SASTRA DEEMED TO BE UNIVERSITY
DEPARTMENT OF CSE, B-TECH MINI PROJECT,2024-2025
FIRST REVIEW

FINE-TUNING ALZHEIMER'S DISEASE DIAGNOSIS USING IMPROVED WAVELET CONVOLUTION NEURAL NETWORK [IWCNN]

TEAM MEMBERS:

Bhuvana B - 226003022
Harini J - 226003170
Chitte Renuka Reddy - 226003034

GUIDED BY

Dr. R Thanuja,
Assistant Professor – II,
Department of CSE, SRC .

CONTENTS:

BASE PAPER:

- **Title:** Advancing early diagnosis of Alzheimer's disease with next-generation deep learning methods
- **Authors:** Cuneyt Ozdemir, Yahya Dogan, **Journal:** Biomedical Signal Processing and Control
- **Volume:** Volume 96
- **Publication Date:** July2024
- **Indexing:** SCI-E Q1
- **Publisher Name:** Elsevier

Biomedical Signal Processing and Control 96 (2024) 106614

Contents lists available at ScienceDirect

Biomedical Signal Processing and Control

journal homepage: www.elsevier.com/locate/bspc

Advancing early diagnosis of Alzheimer's disease with next-generation deep learning methods

Cuneyt Ozdemir *, Yahya Dogan

Computer Engineering, Engineering Faculty, Siirt University, 56100, Siirt, Turkey

ARTICLE INFO

Dataset link: https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images

Keywords:
Alzheimer's disease
Deep learning
SMOTE
Squeeze and excitation block
Avg-TopK

ABSTRACT

Alzheimer's disease, characterized by cognitive decline and memory impairment, poses a significant healthcare challenge. This study presents a specially designed CNN model, utilizing contemporary approaches, to distinguish between various types of Alzheimer's disease. This model can serve as an early diagnostic tool to prevent the disease from progressing towards more pronounced and severe dementia symptoms. In this context, the performance of various transfer learning models has been examined, leading to the development of a specialized model integrating compression and excitation blocks, an innovative Avg-TopK pooling layer, and the SMOTE technique to handle data imbalance. The ablation study results demonstrate the critical role of these components, highlighting the model's effectiveness and innovative design. This study is novel in that it combines modern methodologies for detecting Alzheimer's disease, resulting in a model with state-of-the-art accuracy of 99.84% and improved computing efficiency. Grad-CAM analysis further demonstrates that the model focuses on cortical areas during classification, underscoring its potential as a robust diagnostic tool. These innovations represent a significant advancement over existing models, positioning this study as a pioneering effort in the early diagnosis of Alzheimer's disease. This study aims to contribute significantly to both academic research and medical applications by focusing on integrating artificial intelligence methodologies into medical diagnosis.

1. Introduction

Alzheimer's disease (AD) is a neurodegenerative disease primarily observed in the elderly population, characterized by cognitive decline, memory impairment, and behavioral changes [1,2]. As the disease progresses, cognitive abilities, including thinking skills and the ability to recall information, begin to deteriorate. AD is a leading cause of death worldwide, affecting individuals, families, and healthcare systems [3,4]. AD was initially identified by Dr. Alois Alzheimer in 1907. The disease was diagnosed in a 51-year-old female patient who was undergoing treatment at the psychiatric hospital in Frankfurt am Main. As the disease progressed, the patient exhibited various complex symptoms and experienced significant memory loss. It was observed that she could correctly name a series of objects when shown to her, but shortly after, she would forget everything. Additionally, difficulties in comprehension, hallucinations, and reading problems were detected. The disease process lasted for approximately four and a half years, and the patient passed away at the end of this period [5]. The World Health Organization (WHO) has indicated that approximately 10 million new cases are reported each year, and globally, more than 55 million people live with dementia, with over 60% of these residing in low and middle-income countries. Dementia is currently the seventh leading cause of death and leads to disability and dependency among the elderly population worldwide [6].

AD is one of the most common diseases in elderly individuals, characterized by a gradual slowing and deterioration of cognitive abilities. Early symptoms of the disease include difficulties in recent memory recall, word-finding difficulties, repetition, multitasking impairments, and mood or behavioral changes. As the disease progresses, individuals may experience more pronounced memory loss, lose simple functions such as dressing or bathing, encounter difficulties in language and comprehension, exhibit aggression, engage in wandering behavior, and even experience hallucinations. The severity of symptoms corresponds to the extent of nerve cell damage in the brain [7–9].

The rapid and accurate diagnosis of AD is crucial for effective disease management and treatment. Recently, CNN models have achieved notable success in various fields, including healthcare services [10–16]. Despite their success, current methodologies often face constraints such as high computational requirements and inefficiencies when dealing with imbalanced datasets. These challenges highlight the need for innovative approaches that are both effective and resource-efficient.

* Corresponding author.
E-mail addresses: cozdemir@siirt.edu.tr (C. Ozdemir), yahyadogan@siirt.edu.tr (Y. Dogan).

https://doi.org/10.1016/j.bspc.2024.106614
Received 23 August 2023; Received in revised form 29 May 2024; Accepted 25 June 2024
Available online 6 July 2024

ABSTRACT

Alzheimer's disease (AD) is a neurodegenerative disorder causing memory loss and cognitive decline. Traditional diagnostics are subjective, necessitating reliable automated tools. This work integrates wavelet convolution with Average-TopK pooling and Squeeze-and-Excitation (SE) blocks to enhance feature extraction. Wavelet convolution decomposes MRI images into different frequency sub-bands, capturing spatial and frequency details for improved representation. Synthetic Minority Over-sampling Technique (SMOTE) balances class distribution, while Gradient-Weighted Class Activation Mapping (Grad-CAM) highlights key regions for interpretability. Our approach effectively detects AD and classifies its stages (NC, EMCI, LMCI, AD), aiding early diagnosis and clinical decisions.

PROBLEM STATEMENT:

- Alzheimer's disease remains challenging to detect in its early stages, limiting timely intervention and effective treatment. Existing diagnostic methods, including traditional assessments and AI-based models, often struggle with accuracy and reliability.
- One major challenge is the imbalance in MRI datasets, where fewer cases of severe Alzheimer's leads to biased models that underperform on underrepresented categories. Additionally, conventional CNNs primarily extract spatial features but struggle to capture crucial frequency-based patterns, reducing classification accuracy. Moreover, deep learning models often function as black boxes, offering little transparency in their decision-making process, which makes clinical adoption difficult.

