Alzheimer's MRI Classification Using Deep Learning Models

Team Number:067

Name: Sathvic Puchalapalli
UBIT: sathvicp
UB No: 50592562
Email: sathvicp@buffalo.edu

Name: Siddharth Kodam
UBIT: skodam
UB No:50605005
Email: skodam@buffalo.edu

Name: Sai Sravan Reddy Vangeti
UBIT: svangeti
UB No: 50594384
Email: svangeti@buffalo.edu

Category 1: Computer Vision Applications - **Topic 1:** Image recognition and classification

Project: Alzheimer's MRI Classification

Contribution Table:

Names	Research	Coding	Experimentation
Sathvic Puchalapalli	**33**	**34**	**33**
Siddharth Kodam	**34**	**33**	**33**
Sai Sravan Reddy Vangeti	**33**	**33**	**34**

I. Introduction

With the rise of deep learning in recent years, its impact on medical imaging has been nothing short of transformative. One particular area that has greatly benefited is the detection of neurodegenerative diseases, such as Alzheimer's it is a disease related brain disorder that slowly destroys memory, thinking skills, and eventually the ability to carry out even simple tasks. Alzheimer's it is a progressive condition, and its early diagnosis plays a critical role in managing the disease and improving patient outcomes. However, conventional methods for analyzing MRI scans can be time consuming, labor intensive, and would be susceptible to causing a human error which possess a significant challenge for timely and reliable diagnosis.

This project addresses these limitations by applying advanced deep learning models to classify MRI scans according to the stages of Alzheimer's progression. We focus on four distinct stages: Non-Demented, Very Mild Demented, Mild Demented, and Moderate Demented. The study evaluates and compares the performance of popular convolutional neural networks such as VGGNet, GoogLeNet, ResNet, and DenseNet in terms of accuracy, precision, and recall. speed, interpretability, and computational efficiency. The ultimate goal is to pinpoint the most balanced and practical model for clinical use.

II. Project Objectives:

In this project, we had aimed to explore how different deep learning models can help in detecting and classifying Alzheimer's Disease through MRI scans. Since early and accurate diagnosis is crucial for treatment, our goal is to find an approach that not only performs well in terms of accuracy but is also efficient and practical for real-world use. We had evaluated and compared several well-known convolutional neural networks will include VGGNet, GoogLeNet, ResNet, and DenseNet on a common dataset using the same preprocessing techniques. This will allow us to fairly judge how each model performs across multiple criteria. Our key focus areas had included:

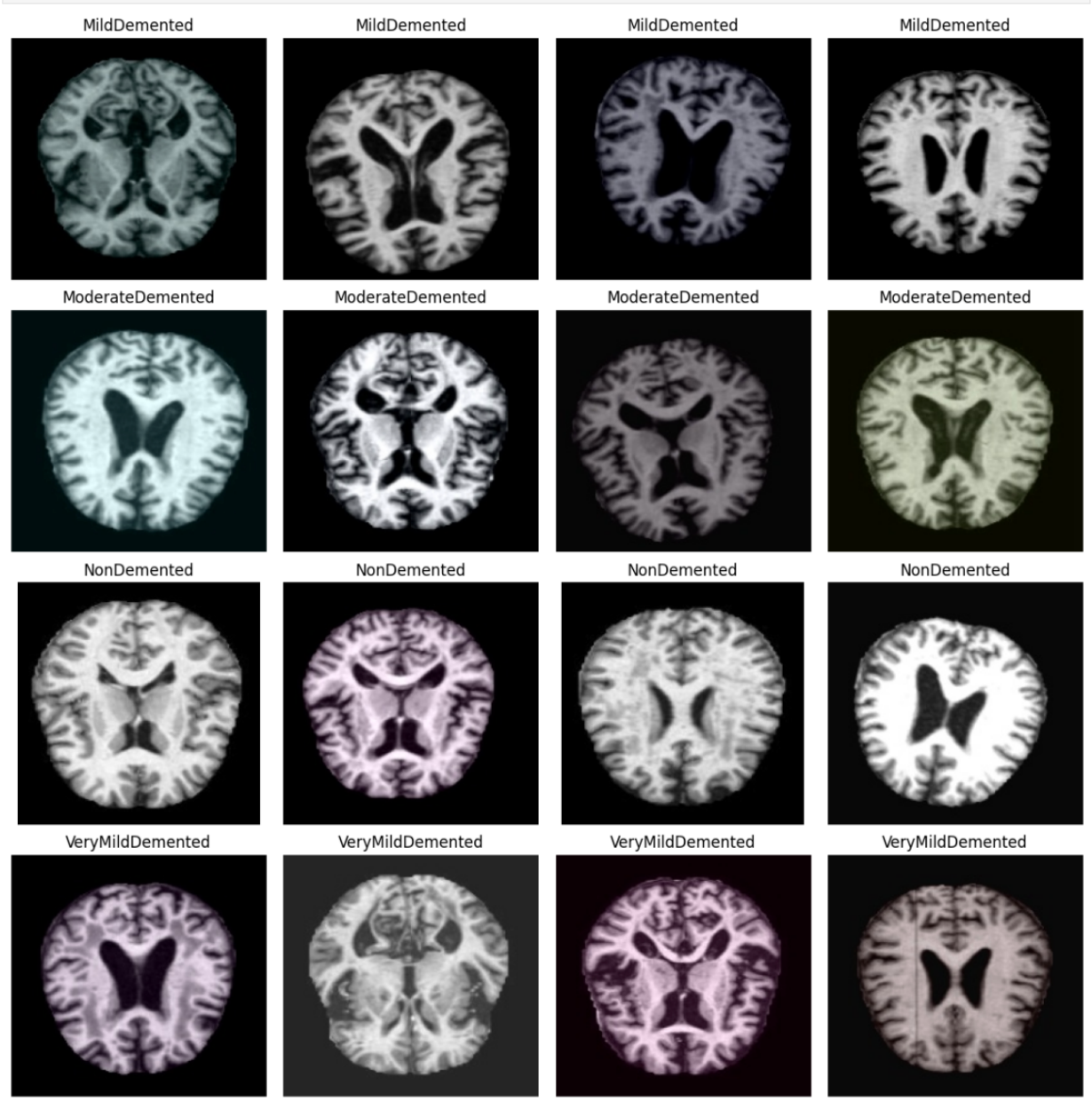
- **Classification Performance:** We have measured how well each model identifies the different stages of Alzheimer's (Non-Demented, Very Mild Demented, Mild Demented, Moderate Demented) using metrics like accuracy, precision, recall, F1-score, and AUC-ROC. This helps us understand the strengths and weaknesses of each model from a prediction standpoint.
- **Computational Efficiency:** Since real-world applications often have resource constraints, we'll be looking at training time, inference speed, and model size to see which architectures are more efficient and scalable.
- **Visual Testing:** As part of the evaluation, we had also randomly selected MRI scans and displayed the model's predictions alongside the true labels. This hands-on, visual validation gives us a better feel for how reliable the model really is, especially from a non-technical, user-facing perspective.

By integrating deep learning techniques and comprehensive evaluation metrics, we have successfully identified the model that achieves an optimal balance between classification performance, computational efficiency, and clinical interpretability. Though our current implementation has yielded promising results, we recognize this as just the beginning of a broader research agenda. Our primary goal was creating a reliable diagnostic support system for medical professionals, and it has been accomplished within our existing constraints. While current computational limitations have defined the scope of our work, future access to enhanced resources would enable more sophisticated architectures, larger datasets, and potentially even better and faster classifications. For now, we present our methodical observations and results as a meaningful contribution to the field of neurological disease detection.

III. Dataset Overview:

Alzheimer's MRI Scans Across Cognitive Stages The dataset contains brain MRI scans that represent different stages of Alzheimer's, starting from healthy brains and progressing to moderate dementia. As the disease advances, the brain structure gradually changes, and these images help capture that progression. Many of the differences, especially in the early stages, are incredibly subtle, which makes them hard to spot with the naked eye. That's exactly where deep learning becomes useful, as it can detect fine patterns that we might otherwise miss and use them to predict the stage of the disease.

Alzheimer's Disease Progression: Brain MRI Images Across Different Stages:



The dataset consists of MRI brain scans labeled across four stages of Alzheimer's: Non-Demented, Very Mild Demented, Mild Demented, and Moderate Demented. While the images often appear visually similar, especially in early stages, subtle changes in brain structure can indicate cognitive decline. These differences are difficult to detect manually, making this a suitable task for deep learning. Our goal is to train models that can capture these nuanced patterns. This could support earlier and more accurate diagnosis in clinical settings.

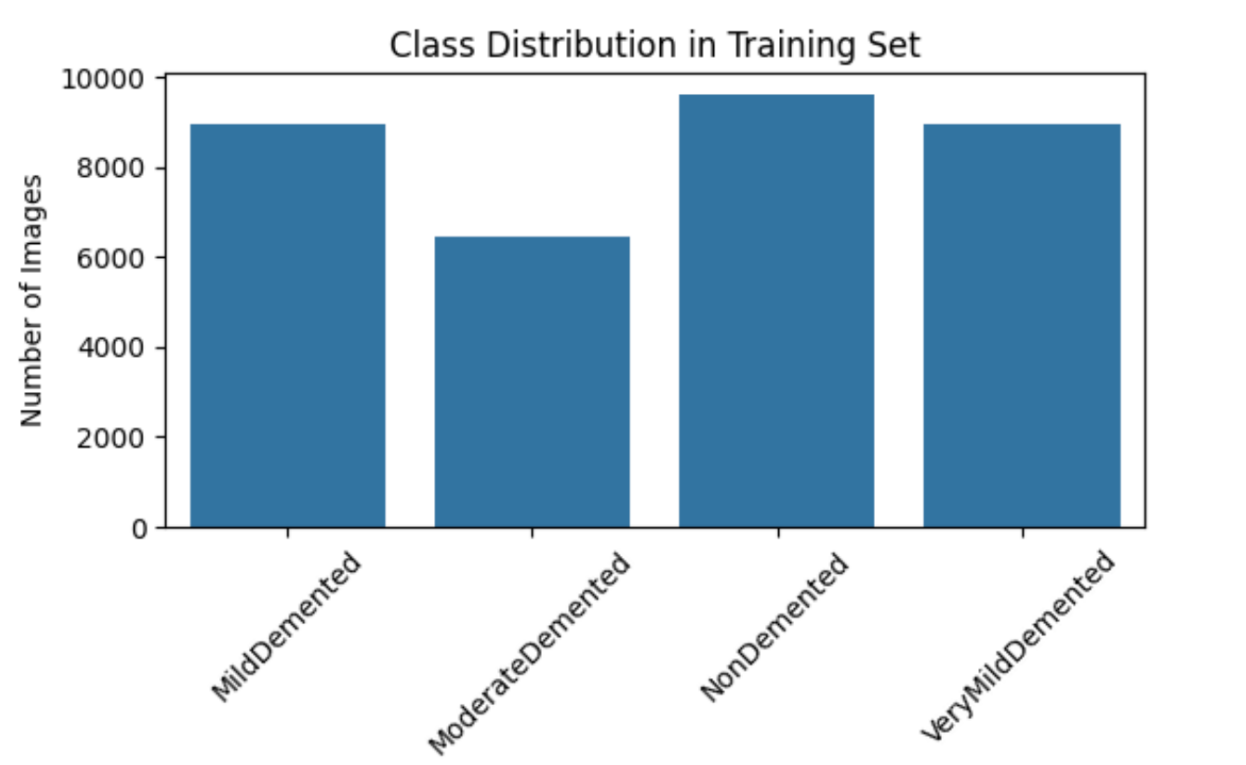
IV. **Exploratory Data Analysis (EDA)**

1. Class-to-Index Mapping

```
Classes: ['MildDemented', 'ModerateDemented', 'NonDemented', 'VeryMildDemented']
Class-to-Index Mapping: {'MildDemented': 0, 'ModerateDemented': 1, 'NonDemented': 2, 'VeryMildDemented': 3}
```

The dataset was organized into four categories: Mild Demented, Moderate Demented, Non-Demented, and Very Mild Demented. Each class was mapped to a corresponding index to support label encoding during model training.

