 Very Mild Demented

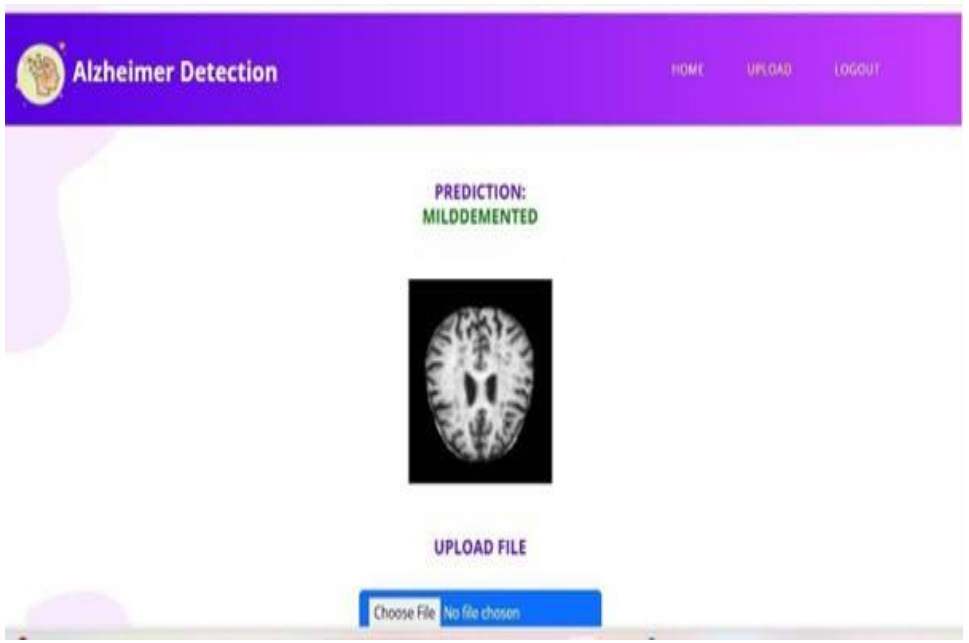


Fig.18 Mild Demented

Future Works

- **Multi-Modal Data Integration:** Incorporate Positron Emission Tomography (PET) or functional MRI (fMRI) data alongside structural MRI to improve classification accuracy.
- **Advanced Deep Learning Techniques:** Implement Transformers and Vision Transformers (ViTs) for improved spatial attention in brain imaging.
- **Real-Time and Mobile Deployment:** Optimize the ensemble model for edge computing using model quantization and pruning, allowing real-time diagnosis on mobile or embedded devices.
- **Longitudinal Analysis for Disease Progression:** Train models on longitudinal MRI data to predict disease progression from MCI to AD over time.
- **Clinical Validation and Deployment:** Conduct clinical trials to compare the AI model's performance against traditional diagnostic methods.

Conclusion

Our literature review highlights the advancements in machine learning and deep learning for early Alzheimer's Disease detection. Existing studies emphasize the importance of MRI and PET imaging, CNN-based architectures, and ensemble models to improve classification accuracy. Based on these insights, our project focuses on preprocessing brain imaging datasets, developing modified deep learning models such as CNN, MobileNet and DenseNET and integrating an ensemble approach for better detection of Healthy Controls (HC), Mild Cognitive Impairment (MCI), and Alzheimer's Disease (AD). By leveraging the strengths of multiple CNN models, we aim to enhance prediction accuracy, robustness, and clinical applicability in diagnosing Alzheimer's at an early stage.

Timeline

ACTIVITIES	January		February				March				April				May	
	Week3	Week4	Week1	Week2	Week3	Week4	Week 1	Week2	Week3	Week4	Week1	Week2	Week3	Week4	Week1	Week2
Task 1 - Literature Survey																
Task 2 - Dataset Analysis																
Task 3 - Code Implementation																
Task 4 - Webpage Creation																
Task 5 - Documentation																

References

[1] Diogo, V.S., Ferreira, H.A., Prata, D. *et al.* Early diagnosis of Alzheimer's disease using machine learning: a multi-diagnostic, generalizable approach. *Alz Res Therapy* **14**, 107 (2022). https://doi.org/10.1186/s13195-022-01047-y

[2] B. S. Rao and M. Aparna, "A Review on Alzheimer's Disease Through Analysis of MRI Images Using Deep Learning Techniques," in IEEE Access, vol. 11, pp. 71542-71556, 2023, doi: 10.1109/ACCESS.2023.3294981.

[3] Lazli L, Boukadoum M, Ait Mohamed O. Computer-Aided Diagnosis System of Alzheimer's Disease Based on Multimodal Fusion: Tissue Quantification Based on the Hybrid Fuzzy-Genetic-Possibilistic Model and Discriminative Classification Based on the SVDD Model. Brain Sci. 2019 Oct 22;9(10):289. doi: 10.3390/brainsci9100289. PMID:

[4] Shukla, Gargi Pant, et al. "Diagnosis and detection of Alzheimer's disease using learning algorithm." *Big Data Mining and Analytics* 6.4 (2023): 504-512.

[5] Wang, Caihua, et al. "A multimodal deep learning approach for the prediction of cognitive decline and its effectiveness in clinical trials for Alzheimer's disease." Translational psychiatry 14.1 (2024): 105.

[6]Al-Shoukry, T. H. Rassem and N. M. Makbol, "Alzheimer's Diseases Detection by Using Deep Learning Algorithms: A Mini-Review," in IEEE Access, vol. 8, pp. 77131-77141, 2020, doi: 10.1109/ACCESS.2020.2989396.keywords: {Dementia;Magnetic resonance imaging;Brain;Single photon emission computed tomography;Biomedical imaging;Alzheimer's disease;deep learning;early stage detection and diagnosis},

AMRITA VISHWA VIDYAPEETHAM

श्रद्धावान् लभते ज्ञानम्

Thank You