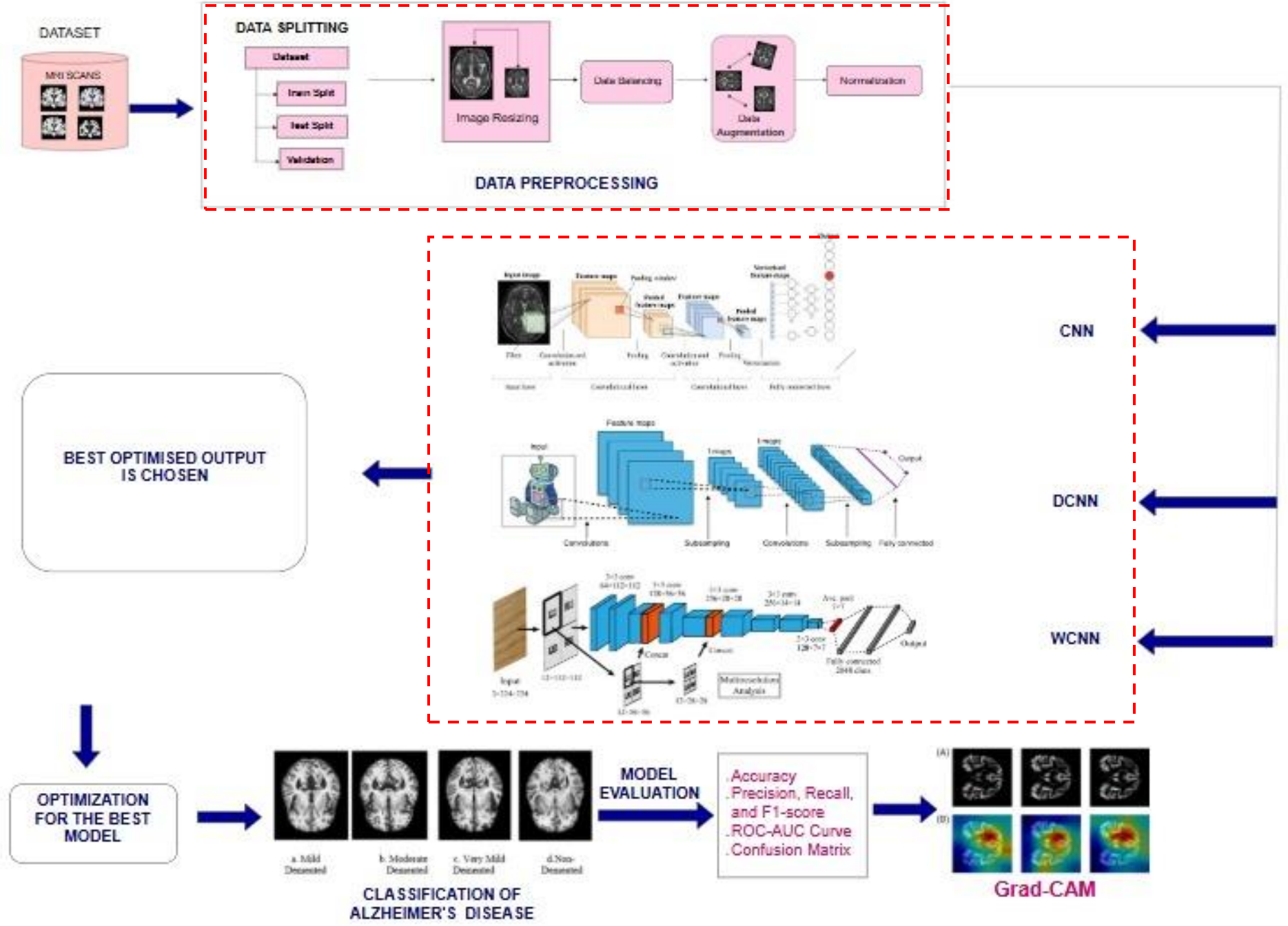
LITERATURE SURVEY

S.NO	TITLE	JOURNAL	YEAR	AUTHOR(S)	TECHNIQUES USED	MERITS	DEMERITS
1.	**Pipelined Deep Learning Architecture for the Detection of Alzheimer's Disease**	Biomedical Signal Processing and Control (Science Direct)	2024	T. Prasath, V. Sumathi	Pipelined LeNet (PLN) architecture, Image Fusion, MRI preprocessing	High classification accuracy (99.5%), faster execution time (0.65 ms)	Limited evaluation on real-world diverse datasets
2.	**Machine and Deep Learning Approaches for Alzheimer's Disease Detection Using Magnetic Resonance Images: An Updated Review**	Measurement (Science Direct)	2024	M. Menagadevi, Somasundaram Devaraj, et al.	CNN, SVM, Transfer Learning, Preprocessing (Histogram Equalization, Contrast Stretching)	Comprehensive review covering 2013-2023 studies, highlights key ML and DL techniques	Lacks experimental validation of reviewed techniques
3.	**Bio-Inspired Deep Learning-Personalized Ensemble Alzheimer's Diagnosis Model for Mental Well-Being**	SLAS Technology (Science Direct)	2024	Ajmeera Kiran, Mahmood Alsaadi, et al.	Personalized dynamically ensemble CNN (PDECNN), Attention Mechanism	Improves classification accuracy by 4%-11%, adapts to variations in brain degeneration	High computational complexity

s.no	TITLE	JOURNAL	YEAR	AUTHOR(S)	TECHNIQUES USED	MERITS	DEMERITS
4.	**Fine-Grained and Multiple Classification for Alzheimer's Disease With Wavelet Convolution Unit Network**	IEEE Transactions on Biomedical Engineering	2023	Jinyu Wen, Yang Li, Meie Fang, et al.	Wavelet Convolution Unit (WCU), Multi-scale Wavelet Decomposition, Diffusion Tensor Imaging (DTI)	Achieves state-of-the-art fine-grained classification accuracy (97.89%)	Requires extensive computational resources
5.	**Detection of Alzheimer's Disease Using Deep Learning Models: A Systematic Literature Review**	Informatics in Medicine Unlocked	2024	Eqtidar M. Mohammed, Ahmed M. Fakhrudeen, et al.	Various CNN architectures (ResNet, AlexNet, GoogleNet, EfficientNetB7), RNN, Deep Belief Networks	Summarizes 45 research papers on AD detection using deep learning	Lacks focus on non-imaging biomarkers