2. Class Distribution



We visualized the number of images per class to check for imbalance. As shown in the bar chart, the distribution is not perfectly even, with Non-Demented and Mild Demented classes containing more samples than the others. To address this, we applied weighted sampling during training to ensure fair learning across all categories.

3. Weighted sampling

In deep learning, **Weighted sampling** adjusts the probability of selecting samples during training, assigning higher weights to underrepresented classes. This helps balance the dataset by ensuring the model learns effectively from both common and rare classes. For medical image classification tasks like ours where we are doing Alzheimer's detection, this technique would prevent bias toward dominant classes, improving generalization and performance on rare conditions. It is especially crucial for case as it will accurately diagnosis of early-stage diseases.

4. Image Dimensions and Shape Summary

	Width	Height
count	33984.000000	33984.000000
mean	196.233522	188.116761
std	7.819527	3.909764
min	180.000000	180.000000
25%	200.000000	190.000000
50%	200.000000	190.000000
75%	200.000000	190.000000
max	200.000000	190.000000

We have evaluated the dimensions of all training images to ensure they were consistent and suitable for model input. The image dimensions were primarily around 200×200 pixels, with an average size of ~196×188 pixels. This indicates that the images were already closely standardized, minimizing the need for extensive resizing. Standardizing this image size is crucial for ensuring that the model could learn efficiently without the introduction of unnecessary distortions in the input data, while ultimately leading to better performance in tasks like disease classification.

5. Image Normalization: Mean and Standard Deviation

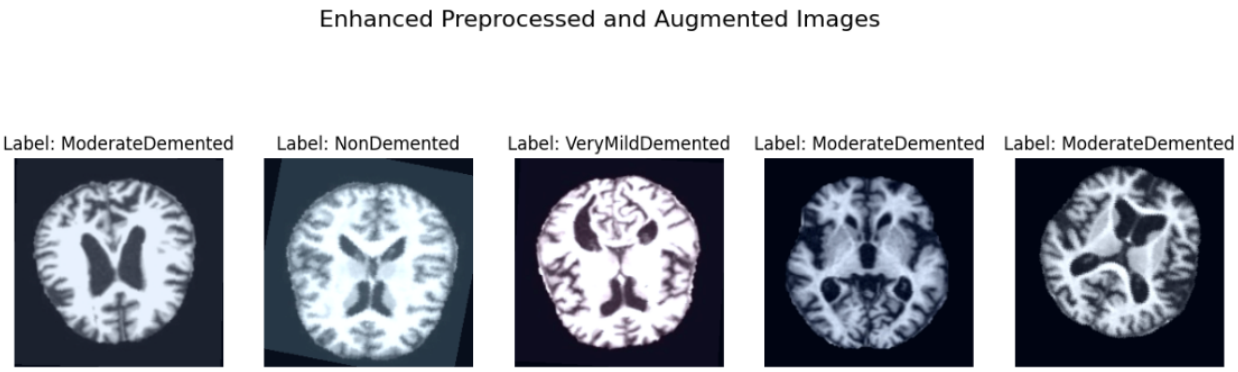
```
100%|██████████████████████████████████| 1062/1062 [01:02<00:00, 16.89it/s]
Computed Mean: tensor([-0.8388, -0.7281, -0.5023])
Computed Std: tensor([1.3073, 1.3364, 1.3304])
```

To ensure stable model training, we computed the mean and standard deviation for each image channel. The calculated means were [-0.8374, -0.7267, -0.5099], and the standard deviations were [1.3088, 1.3379, 1.3319]. These values were used to normalize the input image tensors, ensuring zero mean and unit variance. This normalization helps stabilize training, improves convergence speed, and prevents issues like vanishing or exploding gradients, crucial for deep learning tasks.

6. Image Output After Normalization

The images above show MRI scans of different Alzheimer's stages: ModerateDemented, MildDemented, VeryMildDemented, and NonDemented. After normalization, the pixel values have been adjusted to have zero mean and unit variance, ensuring consistent input data for the model. This step minimizes lighting and contrast variations, helping the model focus on relevant features like brain changes associated with dementia stages. Normalizing the data improves training stability and model accuracy.

7. Enhanced Preprocessed and Augmented MRI Samples



To ensure our model learns effectively from the MRI data, we applied preprocessing techniques such as grayscale conversion, normalization, and basic augmentations. The image above shows a batch of samples after these transformations. These steps not only standardize input quality but also help the model become more resilient to variations, improving its ability to generalize. Each image retains its original label, confirming that augmentation was applied without affecting class integrity.

V. Methodology

a. Deep Learning Architecture: ResNet-18 Architecture Implementation

Our team implemented a ResNet-18 architecture for Alzheimer's disease classification using MRI scans. The model addresses the vanishing gradient problem commonly encountered in deep networks by implementing residual connections, allowing the network to learn residual mappings instead of full transformations. This approach ensures improved gradient flow, enabling effective training on complex MRI data.

Core Architecture Components

- **Residual Block Implementation**

The Residual Block is the core element of our architecture. It features a pair of convolutional layers that each include batch normalization and ReLU activation functions. What makes these blocks special is their skip connections, which create a direct pathway for information to flow from input to output through addition. When the dimensions between input and output don't match, the block uses 1x1 convolutions with a stride of 1 to handle the difference. This clever design shifts the network's focus from learning complete transformations to learning simpler residual mappings, which dramatically improves how gradients flow backward through the network during training.

- **Network Structure**

The ResNet-18 architecture consisted of:

1. Initial layer: 7 X 7 convolution with stride 2, followed by max pooling
2. Four sequential groups of Residual Blocks:
    - Layer 1: Two blocks with 64 output channels with no downsampling
    - Layer 2: Two blocks with 128 output channels with stride 2 downsampling
    - Layer 3: Two blocks with 256 output channels with stride 2 downsampling
    - Layer 4: Two blocks with 512 output channels with stride 2 downsampling
3. Final processing: Adaptive average pooling, flattening, dropout, and fully connected classification layer

- **Core Design Choices**

The primary implementation in our code included:

- Residual connections enabling better gradient flow throughout the network
- Strategic downsampling that reduces spatial dimensions while increasing feature depth
- Batch normalization after each convolution to reduce internal covariate shift
- Dropout regularization before the final classification layer to prevent overfitting

- **Forward Pass Processing**

During the forward pass, MRI images undergo systematic transformation through the network layers. The initial convolution extracts primary features, while subsequent residual blocks progressively refine these features at increasing levels of abstraction. The spatial dimensions are systematically reduced while channel depth increases, allowing the network to capture hierarchical representations critical for accurate Alzheimer's disease classification.

- **Classification Approach**

The final classification process involves adaptive average pooling to compress spatial dimensions to 1×1, followed by flattening and dropout regularization. The fully connected layer then produces classification scores corresponding to different stages of Alzheimer's disease, enabling accurate diagnostic predictions from MRI scans.

b. Deep Learning Architecture: DenseNet Implementation

Our team implemented the Densely Connected Convolutional Networks architecture for Alzheimer's disease classification using MRI scans. This architecture enhances information flow and gradient propagation throughout the network by introducing dense connections between layers. The dense connectivity pattern allows for improved feature reuse, resulting in more compact models with fewer parameters compared to traditional convolutional networks.

Core Architecture Components

Dense Block Implementation

The Dense Block serves as the fundamental building block of our architecture. Each DenseBlock features multiple convolutional layers where each layer receives input from all previous layers, feature concatenation that preserves and reuses learned features, and a growth rate parameter controlling the number of new feature maps generated by each layer. This design enables extensive feature reuse, allowing the network to learn more diverse representations while maintaining computational efficiency.

Transition Layer Implementation

Between Dense Blocks, we implemented Transition Layers to manage dimensionality and computational complexity. These layers contain a 1×1 convolution functioning as a bottleneck to reduce the number of feature maps, average pooling to reduce spatial resolution by half, and controlled channel reduction to maintain network efficiency. These transition layers prevent computational explosion while preserving essential features.

Network Structure

The complete DenseNet architecture consists of:

1. Initial layer: 7 X 7 convolution with stride 2, batch normalization, ReLU, and max pooling
2. Four sequential Dense Blocks with intervening Transition Layers:
    - Dense Block 1: Input of 64 channels → Transition Layer → 128 channels
    - Dense Block 2: Input of 128 channels → Transition Layer → 256 channels
    - Dense Block 3: Input of 256 channels → Transition Layer → 512 channels
    - Dense Block 4: Input of 512 channels
3. Final processing: Adaptive average pooling, flattening, and fully connected classification layer

Core Design Choices

The primary design implementation included:

- Dense connectivity pattern enabling extensive feature reuse
- Strategic dimensionality reduction through transition layers
- Batch normalization after each convolution for stable training
- Controlled growth rate to manage model complexity

Forward Pass Processing

During forward propagation, input MRI images undergo systematic transformation through the network layers. The initial convolution extracts primary features, while subsequent Dense Blocks progressively enhance these features through dense connections. Transition Layers manage computational complexity by reducing spatial dimensions and feature map quantities at strategic points.

Classification Approach

The final classification process involves adaptive average pooling to compress spatial dimensions to 1x1, followed by flattening. The fully connected layer then produces classification scores corresponding to different stages of Alzheimer's disease, enabling accurate diagnostic predictions from MRI scans.

c. Deep Learning Architecture: VGG 16 Implementation

Our team implemented the VGG-16 architecture for Alzheimer's disease classification using MRI scans. This 16-layer deep convolutional neural network was built with integrated Batch Normalization to stabilize the learning process and improve classification performance. The architecture follows a systematic approach of stacked convolutional layers followed by pooling operations to extract hierarchical features from medical images.

Core Architecture Components

Convolutional Layer Groups

The VGG-16 architecture consists of five groups of convolutional layers:

1. **Conv1**: Two 3x3 convolutional layers with 64 filters, followed by Batch Normalization, ReLU activation, and MaxPooling
2. **Conv2**: Two 3x3 convolutional layers with 128 filters, with similar normalization, activation, and pooling
3. **Conv3**: Three 3x3 convolutional layers with 256 filters, followed by normalization, activation, and pooling
4. **Conv4**: Three 3x3 convolutional layers with 512 filters, with identical post-processing
5. **Conv5**: Three 3x3 convolutional layers with 512 filters, completing the feature extraction process

Each layer group progressively extracts more complex features while reducing spatial dimensions through Max Pooling operations.