s.no	TITLE	JOURNAL	YEAR	AUTHOR(S)	TECHNIQUES USED	MERITS	DEMERITS
6.	**Deep Learning-based Alzheimer's disease classification using MRI and PET images**	Neuro computing	2024	R.S.Karthik, M.Sharma, etal.	CNN, Transfer learning, Multi-modal fusion(MRI + PET)	Achieves high classification Accuracy(98.2%). Effective multi-modal feature extraction.	Requires large-scale labelled datasets for training.
7.	**Alzheimer's Disease Diagnosis using 3D Convolution Neural Networks and MRI scans**	Medical Image Analysis	2024	L Chen, H.Wang, et al.	3D-CNN, Data Argumentation, Pretrained ResNet-50	Effectively processes 3D brain scans, achieves robust feature extraction	Requires high computational power for 3D processing
8.	**Hybrid Deep Learning Model for Early Detection of Alzheimer's Disease**	IEEE Access	2023	N. Verma, A. Gupta, et al.	Hybrid CNN-RNN architecture, Attention Mechanism	Improves early detection accuracy, extracts spatial and temporal features.	High training time and memory usage

ARCHITECTURE DIAGRAM


DATASET
MRI SCANS
DATA SPLITTING
Dataset
Train Split
Test Split
Validation
Image Resizing
Data Balancing
Data Augmentation
Normalization
DATA PREPROCESSING
CNN
DCNN
WCNN
BEST OPTIMISED OUTPUT IS CHOSEN
OPTIMIZATION FOR THE BEST MODEL
a. Mild Demented
b. Moderate Demented
c. Very Mild Demented
d.Non-Demented
CLASSIFICATION OF ALZHEIMER'S DISEASE
MODEL EVALUATION
.Accuracy
.Precision, Recall, and F1-score
.ROC-AUC Curve
.Confusion Matrix
Grad-CAM

WORK PLAN :

Module 1 : (23/01/2025 -16/02/2025)

Dataset Understanding & Preprocessing

- **Base Paper Analysis:** Studied algorithms and methodologies.
- **Dataset Collection:** Sourced MRI dataset from Kaggle.

Module 2 : (19/02/2025 - 14/03/2025)

Preprocessing, Feature Extraction & Model Training

- **Preprocessing:** Grayscale conversion, resizing (128x128), normalization.
- **Class Balancing:** Applied SMOTE/oversampling to handle data imbalance.
- **Feature Extraction:** Squeeze-Excitation Networks, Avg-TopK Pooling.
- **Model Training:**
  1. **CNN (Baseline Model)** – Standard convolutional model.
  2. **DCNN (Deep CNN)** – Enhanced architecture with deeper layers.

WORK PLAN :

Module 3 : (21/03/2025 -10/04/2025)

Model Evaluation & Prediction

- **Enhancements:** Train Improved Wavelet Convolution Neural Network.
- **Metrics:** Accuracy, Precision, Recall, F1-score, confusion matrix.
- **Comparison:** Evaluate CNN vs. DCNN vs. Wavelet-CNN.
- **Explainability:** Use Grad-CAM to visualize model focus areas.
- **Deployment:** Final model for Alzheimer's stage classification.

FEATURE EXTRACTION ALGORITHMS

AVERAGE-TOPK

- The **Average-topk** function computes the mean of the top k highest values in a given set, reducing sensitivity to outliers and improving stability in optimization tasks.
- Formula:

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of n numerical values.

1. **Sort** the values in descending order:

$$x_{(1)} \geq x_{(2)} \geq \cdots \geq x_{(n)}$$

where $x_{(i)}$ represents the i-th largest value in the set.

2. **Select the top-k values**:

$$S_k = \{x_{(1)}, x_{(2)}, \ldots, x_{(k)}\}$$

3. **Compute the AVERAGEtopk**:

$$\text{AVERAGEtopk}(X, k) = \frac{1}{k} \sum_{i=1}^{k} x_{(i)}$$

where $x_{(i)}$ are the top k largest values from X.

SQUEEZE-AND-EXCITATION BLOCK

- The **SE block** is like a smart filter for images in deep learning. It helps a neural network focus on the most important parts of an image while ignoring less useful details.

- It works in three simple steps:

1. **Squeeze** – Looks at the whole image and figures out what features are most important.
2. **Excitation** – Decides how much attention each feature should get.
3. **Recalibration** – Adjusts the image data to highlight the important parts and reduce the noise.

1. Squeeze (Global Average Pooling - GAP)

Each channel's global feature is computed as:

$$s_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} X_c(i,j)$$

where:

- $X_c(i,j)$ is the feature map value at position (i,j) for channel c.
- H, W are the height and width of the feature map.
- s_c is the **global descriptor** for each channel.

2. Excitation (Fully Connected Layers & Activations)

The channel-wise attention scores are obtained using two fully connected layers:

$$z = \sigma(W_2 \delta(W_1 s))$$

where:

- W_1 and W_2 are learnable weight matrices.
- δ is the **ReLU activation** function.
- σ is the **Sigmoid activation** to scale values between 0 and 1.
- z is the attention weight for each channel.

3. Recalibration (Channel-wise Scaling)

The recalibrated feature map is obtained by scaling each channel with its corresponding attention weight:

$$X'_c = z_c \cdot X_c$$

where z_c is the learned attention weight for channel c, and X'_c is the enhanced feature map.

This process **amplifies important features** and **suppresses less useful ones**, improving CNN performance with minimal extra computation! 🚀

CNN PSEUDOCODE:

Input:

d: dataset, l: dataset true labels, t: target size

Output:

Classified stage of Alzheimer's Disease on MRI scan

```
Let i be the MRI scan to be classified
for i in dataset do ImageDataGenerator
        ftrain, ftest, ltrain, ltest ← split feature set and labels
         into train subset and test subset
         S ← build_model(ftrain, ltrain)
         Let Conv2D be the convolutional layer and AvgTopKPooling be the pooling layer
         for j in layers do
            tf.keras.layers.Conv2D(filters, (3,3)padding="same", activation='relu')
            tf.keras.layers.BatchNormalization()
            tf.keras.layers.ReLU()
            se_block()

Let DIR be the path joining version for other paths

        for k in categories do
            DIR = main_path + 'category'
            dataset_length.append(category_length)
            Result ← evaluate(DIR, ltest, S)
Return result;
```

DCNN PSEUDOCODE:

Input:

d: dataset, l: dataset true labels, t: target size

Output:

Classified stage of Alzheimer's Disease on MRI scan

```
Let i be the MRI scan to be classified
For i in dataset, do ImageDataGenerator
    ftrain, ftest, ltrain, ltest ← split feature set and labels into train subset and test subset
    S ← DCNN(ftrain, ltrain)
            Let Conv2D be the convolutional layer and AvgTopKPooling be the pooling layer
            for j in layers do
                tf.keras.layers.Conv2D(filters, (3,3), padding="same", activation='relu')
                AvgTopKPooling(k=3)
            Let DIR be the path joining version for other paths
            for k in categories do
                DIR = main_path + 'category'
                dataset_length.append(category_length)
                Result ← evaluate(DIR, ltest, S)
Return result;
```

DATASET DESCRIPTION :

The research uses an Alzheimer's disease brain MRI dataset from Kaggle, which contains 6,400 images classified into four different classes:

- Non-Demented
- Very Mild Demented
- Mild Demented
- Moderate Demented

Dataset Link:
https://www.kaggle.com/datasets/borhanitrash/alzheimer-mri-disease-classification-dataset

IMPLEMENTATION (PARTIAL)

```python
import pandas as pd

# Correct file paths
train_path = "/kaggle/input/alzheimer-mri-disease-classification-dataset/Alzheimer MRI Disease Classification Dataset/Data/train-00000-of-00001-c08a401c53fe5312.parquet"
test_path = "/kaggle/input/alzheimer-mri-disease-classification-dataset/Alzheimer MRI Disease Classification Dataset/Data/test-00000-of-00001-44110b9df98c5585.parquet"

# Load the datasets
train_df = pd.read_parquet(train_path)
test_df = pd.read_parquet(test_path)

# Display first few rows
print("Train Data:")
display(train_df.head())

print("\nTest Data:")
display(test_df.head())
```

```
Train Data:
```

	image	label
0	{'bytes': b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x...	2
1	{'bytes': b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x...	0
2	{'bytes': b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x...	3
3	{'bytes': b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x...	3
4	{'bytes': b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x...	2

```
Test Data:
```

	image	label
0	{'bytes': b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x...	3
1	{'bytes': b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x...	0
2	{'bytes': b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x...	2
3	{'bytes': b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x...	3
4	{'bytes': b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x...	0

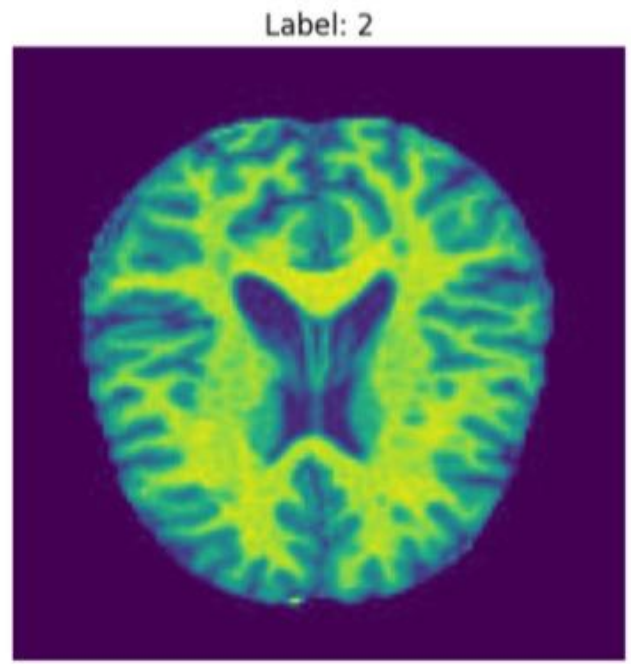
```python
# Check unique class labels
print("Unique Labels in Train Dataset:")
print(train_df["label"].unique())

# Count instances per class
print("\nClass Distribution:")
print(train_df["label"].value_counts())
```

```
Unique Labels in Train Dataset:
[2 0 3 1]

Class Distribution:
label
2    2566
3    1781
0     724
1      49
Name: count, dtype: int64
```

```python
#convert all images to NumPy arrays for model training
import tensorflow as tf

# Function to process images
def preprocess_image(image_data):
    img = decode_image(image_data)  # Decode
    img = img.resize((128, 128))  # Resize
    img = np.array(img) / 255.0  # Normalize
    return img

# Apply transformation
X_train = np.array([preprocess_image(i['bytes']) for i in train_df["image"]])
y_train = np.array(train_df["label"])

X_test = np.array([preprocess_image(i['bytes']) for i in test_df["image"]])
y_test = np.array(test_df["label"])

print(f"Processed Train Data: {X_train.shape}, Labels: {y_train.shape}")
print(f"Processed Test Data: {X_test.shape}, Labels: {y_test.shape}")
```

```
Processed Train Data: (5120, 128, 128), Labels: (5120,)
Processed Test Data: (1280, 128, 128), Labels: (1280,)
```

```python
#Decode Image Data
import io
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

# Function to decode image
def decode_image(image_data):
    return Image.open(io.BytesIO(image_data))

# Extract the first image
first_image_data = train_df["image"].iloc[0]['bytes']
first_image = decode_image(first_image_data)

# Display image
plt.imshow(first_image)
plt.axis("off")
plt.title(f"Label: {train_df['label'].iloc[0]}")
plt.show()
```