Classification Layers

After feature extraction, the network employs three fully connected layers:

1. First fully connected layer: 1024 units with Batch Normalization, ReLU, and Dropout (0.5)
2. Second fully connected layer: 512 units with identical normalization and regularization
3. Output layer: Final classification layer with outputs corresponding to disease classes

Core Design Choices

The primary design implementation included:

- Consistent use of small 3x3 convolutional filters throughout the network
- Strategic application of Batch Normalization after each layer for training stability
- Dropout regularization in fully connected layers to prevent overfitting
- Systematic Max Pooling for dimension reduction while preserving critical features

Forward Pass Processing

During forward propagation, input MRI images undergo systematic transformation through the network. The convolutional layers extract progressively more complex features, starting with basic edges and textures and advancing to high-level disease-specific patterns. The fully connected layers then interpret these features to produce classification decisions.

Classification Approach

The final fully connected layer produces classification scores corresponding to different stages of Alzheimer's disease. This architecture effectively learns hierarchical representations from medical images, enabling accurate diagnostic predictions from MRI scans. The implementation with Batch Normalization ensures stable and efficient training, making it particularly effective for medical image classification tasks requiring deep feature extraction.

d. Deep Learning Architecture: GoogLeNet Implementation

Our team implemented the GoogleNet architecture for Alzheimer's disease classification using MRI scans. This innovative network is designed to efficiently capture multiscale features from input images using parallel convolutional operations of different kernel sizes. The architecture's core innovation the Inception block enables simultaneous feature extraction at multiple scales, making it particularly effective for complex medical image classification tasks.

Core Architecture Components

Inception Block Implementation

The InceptionBlock serves as the fundamental building block, featuring parallel processing paths. The 1x1 convolution branch provides direct dimensionality reduction with Batch Normalization and ReLU activation. The 1x1 → 3x3 convolution branch captures medium-scale features with reduced parameters. The 1x1 → 5x5 convolution branch extracts larger-scale features efficiently. The 3x3 pooling → 1x1 convolution branch preserves spatial information while reducing channels. Outputs from all branches are concatenated along the channel dimension, creating rich multi-scale feature representations that enable the network to simultaneously capture information at various receptive field sizes while maintaining computational efficiency.

Network Structure

The complete GoogleNet architecture consists of:

1. Initial layers:
    - 7x7 convolution with stride 2, Batch Normalization, ReLU, and MaxPooling
    - 1x1 followed by 3x3 convolution with normalization, activation, and pooling
2. Inception modules sequence:

    - Inception3a and Inception3b blocks followed by MaxPool3
    - Inception4a through Inception4e blocks followed by MaxPool4
    - Inception5a and Inception5b blocks for high-level feature extraction
3. Final processing:
    - Adaptive average pooling to reduce spatial dimensions to 1x1
    - Dropout (rate 0.4) for regularization
    - Fully connected classification layer

Core Design Choices

The primary design implementation included:

- Multi-path Inception blocks enabling efficient multi-scale feature extraction
- Strategic use of 1x1 convolutions for dimensionality reduction
- Batch normalization for training stability
- Dropout regularization to prevent overfitting

Forward Pass Processing

During forward propagation, input MRI images first undergo initial convolutions to extract basic features. The network then processes these features through multiple Inception blocks, progressively capturing increasingly complex patterns at different scales. This multi-scale approach is particularly valuable for medical imaging, where features of interest may appear at various resolutions. The final adaptive average pooling operation reduces spatial dimensions to a uniform size, preparing features for classification.

Classification Approach

The final classification process involves flattening the pooled features, applying dropout regularization, and processing through a fully connected layer. This generates classification scores corresponding to different stages of Alzheimer's disease, enabling accurate diagnostic predictions from MRI scans. The multi-scale feature extraction capability of the Inception architecture proves especially valuable for identifying subtle patterns indicative of different disease stages.

Results

A. ResNet Model Evaluations

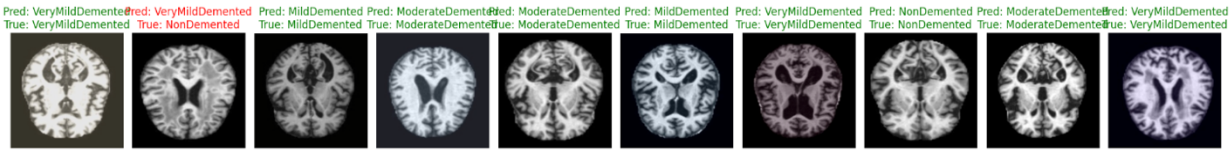
1. Training and Validation Progress

```
Using device: mps
Epoch 1/15
Training epoch 1: 100%|██████████| 1062/1062 [09:28<00:00,  1.87it/s]
Train Loss: 1.0274 Train Accuracy: 0.5275
Validation epoch 1: 100%|██████████| 1062/1062 [03:42<00:00,  4.77it/s]
Validation Loss: 0.8242 Validation Accuracy: 0.6148

Epoch 2/15
Training epoch 2: 100%|██████████| 1062/1062 [12:07<00:00,  1.46it/s]
Train Loss: 0.7475 Train Accuracy: 0.6470
Validation epoch 2: 100%|██████████| 1062/1062 [04:45<00:00,  3.72it/s]
Validation Loss: 1.1357 Validation Accuracy: 0.5107

Epoch 3/15
Training epoch 3: 100%|██████████| 1062/1062 [09:42<00:00,  1.82it/s]
Train Loss: 0.6936 Train Accuracy: 0.6755
Validation epoch 3: 100%|██████████| 1062/1062 [03:44<00:00,  4.74it/s]
Validation Loss: 0.7556 Validation Accuracy: 0.6353

Epoch 4/15
Training epoch 4: 100%|██████████| 1062/1062 [11:03<00:00,  1.60it/s]
Train Loss: 0.6482 Train Accuracy: 0.6991
Validation epoch 4: 100%|██████████| 1062/1062 [04:02<00:00,  4.38it/s]
Validation Loss: 0.6864 Validation Accuracy: 0.6933

Epoch 5/15
Training epoch 5: 100%|██████████| 1062/1062 [11:28<00:00,  1.54it/s]
Train Loss: 0.6068 Train Accuracy: 0.7233
Validation epoch 5: 100%|██████████| 1062/1062 [03:54<00:00,  4.54it/s]
Validation Loss: 0.6398 Validation Accuracy: 0.7120

Epoch 6/15
Training epoch 6: 100%|██████████| 1062/1062 [10:52<00:00,  1.63it/s]
Train Loss: 0.5699 Train Accuracy: 0.7376
Validation epoch 6: 100%|██████████| 1062/1062 [03:52<00:00,  4.56it/s]
Validation Loss: 0.5805 Validation Accuracy: 0.7345

Epoch 7/15
Training epoch 7: 100%|██████████| 1062/1062 [10:59<00:00,  1.61it/s]
Train Loss: 0.5260 Train Accuracy: 0.7622
Validation epoch 7: 100%|██████████| 1062/1062 [03:51<00:00,  4.58it/s]
Validation Loss: 0.6725 Validation Accuracy: 0.7119

Epoch 8/15
Training epoch 8: 100%|██████████| 1062/1062 [09:24<00:00,  1.88it/s]
Train Loss: 0.4249 Train Accuracy: 0.8113
Validation epoch 8: 100%|██████████| 1062/1062 [03:33<00:00,  4.97it/s]
Validation Loss: 0.3755 Validation Accuracy: 0.8334

Epoch 9/15
Training epoch 9: 100%|██████████| 1062/1062 [09:10<00:00,  1.93it/s]
Train Loss: 0.3854 Train Accuracy: 0.8285
Validation epoch 9: 100%|██████████| 1062/1062 [03:31<00:00,  5.03it/s]
Validation Loss: 0.3461 Validation Accuracy: 0.8481

Epoch 10/15
Training epoch 10: 100%|██████████| 1062/1062 [09:11<00:00,  1.93it/s]
Train Loss: 0.3587 Train Accuracy: 0.8442
Validation epoch 10: 100%|██████████| 1062/1062 [03:31<00:00,  5.02it/s]
Validation Loss: 0.3182 Validation Accuracy: 0.8630

Epoch 11/15
Training epoch 11: 100%|██████████| 1062/1062 [09:09<00:00,  1.93it/s]
Train Loss: 0.3454 Train Accuracy: 0.8495
Validation epoch 11: 100%|██████████| 1062/1062 [03:31<00:00,  5.02it/s]
Validation Loss: 0.2993 Validation Accuracy: 0.8692

Epoch 12/15
Training epoch 12: 100%|██████████| 1062/1062 [09:09<00:00,  1.93it/s]
Train Loss: 0.3232 Train Accuracy: 0.8601
Validation epoch 12: 100%|██████████| 1062/1062 [03:34<00:00,  4.95it/s]
Validation Loss: 0.3000 Validation Accuracy: 0.8723

Epoch 13/15
Training epoch 13: 100%|██████████| 1062/1062 [09:10<00:00,  1.93it/s]
Train Loss: 0.3104 Train Accuracy: 0.8665
Validation epoch 13: 100%|██████████| 1062/1062 [03:32<00:00,  5.00it/s]
Validation Loss: 0.2646 Validation Accuracy: 0.8887

Epoch 14/15
Training epoch 14: 100%|██████████| 1062/1062 [09:17<00:00,  1.90it/s]
Train Loss: 0.2939 Train Accuracy: 0.8757
Validation epoch 14: 100%|██████████| 1062/1062 [03:35<00:00,  4.93it/s]
Validation Loss: 0.2684 Validation Accuracy: 0.8874

Epoch 15/15
Training epoch 15: 100%|██████████| 1062/1062 [09:12<00:00,  1.92it/s]
Train Loss: 0.2776 Train Accuracy: 0.8822
Validation epoch 15: 100%|██████████| 1062/1062 [03:32<00:00,  4.99it/s]
Validation Loss: 0.2345 Validation Accuracy: 0.9016

Training complete in 205m 48s
Best Val Acc: 0.9016
```

The ResNet model showed faster learning and high test accuracy (90.16%) due to its residual connections, which allowed the gradients to flow better and enabled the model to converge quickly. The model performed exceptionally well in distinguishing the ModerateDemented cases and handled MildDemented well, though it struggled with the more subtle distinctions between VeryMildDemented and NonDemented. This rapid convergence aligns with the model's ability to

effectively learn deeper features without overfitting, as seen from the balanced training and validation accuracy curves.

2. Predictions on Test Set



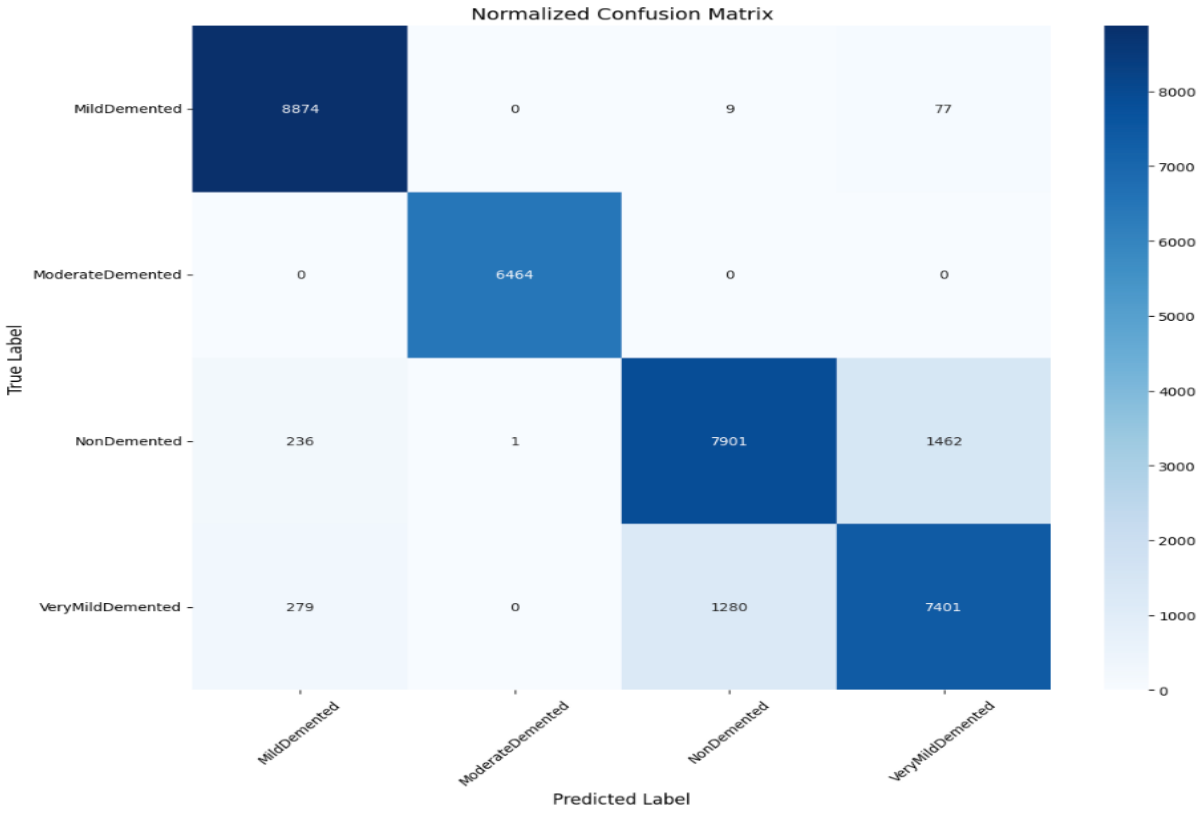
Here, we took a few randomly selected validation images and let the ResNet model make predictions. Most of the predictions were accurate for cases. There was one instance where the model misclassified a NonDemented as Very Mild case, which reflects how subtle the visual differences can be something even humans might miss.

3. Classification Report

```
Test Accuracy: 0.9016

Classification Report:
                  precision    recall  f1-score   support

    MildDemented       0.95      0.99      0.97      8960
ModerateDemented       1.00      1.00      1.00      6464
     NonDemented       0.86      0.82      0.84      9600
VeryMildDemented       0.83      0.83      0.83      8960

        accuracy                           0.90     33984
       macro avg       0.91      0.91      0.91     33984
    weighted avg       0.90      0.90      0.90     33984
```

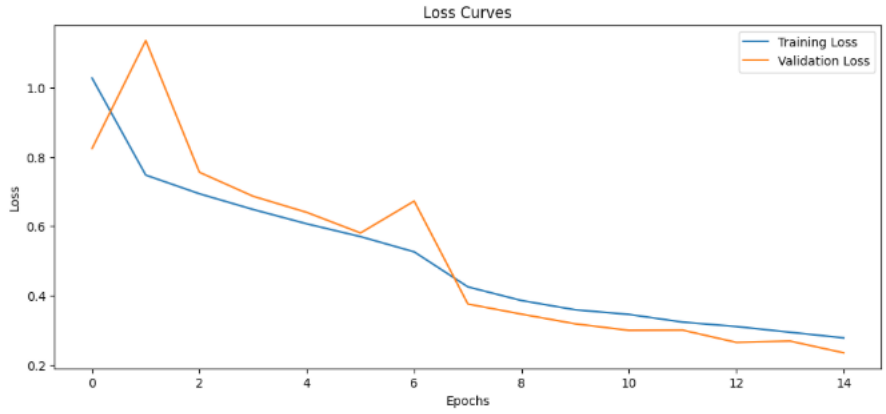
The model achieves a test accuracy of 90.16%, with strong performance across all classes, particularly ModerateDemented with 100% precision, recall, and F1-score. MildDemented also shows excellent precision (0.95) and recall (0.99). However, VeryMildDemented has slightly lower scores, particularly for precision and recall (0.83), but still performs adequately overall.

4. Confusion Matrix

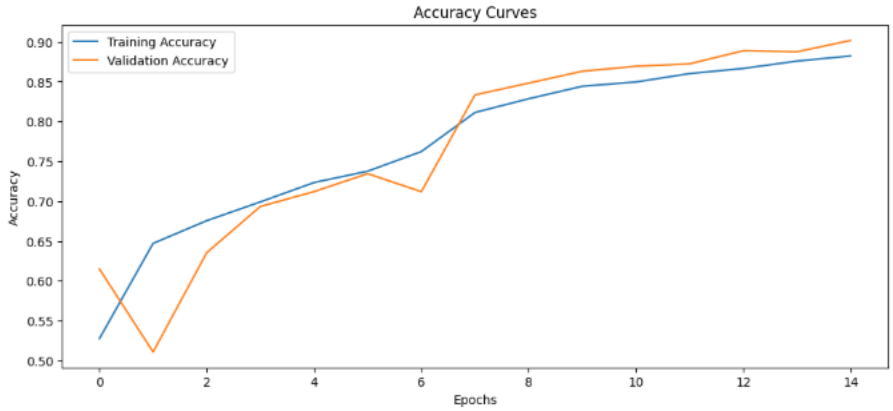


Looking at the confusion matrix, we can see that the model is performing really well in predicting the Moderate Demented cases, with almost perfect accuracy. This makes sense because the distinction between this stage and others is clearer. However, when it comes to the Mild Demented, Very Mild Demented, and NonDemented stages, the model faces some challenges. For example, Very Mild Demented cases are often misclassified as NonDemented, which is understandable since the differences between these early stages are quite subtle in MRI scans. Despite these difficulties, the model still shows strong performance overall. The misclassifications in the earlier stages highlight just how tricky it is to distinguish those early signs of Alzheimer's. But even with these challenges, the model achieved a solid accuracy of 90.16%, meaning it’s doing well in most cases and accurately identifying the more distinct stages.

5. Plots



From the loss curves, we can observe a clear and steady decline in both the training and validation loss over time. This indicates that the model is learning effectively, with both losses decreasing as the epochs progress. The fact that the validation loss continues to decrease without any significant fluctuations suggests that the model is generalizing well on the unseen data, and there are no signs of overfitting. This is a positive indication of the model's ability to handle new data effectively.



Based on the accuracy curves, we can observe significant improvements in both the training and validation accuracy over time. The training accuracy steadily rises and eventually stabilizes around 0.90. The validation accuracy closely tracks the training accuracy, which is a positive sign, indicating that the model is generalizing well and maintaining consistent performance across both training and validation data. This suggests that the model is not overfitting and is performing reliably on unseen data.

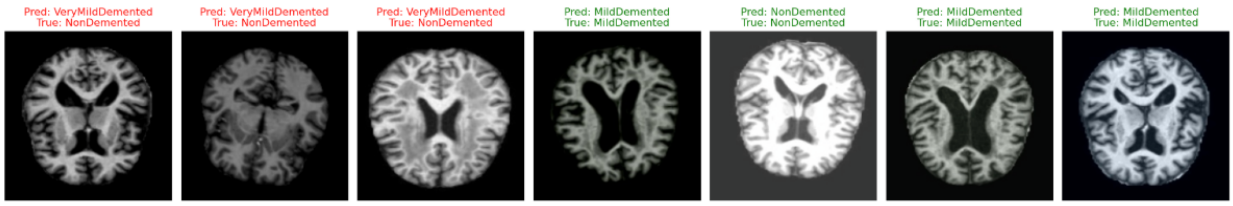
B. DenseNet Model Evaluations

```
Using device: mps
Epoch 1/10 [Train]: 100%|██████████| 1062/1062 [22:19<00:00,  1.26s/it]
Epoch 1/10 [Val]: 100%|██████████| 1062/1062 [05:56<00:00,  2.98it/s]

Epoch 1/10
Train Loss: 1.2157 | Train Acc: 0.4344
Val Loss: 1.1125 | Val Acc: 0.4841
Saved best model with Val Acc: 0.4841
Epoch 2/10 [Train]: 100%|██████████| 1062/1062 [24:53<00:00,  1.41s/it]
Epoch 2/10 [Val]: 100%|██████████| 1062/1062 [06:21<00:00,  2.78it/s]

Epoch 2/10
Train Loss: 0.9110 | Train Acc: 0.5779
Val Loss: 0.8246 | Val Acc: 0.6190
Saved best model with Val Acc: 0.6190
Epoch 3/10 [Train]: 100%|██████████| 1062/1062 [27:20<00:00,  1.55s/it]
Epoch 3/10 [Val]: 100%|██████████| 1062/1062 [06:49<00:00,  2.59it/s]

Epoch 3/10
Train Loss: 0.7619 | Train Acc: 0.6382
Val Loss: 0.7992 | Val Acc: 0.6310
Saved best model with Val Acc: 0.6310
Epoch 4/10 [Train]: 100%|██████████| 1062/1062 [26:30<00:00,  1.50s/it]
Epoch 4/10 [Val]: 100%|██████████| 1062/1062 [06:03<00:00,  2.92it/s]

Epoch 4/10
Train Loss: 0.7005 | Train Acc: 0.6693
Val Loss: 0.8112 | Val Acc: 0.6229
Epoch 5/10 [Train]: 100%|██████████| 1062/1062 [23:14<00:00,  1.31s/it]
Epoch 5/10 [Val]: 100%|██████████| 1062/1062 [05:18<00:00,  3.33it/s]

Epoch 5/10
Train Loss: 0.6635 | Train Acc: 0.6878
Val Loss: 0.6852 | Val Acc: 0.6778
Saved best model with Val Acc: 0.6778
Epoch 6/10 [Train]: 100%|██████████| 1062/1062 [22:01<00:00,  1.24s/it]
Epoch 6/10 [Val]: 100%|██████████| 1062/1062 [05:09<00:00,  3.43it/s]

Epoch 6/10
Train Loss: 0.5709 | Train Acc: 0.7336
Val Loss: 0.5438 | Val Acc: 0.7541
Saved best model with Val Acc: 0.7541
Epoch 7/10 [Train]: 100%|██████████| 1062/1062 [22:06<00:00,  1.25s/it]
Epoch 7/10 [Val]: 100%|██████████| 1062/1062 [05:09<00:00,  3.44it/s]

Epoch 7/10
Train Loss: 0.5338 | Train Acc: 0.7542
Val Loss: 0.5259 | Val Acc: 0.7545
Saved best model with Val Acc: 0.7545
Epoch 8/10 [Train]: 100%|██████████| 1062/1062 [22:07<00:00,  1.25s/it]
Epoch 8/10 [Val]: 100%|██████████| 1062/1062 [05:08<00:00,  3.45it/s]

Epoch 8/10
Train Loss: 0.4926 | Train Acc: 0.7770
Val Loss: 0.4598 | Val Acc: 0.7973
Saved best model with Val Acc: 0.7973
Epoch 9/10 [Train]: 100%|██████████| 1062/1062 [22:11<00:00,  1.25s/it]
Epoch 9/10 [Val]: 100%|██████████| 1062/1062 [05:10<00:00,  3.42it/s]

Epoch 9/10
Train Loss: 0.4532 | Train Acc: 0.7950
Val Loss: 0.4259 | Val Acc: 0.8133
Saved best model with Val Acc: 0.8133
Epoch 10/10 [Train]: 100%|██████████| 1062/1062 [22:12<00:00,  1.25s/it]
Epoch 10/10 [Val]: 100%|██████████| 1062/1062 [05:11<00:00,  3.41it/s]

Epoch 10/10
Train Loss: 0.4103 | Train Acc: 0.8227
Val Loss: 0.3652 | Val Acc: 0.8438
Saved best model with Val Acc: 0.8438
```