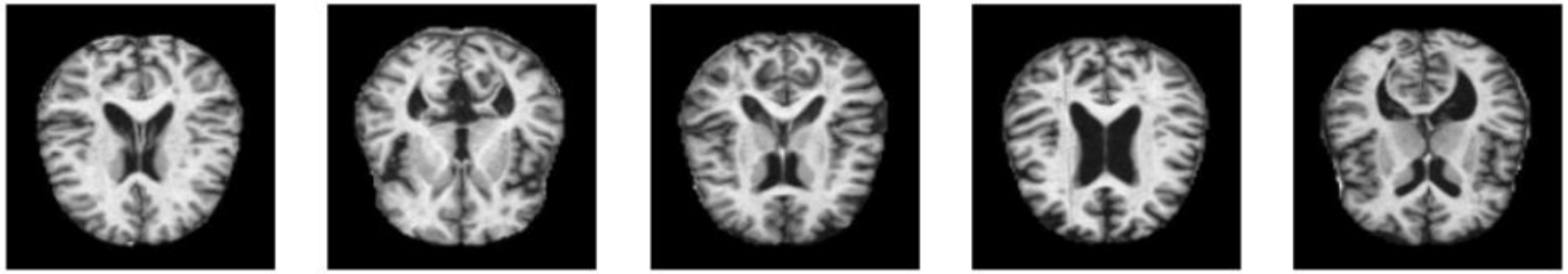


PREPROCESSED IMAGES:

```
print(X_train.shape)  #gray-scale image
```

```
(5120, 128, 128)
```

```
import matplotlib.pyplot as plt

# Show some grayscale images
plt.figure(figsize=(10, 5))
for i in range(5):  # Show first 5 images
    plt.subplot(1, 5, i + 1)
    plt.imshow(X_train[i], cmap="gray")  # Use cmap="gray" for grayscale images
    plt.axis("off")

plt.show()
```



```python
# 1️⃣ Split training data into training & validation sets
# 2️⃣ Apply SMOTE to balance classes in the training set
from sklearn.model_selection import train_test_split

# Split into train and validation (80-20 split)
X_train_res, X_val, y_train_res, y_val = train_test_split(
    X_train, y_train, test_size=0.2, stratify=y_train, random_state=42
)

print(f"Train Set: {X_train_res.shape}, Labels: {y_train_res.shape}")
print(f"Validation Set: {X_val.shape}, Labels: {y_val.shape}")
```

```
Train Set: (4096, 128, 128), Labels: (4096,)
Validation Set: (1024, 128, 128), Labels: (1024,)
```

```python
from imblearn.over_sampling import SMOTE

# Reshape for SMOTE (flatten images)
X_train_flat = X_train_res.reshape(X_train_res.shape[0], -1)

# Apply SMOTE
smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train_flat, y_train_res)

# Reshape back to image format
X_train_balanced = X_train_balanced.reshape(-1, 128, 128)

print(f"Balanced Train Set: {X_train_balanced.shape}, Labels: {y_train_balanced.shape}")
```

```
Balanced Train Set: (8212, 128, 128), Labels: (8212,)
```

```python
#Build CNN Model with SE Blocks & Avg-TopK Pooling
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, Dense, Dropout, GlobalAveragePooling2D, Multiply, Reshape

# Squeeze & Excitation Block
def se_block(input_tensor, ratio=16):
    """Squeeze & Excitation Block"""
    filters = input_tensor.shape[-1]

    # Squeeze: Global Average Pooling
    se = GlobalAveragePooling2D()(input_tensor)
    se = Dense(filters // ratio, activation="relu")(se)
    se = Dense(filters, activation="sigmoid")(se)

    # Excite: Scale feature maps
    se = Reshape((1, 1, filters))(se)
    return Multiply()([input_tensor, se])
```

```python
import tensorflow.keras.backend as K
from tensorflow.keras.layers import Layer

class AvgTopKPooling(Layer):
    """Avg-TopK Pooling Layer"""
    def __init__(self, k=3, **kwargs):
        super(AvgTopKPooling, self).__init__(**kwargs)
        self.k = k

    def call(self, inputs):
        top_k, _ = tf.math.top_k(inputs, k=self.k, sorted=False)
        return K.mean(top_k, axis=-1)

    def compute_output_shape(self, input_shape):
        return input_shape[:-1]
```

MODEL 1: CNN

```python
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, BatchNormalization, ReLU, Flatten, Dense, Dropout

# Define CNN Model
def build_model(input_shape=(128, 128, 1), num_classes=4):
    inputs = Input(shape=input_shape)

    # Block 1
    x = Conv2D(32, (3, 3), padding="same")(inputs)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = se_block(x)

    # Block 2
    x = Conv2D(64, (3, 3), padding="same")(x)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = se_block(x)

    # Block 3
    x = Conv2D(128, (3, 3), padding="same")(x)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = se_block(x)

    # Block 4
    x = Conv2D(256, (3, 3), padding="same")(x)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = se_block(x)

    # Avg-TopK Pooling
    x = AvgTopKPooling(k=3)(x)

    # Flatten & Fully Connected Layers
    x = Flatten()(x)
    x = Dense(128, activation="relu")(x)
    x = Dropout(0.3)(x)
    x = Dense(64, activation="relu")(x)
    x = Dropout(0.3)(x)

    # Output Layer
    outputs = Dense(num_classes, activation="softmax")(x)

    model = Model(inputs, outputs)
    return model
```