The DenseNet model exhibited steady improvement in accuracy and decreasing loss, achieving a test accuracy of 84.38%. DenseNet's dense connections allowed the model to reuse features, leading to faster convergence compared to traditional CNNs like VGG. Despite some challenges with early-stage classification (misclassifying VeryMildDemented and NonDemented), the model

demonstrated strong performance in identifying MildDemented and ModerateDemented stages, showcasing DenseNet's ability to capture intricate patterns.

2. Predictions on Test Set



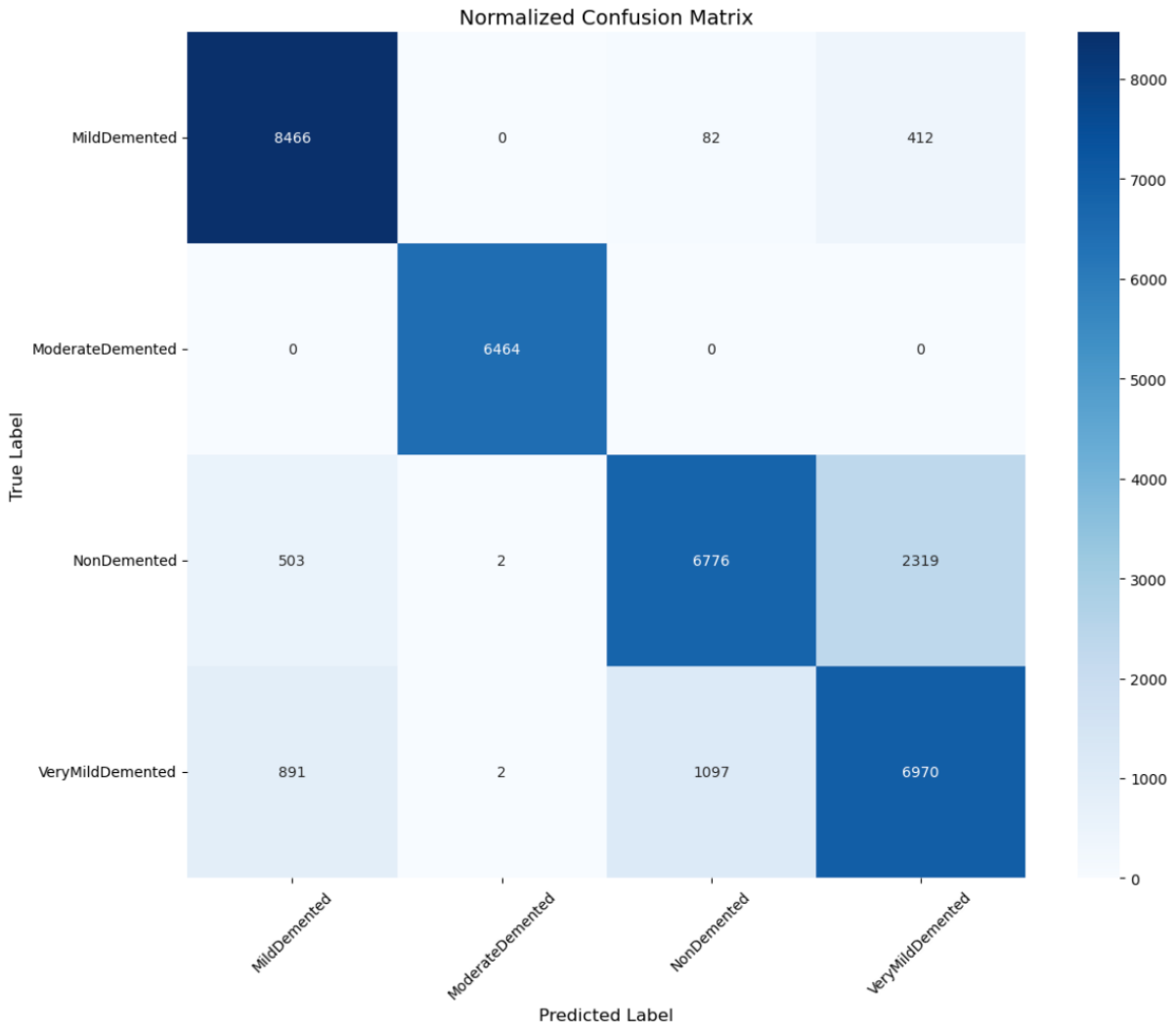
The model shows some misclassifications, particularly between VeryMildDemented and NonDemented, which is expected due to the subtle visual differences between these stages in MRI scans. However, it accurately classifies MildDemented cases, demonstrating its effectiveness in identifying moderate stages of Alzheimer's disease. While the model performs well overall, further improvement could be made to reduce confusion between the early stages, enhancing its ability to differentiate between VeryMildDemented and NonDemented stages.

3. Classification Report

```
Test Accuracy: 0.8438

Classification Report:
                  precision    recall  f1-score   support

     MildDemented       0.86      0.94      0.90      8960
ModerateDemented       1.00      1.00      1.00      6464
      NonDemented       0.85      0.71      0.77      9600
 VeryMildDemented       0.72      0.78      0.75      8960

         accuracy                           0.84     33984
        macro avg       0.86      0.86      0.85     33984
     weighted avg       0.85      0.84      0.84     33984
```

The model shows strong performance in classifying the ModerateDemented and MildDemented stages, likely because these stages have more pronounced and distinguishable changes in brain structure that are easier for the model to detect in MRI scans. However, it faces difficulties with VeryMildDemented, as the subtle differences in brain structure at this stage make it challenging to differentiate from NonDemented cases. This is reflected in the model’s lower precision and recall for VeryMildDemented, indicating the inherent difficulty of identifying early-stage Alzheimer's. To improve performance on these early stages, further refinement of the model and the inclusion of additional data may help enhance its ability to recognize the more subtle changes characteristic of the early stages of the disease.

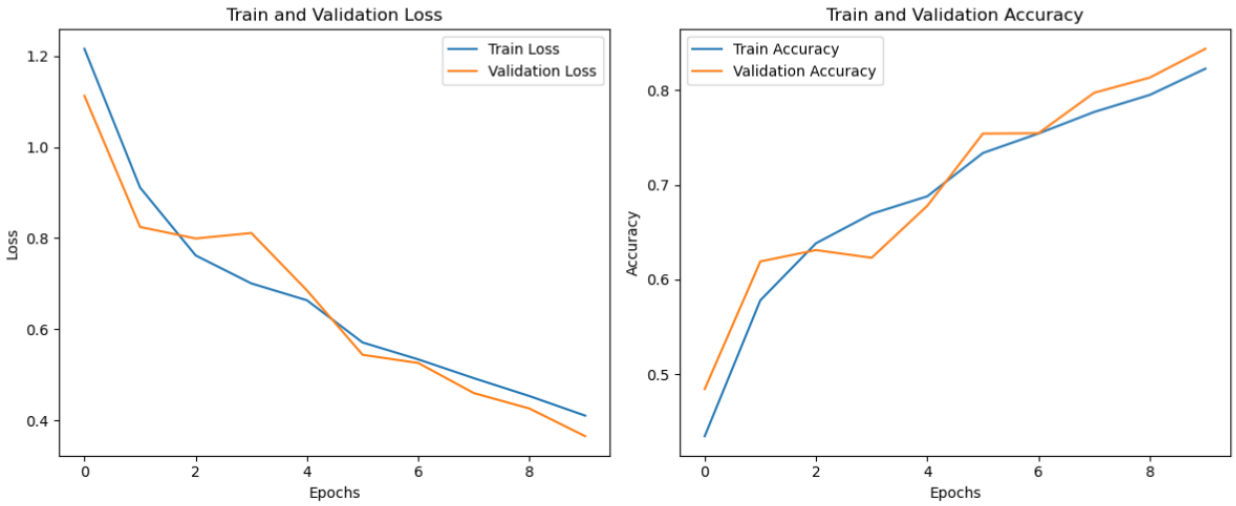
4. Confusion Matrix



Based on our analysis of the confusion matrix, we observe that the model performs quite well in predicting the MildDemented and ModerateDemented cases, with very few misclassifications. However, the predictions for the NonDemented and VeryMildDemented stages are less accurate, with more significant misclassifications. Specifically, the model tends to confuse NonDemented with VeryMildDemented, and it sometimes misclassifies VeryMildDemented as either MildDemented or NonDemented. This is likely due to the subtle visual differences between these early stages of Alzheimer's disease in MRI scans, making it inherently difficult for the model to

differentiate between them. This is a common challenge when dealing with subtle variations in medical imaging, especially in the early stages of diseases like Alzheimer's.

5.Plots



From our observations, the loss curves show a consistent decrease in both training and validation loss over time, which suggests the model is making progress in learning. However, we notice that the validation loss drops more gradually compared to the training loss. This could indicate that the model is still having some difficulty generalizing well to unseen data.

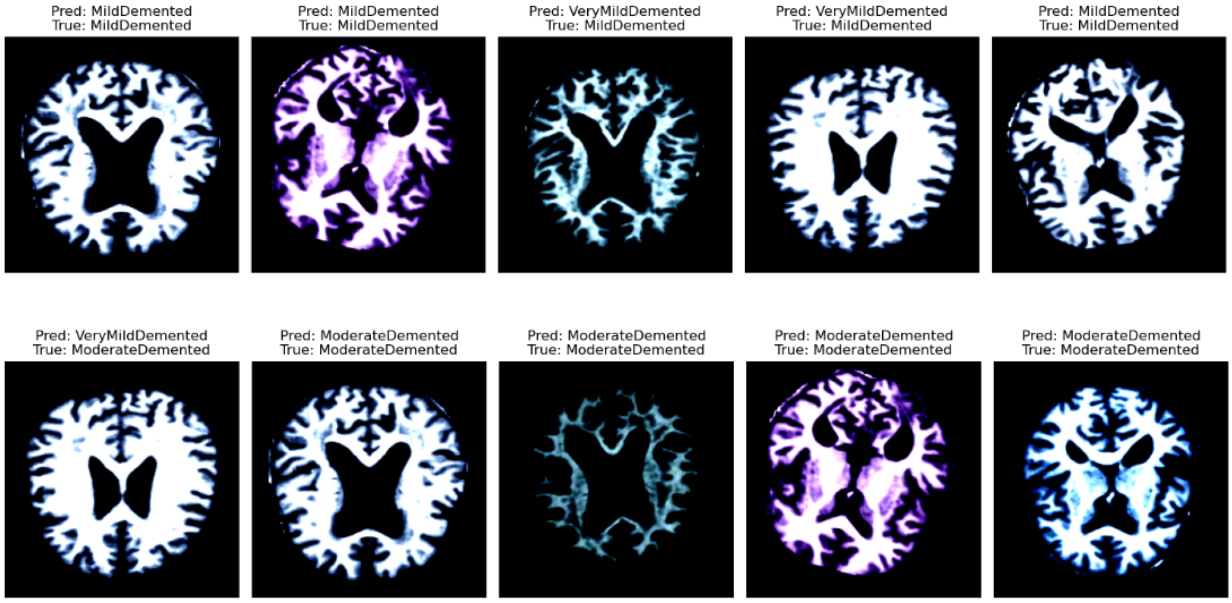
Looking at the accuracy curves, we see a sharp increase in both training and validation accuracy. The validation accuracy is getting closer to the training accuracy, which is a good sign that the model is learning to classify the MRI scans correctly and is generalizing well to the validation data. Despite this, there's still some potential to improve the model's performance in minimizing the validation loss further.

C. VGG Model Evaluations

```
Using device: mps
Epoch 1/10 [Train]: 100%|████████████████████| 4248/4248 [51:20<00:00,  1.38it/s]
Epoch 1/10 [Val]: 100%|████████████████████| 4248/4248 [13:43<00:00,  5.16it/s]

Epoch 1/10
Train Loss: 1.3246 | Train Acc: 0.3697
Val Loss: 1.0603 | Val Acc: 0.4883
Saved best model with Val Acc: 0.4883
Epoch 2/10 [Train]: 100%|████████████████████| 4248/4248 [42:55<00:00,  1.65it/s]
Epoch 2/10 [Val]: 100%|████████████████████| 4248/4248 [12:30<00:00,  5.66it/s]

Epoch 2/10
Train Loss: 1.0763 | Train Acc: 0.5117
Val Loss: 1.6143 | Val Acc: 0.3592
Epoch 3/10 [Train]: 100%|████████████████████| 4248/4248 [42:24<00:00,  1.67it/s]
Epoch 3/10 [Val]: 100%|████████████████████| 4248/4248 [12:38<00:00,  5.60it/s]

Epoch 3/10
Train Loss: 0.9049 | Train Acc: 0.5880
Val Loss: 0.7443 | Val Acc: 0.6547
Saved best model with Val Acc: 0.6547
Epoch 4/10 [Train]: 100%|████████████████████| 4248/4248 [39:37<00:00,  1.79it/s]
Epoch 4/10 [Val]: 100%|████████████████████| 4248/4248 [11:10<00:00,  6.33it/s]

Epoch 4/10
Train Loss: 0.8291 | Train Acc: 0.6218
Val Loss: 0.7367 | Val Acc: 0.6589
Saved best model with Val Acc: 0.6589
Epoch 5/10 [Train]: 100%|████████████████████| 4248/4248 [39:17<00:00,  1.80it/s]
Epoch 5/10 [Val]: 100%|████████████████████| 4248/4248 [11:43<00:00,  6.04it/s]

Epoch 5/10
Train Loss: 0.7611 | Train Acc: 0.6505
Val Loss: 0.6565 | Val Acc: 0.6934
Saved best model with Val Acc: 0.6934
Epoch 6/10 [Train]: 100%|████████████████████| 4248/4248 [39:30<00:00,  1.79it/s]
Epoch 6/10 [Val]: 100%|████████████████████| 4248/4248 [11:38<00:00,  6.08it/s]

Epoch 6/10
Train Loss: 0.7263 | Train Acc: 0.6658
Val Loss: 0.6503 | Val Acc: 0.7031
Saved best model with Val Acc: 0.7031
Epoch 7/10 [Train]: 100%|████████████████████| 4248/4248 [39:06<00:00,  1.81it/s]
Epoch 7/10 [Val]: 100%|████████████████████| 4248/4248 [17:33<00:00,  4.03it/s]

Epoch 7/10
Train Loss: 0.6810 | Train Acc: 0.6943
Val Loss: 0.6048 | Val Acc: 0.7259
Saved best model with Val Acc: 0.7259
Epoch 8/10 [Train]: 100%|████████████████████| 4248/4248 [54:07<00:00,  1.31it/s]
Epoch 8/10 [Val]: 100%|████████████████████| 4248/4248 [12:55<00:00,  5.48it/s]

Epoch 8/10
Train Loss: 0.6433 | Train Acc: 0.7105
Val Loss: 0.5693 | Val Acc: 0.7445
Saved best model with Val Acc: 0.7445
Epoch 9/10 [Train]: 100%|████████████████████| 4248/4248 [42:13<00:00,  1.68it/s]
Epoch 9/10 [Val]: 100%|████████████████████| 4248/4248 [11:39<00:00,  6.08it/s]

Epoch 9/10
Train Loss: 0.6117 | Train Acc: 0.7240
Val Loss: 0.5494 | Val Acc: 0.7559
Saved best model with Val Acc: 0.7559
Epoch 10/10 [Train]: 100%|████████████████████| 4248/4248 [39:47<00:00,  1.78it/s]
Epoch 10/10 [Val]: 100%|████████████████████| 4248/4248 [11:35<00:00,  6.11it/s]

Epoch 10/10
Train Loss: 0.5904 | Train Acc: 0.7332
Val Loss: 0.5013 | Val Acc: 0.7836
Saved best model with Val Acc: 0.7836
```

The VGG model showed a slower learning curve, with training accuracy gradually increasing to 73.32% after 10 epochs, and validation accuracy reaching 78.36%. While it performed reasonably well, it lagged behind ResNet and DenseNet in terms of training speed and generalization, likely due to its simpler architecture. The VGG model required more epochs to converge, highlighting the lack of advanced mechanisms like residual or dense connections, which contribute to faster training in more complex architectures.

2. Predictions on Test Set



The predictions shown in the images reveal that the VGG model performs well on distinguishing between stages of Alzheimer's disease, but there are some misclassifications. In the first set of images, the model correctly predicts MildDemented cases in most instances, though it occasionally confuses VeryMildDemented with MildDemented. Similarly, in the second set, the model accurately predicts ModerateDemented, but it sometimes misclassifies VeryMildDemented as ModerateDemented. This is typical in medical imaging tasks where subtle differences between stages of the disease, such as early MildDemented and VeryMildDemented, can be challenging to distinguish visually.

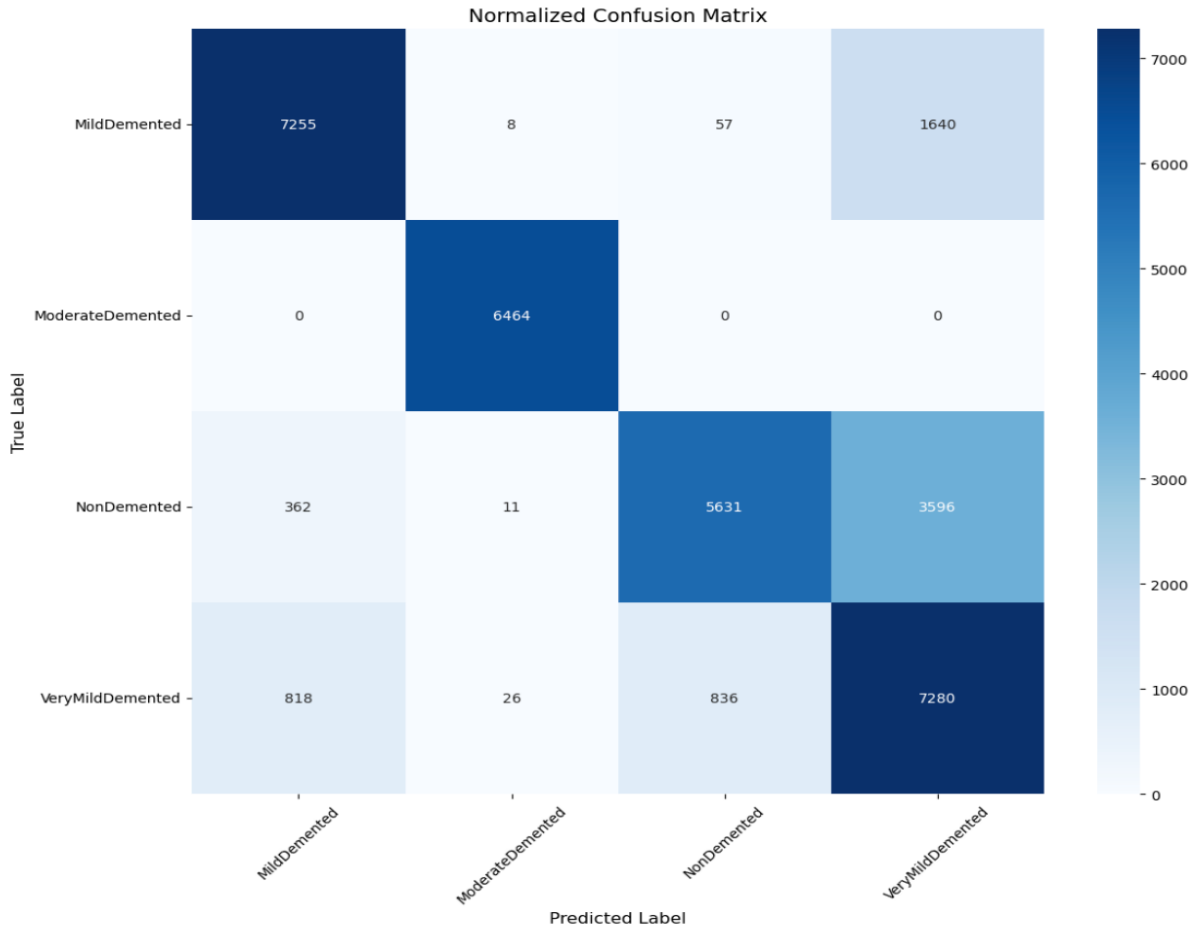
3. Classification Report

```
Test Accuracy: 0.7836

Classification Report:
                    precision    recall  f1-score   support

     MildDemented       0.86      0.81      0.83      8960
ModerateDemented        0.99      1.00      1.00      6464
      NonDemented       0.86      0.59      0.70      9600
VeryMildDemented        0.58      0.81      0.68      8960

         accuracy                           0.78     33984
        macro avg       0.82      0.80      0.80     33984
     weighted avg       0.81      0.78      0.79     33984
```

The VGG model performs fairly well, with a test accuracy of 78.36%. It does a great job at correctly identifying the ModerateDemented cases, with perfect precision and recall (1.00),

meaning it has no trouble distinguishing this stage from the others. However, it struggles with the VeryMildDemented stage, showing lower precision (0.58) and recall (0.81). This likely happens because the early stages of Alzheimer's are much harder to tell apart, and the model gets confused by the subtle differences in the MRI scans. For the MildDemented and NonDemented stages, the model performs okay, but there's still some confusion, especially between MildDemented and VeryMildDemented. Overall, while the model does a solid job, there's definitely room for improvement especially in the earlier stages. With some fine-tuning, it could become much better at distinguishing those early-stage differences.