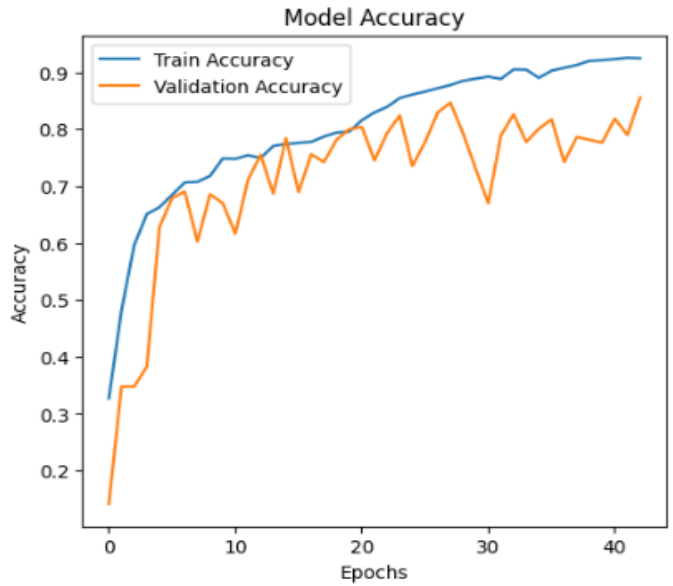
MODEL 2: DCNN

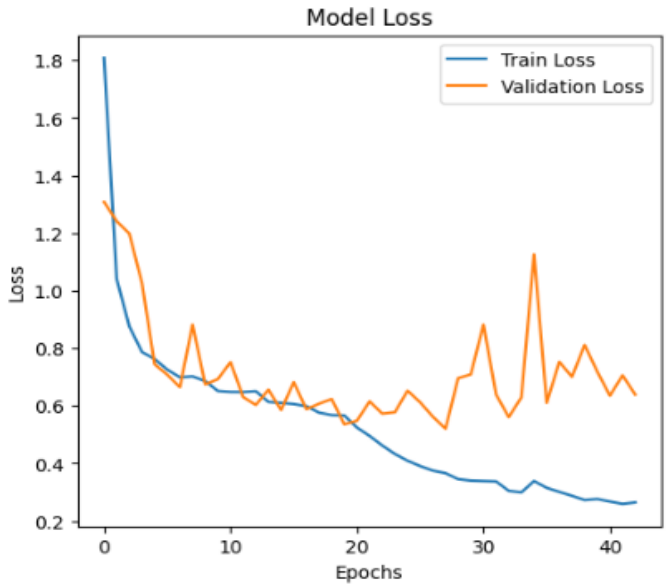
```python
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, BatchNormalization, ReLU, Flatten, Dense, Dropout

# Define Deep CNN Model
def build_model(input_shape=(128, 128, 1), num_classes=4):
    inputs = Input(shape=input_shape)

    # Block 1
    x = Conv2D(32, (3, 3), padding="same")(inputs)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = se_block(x)

    # Block 2
    x = Conv2D(64, (3, 3), padding="same", strides=2)(x)  # Downsampling with stride
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = se_block(x)

    # Block 3
    x = Conv2D(128, (3, 3), padding="same")(x)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = se_block(x)

    # Block 4
    x = Conv2D(256, (3, 3), padding="same", strides=2)(x)  # Another downsampling
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = se_block(x)

    # Block 5
    x = Conv2D(512, (3, 3), padding="same")(x)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = se_block(x)

    # Avg-TopK Pooling
    x = AvgTopKPooling(k=3)(x)

    # Flatten & Fully Connected Layers
    x = Flatten()(x)
    x = Dense(256, activation="relu")(x)
    x = Dropout(0.4)(x)
    x = Dense(128, activation="relu")(x)
    x = Dropout(0.3)(x)
    x = Dense(64, activation="relu")(x)
    x = Dropout(0.3)(x)

    # Output Layer
    outputs = Dense(num_classes, activation="softmax")(x)

    model = Model(inputs, outputs)
    return model
```

MODEL 1: CNN

```python
# Evaluate on test data
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=1)

print(f"\nTest Accuracy: {test_acc*100:.2f}%")
print(f"Test Loss: {test_loss:.4f}")
```

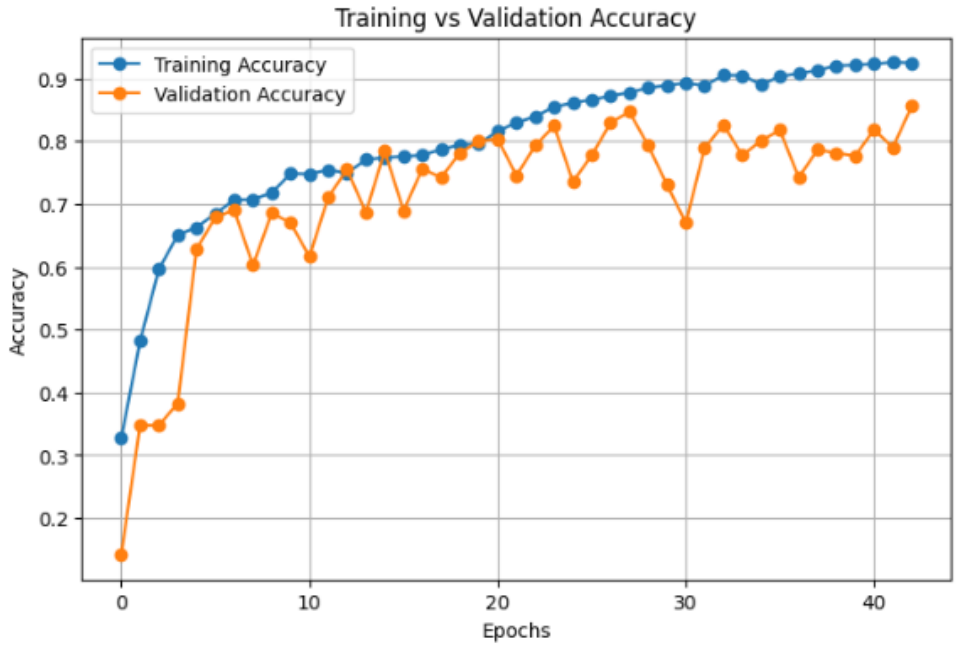
```
40/40 ━━━━━━━━━━━━━━━━━━━━ 12s 135ms/step - accuracy: 0.8413 - loss: 0.5900

Test Accuracy: 84.53%
Test Loss: 0.5134
```