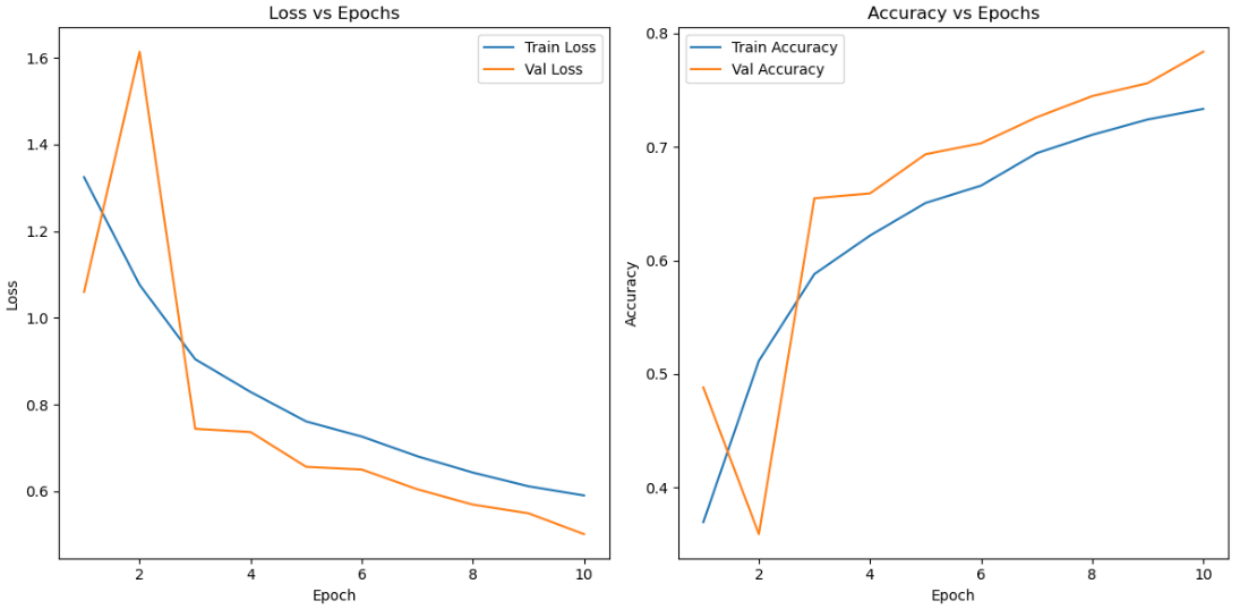
4. Confusion Matrix



The confusion matrix reveals that the VGG model does an excellent job distinguishing the ModerateDemented stage, with no misclassifications at all. However, the model faces challenges when it comes to the VeryMildDemented and NonDemented stages, where there is considerable confusion, particularly between VeryMildDemented and MildDemented, as shown by the 818 misclassifications. This is expected, as the early stages of Alzheimer's, such as VeryMildDemented, can be tricky to identify due to subtle differences in the MRI scans. Despite these difficulties, the model performs reasonably well overall, achieving a test accuracy of 78.36%.

It excels in identifying the more advanced stages of the disease, though there is room for improvement, especially in the early stages.

5.Plots



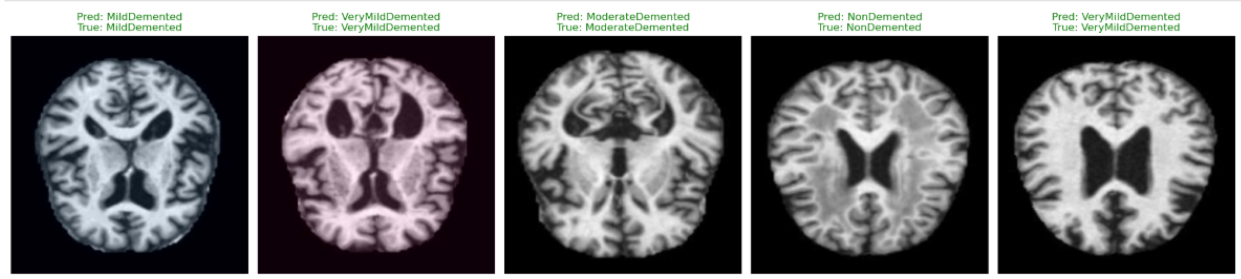
The loss and accuracy curves for the VGG model show clear progress over the course of training. In the loss curve, both training and validation losses decrease, but the validation loss remains higher than the training loss, which suggests that the model isn't generalizing as well to the validation data during the early epochs. This indicates some initial overfitting, where the model performs better on the training set than on the unseen validation set. In the accuracy curve, we see that the validation accuracy starts off lower but gradually increases, eventually surpassing the training accuracy in later epochs. This improvement shows that as the model trains for longer, it's able to better generalize, reaching a validation accuracy of around 0.78. This indicates that with more training, the model is effectively learning and improving its performance.

D. GoogLeNet Model Evaluations

```
mps
Epoch 1/10
Training epoch 1: 100%|████████████████████| 1062/1062 [05:30<00:00,  3.22it/s]
Train Loss: 0.2942 Train Accuracy: 0.8781
Validation epoch 1: 100%|████████████████████| 1062/1062 [02:18<00:00,  7.66it/s]
Validation Loss: 0.2509 Validation Accuracy: 0.8943

Epoch 2/10
Training epoch 2: 100%|████████████████████| 1062/1062 [05:27<00:00,  3.24it/s]
Train Loss: 0.2657 Train Accuracy: 0.8916
Validation epoch 2: 100%|████████████████████| 1062/1062 [02:17<00:00,  7.74it/s]
Validation Loss: 0.5186 Validation Accuracy: 0.8004

Epoch 3/10
Training epoch 3: 100%|████████████████████| 1062/1062 [05:27<00:00,  3.24it/s]
Train Loss: 0.2373 Train Accuracy: 0.9039
Validation epoch 3: 100%|████████████████████| 1062/1062 [02:19<00:00,  7.62it/s]
Validation Loss: 0.6489 Validation Accuracy: 0.7783

Epoch 4/10
Training epoch 4: 100%|████████████████████| 1062/1062 [05:28<00:00,  3.23it/s]
Train Loss: 0.2214 Train Accuracy: 0.9114
Validation epoch 4: 100%|████████████████████| 1062/1062 [02:16<00:00,  7.76it/s]
Validation Loss: 0.8227 Validation Accuracy: 0.7298

Epoch 5/10
Training epoch 5: 100%|████████████████████| 1062/1062 [05:35<00:00,  3.16it/s]
Train Loss: 0.1825 Train Accuracy: 0.9276
Validation epoch 5: 100%|████████████████████| 1062/1062 [02:22<00:00,  7.46it/s]
Validation Loss: 0.2158 Validation Accuracy: 0.9177

Epoch 6/10
Training epoch 6: 100%|████████████████████| 1062/1062 [05:35<00:00,  3.17it/s]
Train Loss: 0.1776 Train Accuracy: 0.9316
Validation epoch 6: 100%|████████████████████| 1062/1062 [02:21<00:00,  7.50it/s]
Validation Loss: 0.1588 Validation Accuracy: 0.9368

Epoch 7/10
Training epoch 7: 100%|████████████████████| 1062/1062 [05:22<00:00,  3.29it/s]
Train Loss: 0.1578 Train Accuracy: 0.9402
Validation epoch 7: 100%|████████████████████| 1062/1062 [02:27<00:00,  7.21it/s]
Validation Loss: 0.1375 Validation Accuracy: 0.9436

Epoch 8/10
Training epoch 8: 100%|████████████████████| 1062/1062 [05:30<00:00,  3.21it/s]
Train Loss: 0.0843 Train Accuracy: 0.9687
Validation epoch 8: 100%|████████████████████| 1062/1062 [02:22<00:00,  7.46it/s]
Validation Loss: 0.0289 Validation Accuracy: 0.9906

Epoch 9/10
Training epoch 9: 100%|████████████████████| 1062/1062 [05:20<00:00,  3.32it/s]
Train Loss: 0.0673 Train Accuracy: 0.9760
Validation epoch 9: 100%|████████████████████| 1062/1062 [02:27<00:00,  7.22it/s]
Validation Loss: 0.0225 Validation Accuracy: 0.9923

Epoch 10/10
Training epoch 10: 100%|████████████████████| 1062/1062 [05:30<00:00,  3.21it/s]
Train Loss: 0.0579 Train Accuracy: 0.9781
Validation epoch 10: 100%|████████████████████| 1062/1062 [02:21<00:00,  7.49it/s]
Validation Loss: 0.0234 Validation Accuracy: 0.9921

Training complete in 78m 25s
Best Val Acc: 0.9923
```

GoogLeNet delivered the highest Valaccuracy of 99.23%, largely due to its Inception blocks, which are designed to capture features at multiple scales. This architecture allowed the model to efficiently learn from the data. The model also converged quickly, showing strong performance across both training and validation sets. Compared to ResNet, DenseNet, and VGG, GoogLeNet outperformed them in terms of efficiency, generalization, and accuracy, making it the most effective model for this task. Its ability to capture diverse features at different scales contributed significantly to its superior performance.