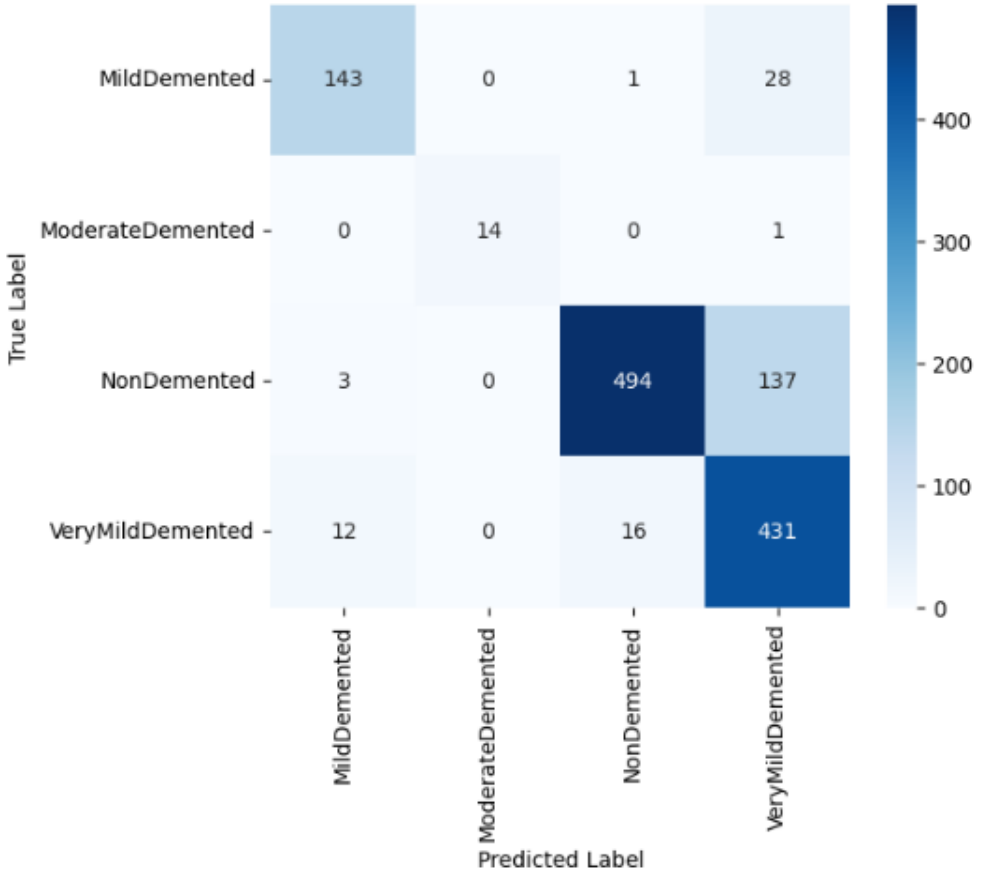




MODEL 1: CNN

```
Final Training Accuracy: 92.46%
Final Validation Accuracy: 85.55%
```





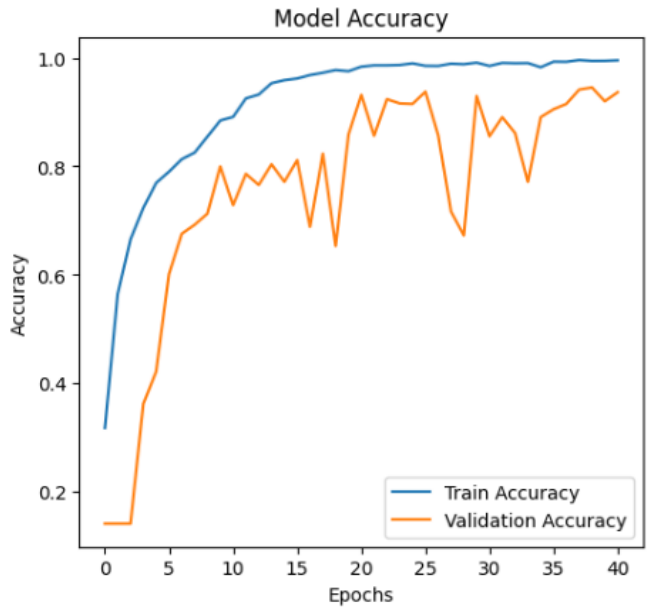
Classification Report:

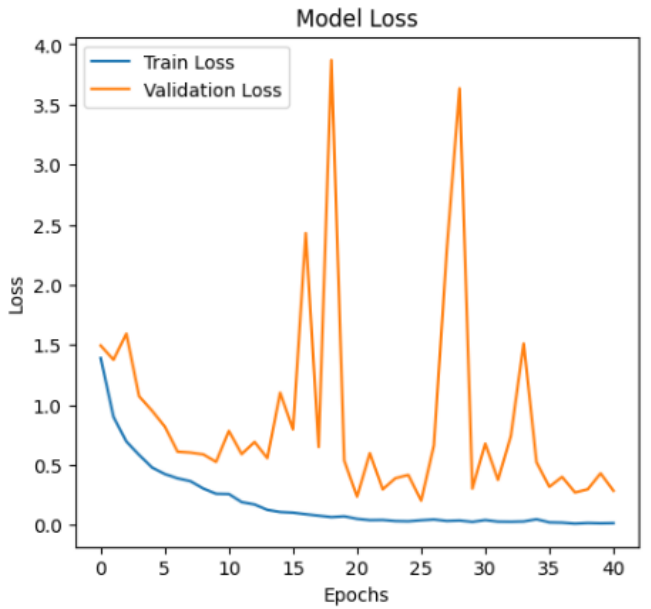
	precision	recall	f1-score	support
MildDemented	0.91	0.83	0.87	172
ModerateDemented	1.00	0.93	0.97	15
NonDemented	0.97	0.78	0.86	634
VeryMildDemented	0.72	0.94	0.82	459
accuracy			0.85	1280
macro avg	0.90	0.87	0.88	1280
weighted avg	0.87	0.85	0.85	1280

MODEL 2: DCNN

```python
# Evaluate on test data
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=1)

print(f"\nTest Accuracy: {test_acc*100:.2f}%")
print(f"Test Loss: {test_loss:.4f}")
```

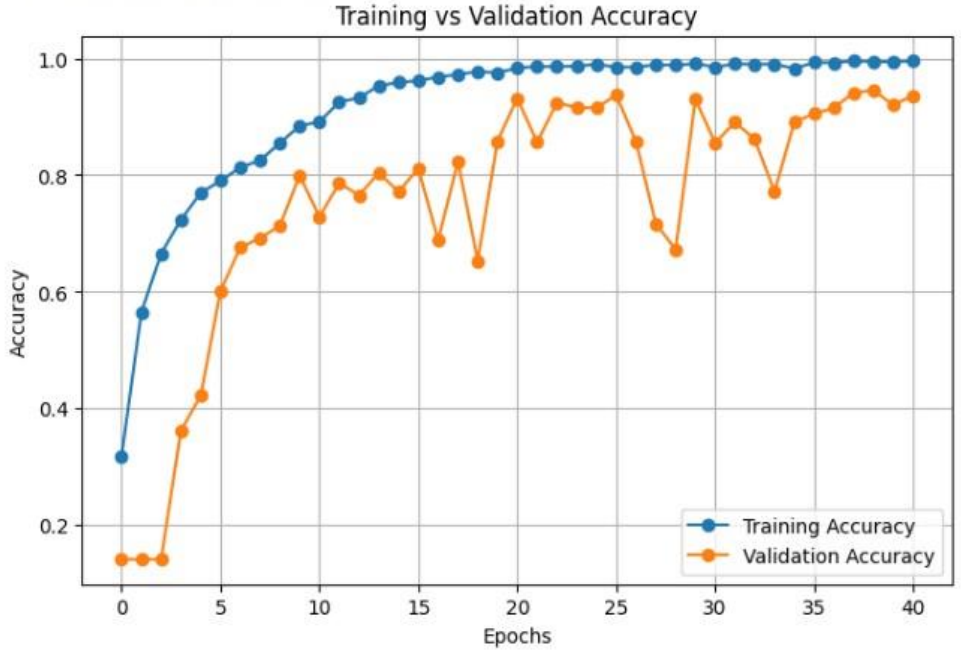
```
40/40 ━━━━━━━━━━━━━━━━━━━━ 6s 35ms/step - accuracy: 0.9311 - loss: 0.3274

Test Accuracy: 93.44%
Test Loss: 0.2707
```

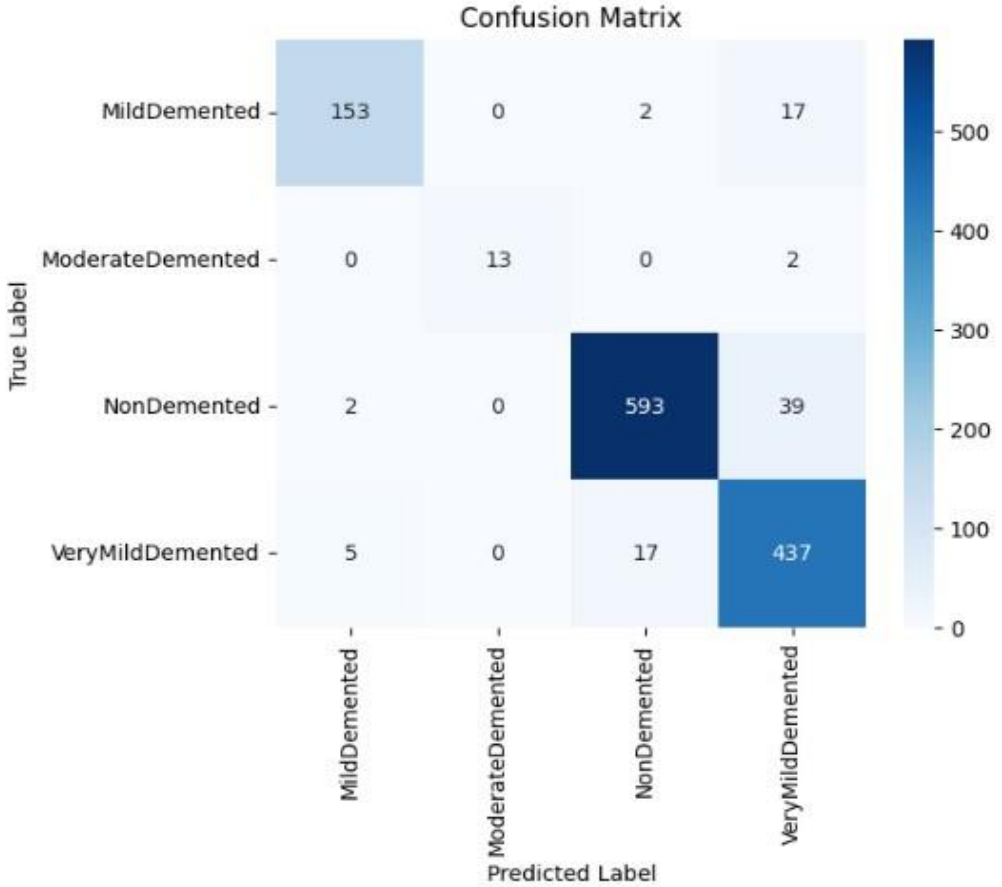




MODEL 2: DCNN

```
Final Training Accuracy: 99.54%
Final Validation Accuracy: 93.65%
```





Classification Report:

	precision	recall	f1-score	support
MildDemented	0.96	0.89	0.92	172
ModerateDemented	1.00	0.87	0.93	15
NonDemented	0.97	0.94	0.95	634
VeryMildDemented	0.88	0.95	0.92	459
accuracy			0.93	1280
macro avg	0.95	0.91	0.93	1280
weighted avg	0.94	0.93	0.93	1280

ACCURACY DESCRIPTION:

Metric	Previous CNN	New DCNN	Improvement 🚀
Training Accuracy	92.46%	99.54%	+7.08% (Better learning)
Validation Accuracy	85.55%	93.65%	+8.1% (Stronger generalization)
Test Accuracy	84.53%	93.44%	+8.91% (Better real-world performance)
Test Loss	0.5134	0.2707	Lower loss → More confident predictions

LIST OF REFERENCES:

- Jinyu Wen, Yang Li, Meie Fang, Lei Zhu and Ping Li, David Dagan Feng, Life Fellow, Member, IEEE.(2023).Fine-Grained and Multiple Classification for Alzheimer's Disease With Wavelet Convolution Unit Network

- Shamrat, F.J.M., Akter, S., Azam, S., Karim, A., Ghosh, P., Tasnim, Z., Hasib, K.M., De Boer, F., & Ahmed, K. (2023). AlzheimerNet: An effective deep learning-based proposition for Alzheimer's disease stages classification from functional brain changes in magnetic resonance images. IEEE Access, 11, 16376–16395.

- World Health Organization. (2020). The top 10 causes of death. Retrieved from https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death.

- Illakiya, T., & Ramamurthy, K. (2023). AHANet: Adaptive Hybrid Attention Network for Alzheimer's Disease Classification Using Brain Magnetic Resonance Imaging. Bioengineering, 10(6), 714.

- Finder, S. E., Amoyal, R., Treister, E., & Freifeld, O. (2024). Wavelet Convolutions for Large Receptive Fields. In Proceedings of the European Conference on Computer Vision (ECCV). Available at https://arxiv.org/abs/2407.05848

THANK YOU!