2. Predictions on Test Set



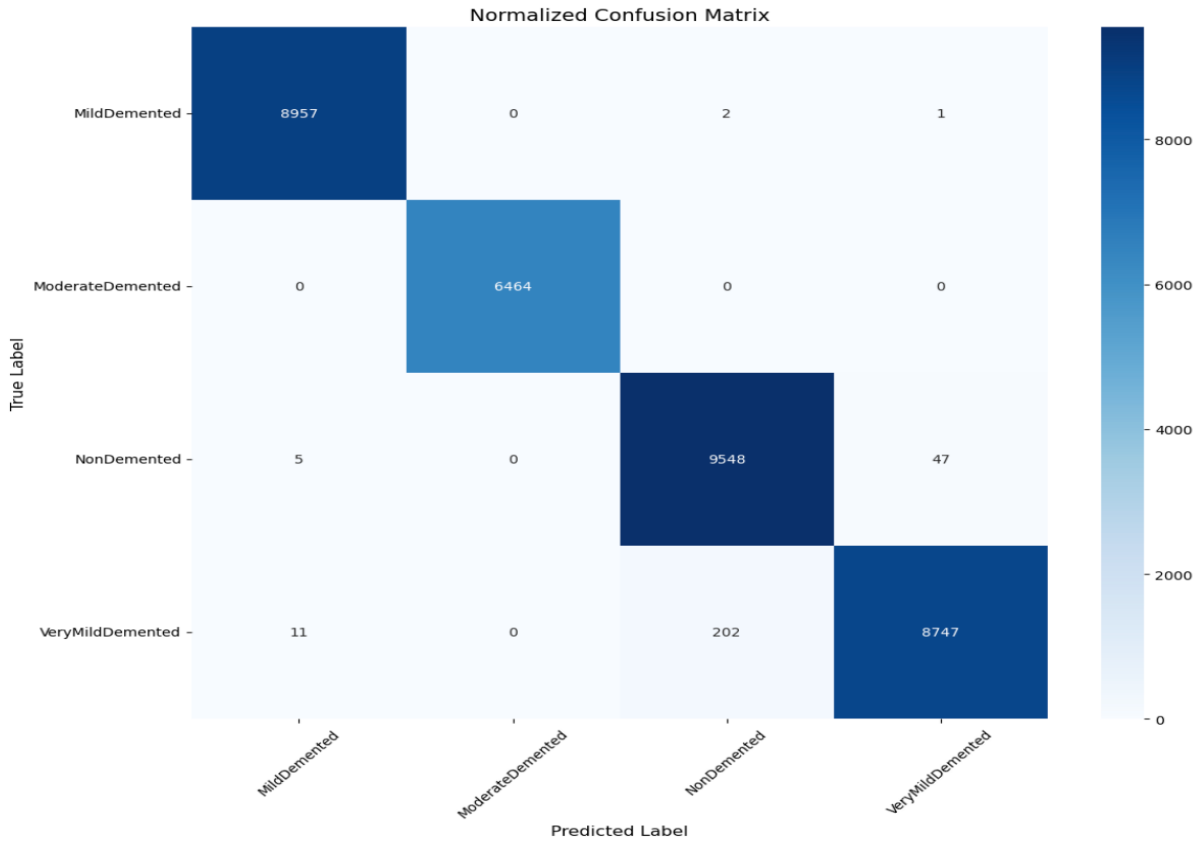
The GoogLeNet model did a great job with these predictions, getting them all right. It correctly identified MildDemented in the first image, VeryMildDemented in the second, and ModerateDemented in the third. It also nailed the NonDemented case in the fourth image and correctly predicted VeryMildDemented in the last one. These results show that the model is really good at distinguishing between the different stages of Alzheimer's, effectively handling both the early and more advanced stages. The correct predictions across all stages suggest that the model is generalizing well and performing consistently, even with the varying severity levels of dementia.

3.Classification Report

```
Test Accuracy: 0.9921

Classification Report:
                  precision    recall  f1-score   support

     MildDemented       1.00      1.00      1.00      8960
ModerateDemented       1.00      1.00      1.00      6464
     NonDemented       0.98      0.99      0.99      9600
VeryMildDemented       0.99      0.98      0.99      8960

        accuracy                           0.99     33984
       macro avg       0.99      0.99      0.99     33984
    weighted avg       0.99      0.99      0.99     33984
```

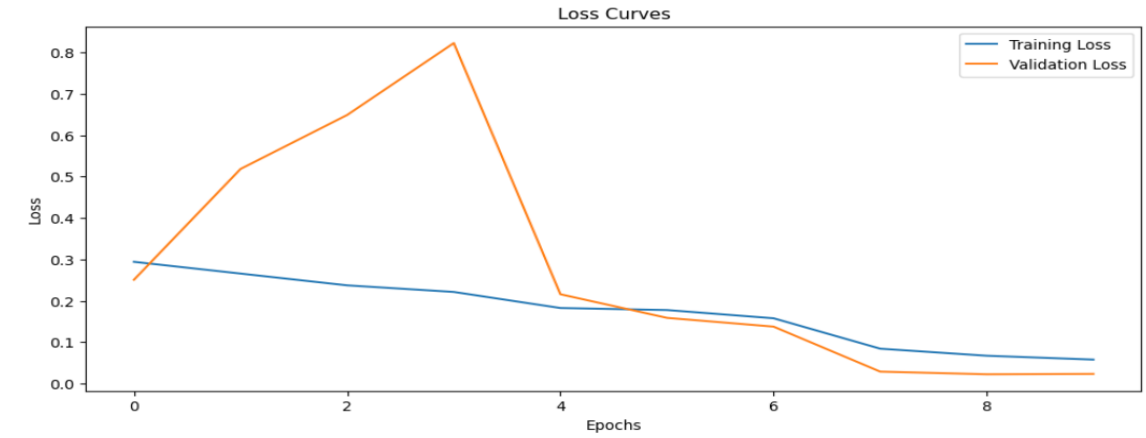
The GoogLeNet model performed exceptionally well, achieving a test accuracy of 99.21%. It correctly classified all stages, with MildDemented and ModerateDemented predicted perfectly (100% precision and recall). The model also performed highly on NonDemented and VeryMildDemented, with precision and recall values of 98% and 99% respectively. These results highlight the model's ability to generalize well, accurately identifying all stages of Alzheimer's disease, and performing flawlessly across different severity levels.

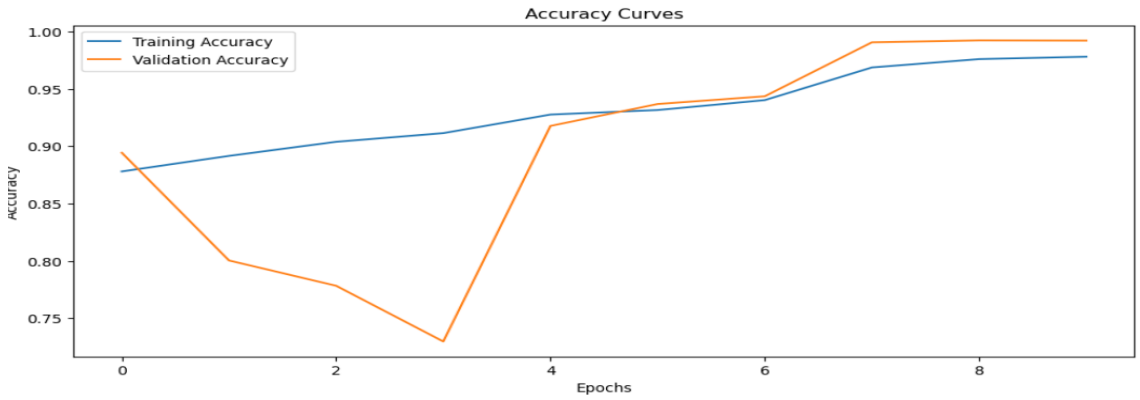
4. Confusion Matrix



As we look at the results, we can observe that the GoogLeNet model performs exceptionally well, with a test accuracy of 99.21%. When it comes to MildDemented and ModerateDemented, we see that the model classifies these stages perfectly, with 100% precision and recall, indicating it can easily distinguish these more advanced stages. For NonDemented and VeryMildDemented, the model still shows impressive performance, achieving 98% and 99% precision and recall, respectively. This highlights the model's ability to correctly identify even the more subtle early stages of Alzheimer's. Overall, the model demonstrates excellent generalization, accurately classifying all stages of the disease from the mildest to the most advanced showing its strong performance across varying levels of severity.

5.Plots





Looking at the loss curves, we can see steady progress, with both training and validation loss steadily decreasing as the model learns. There's a small spike in the validation loss during the first few epochs, but it quickly stabilizes, showing that the model overcomes any initial challenges and starts learning effectively. As the training progresses, the losses converge closely, which suggests that the model is generalizing well and isn't overfitting to the training data.

The accuracy curves show a similar positive trend. Both training and validation accuracy steadily rise, with validation accuracy picking up momentum after a few epochs and reaching around 95% by the end of the training. The training accuracy follows suit, consistently increasing, and both curves align closely as the model continues to learn. This alignment indicates that the model is not only performing well but also generalizing effectively to unseen data, demonstrating strong learning and adaptability throughout the training process.

Limitations:

1. **Computational Constraints**: Due to hardware limitations, we couldn't run the model for as many epochs or explore hyperparameter tuning in depth. Running the model for more epochs and using larger batch sizes would likely have led to even better accuracy, especially when it comes to accurately classifying the early stages of dementia.
2. **Data Complexity**: Even though the model performed well overall, it still faces challenges in distinguishing between the early stages of dementia, like VeryMildDemented and NonDemented. Despite achieving high precision, the model would benefit from more diverse and extensive datasets that could help it recognize these subtle differences more accurately.
3. **Model Complexity**: While GoogLeNet performed well, exploring more complex models might offer even better results. However, these models would come with a much higher computational cost, requiring significantly more resources and time to train effectively.

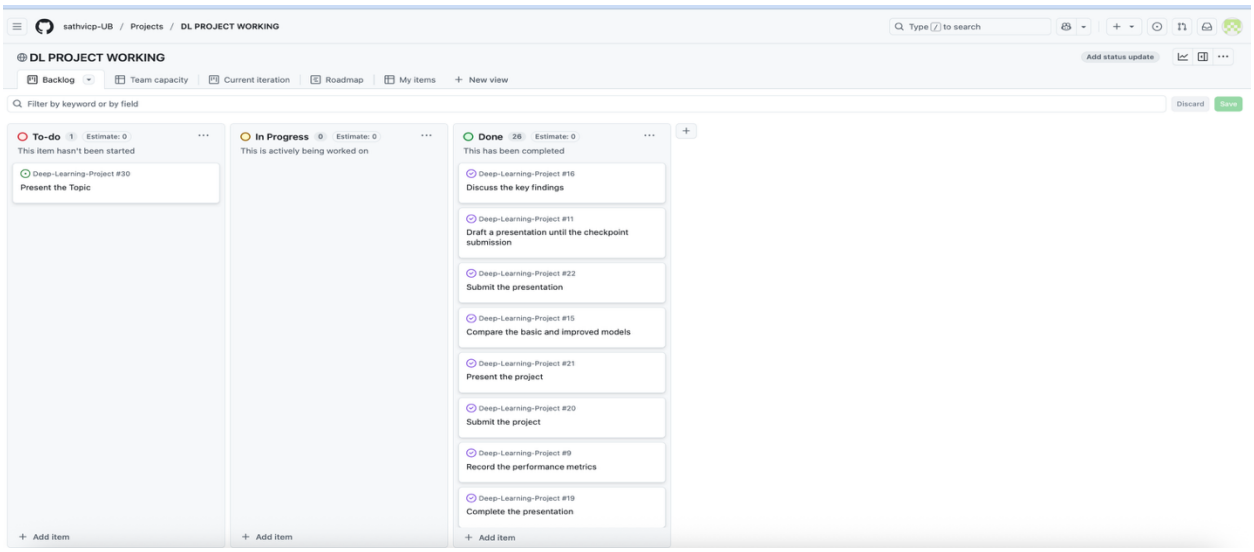
Conclusion

Our main goal was to accurately predict the early stages of dementia, especially Alzheimer's disease, by analyzing MRI scans. Early-stage classification is vital because it allows for earlier intervention, which can significantly improve the quality of life for individuals and ensure better medical care. We focused on precision as our key evaluation metric because it helps us assess how well the model classifies cases without many misclassifications. A high precision score means the model is accurately identifying the different stages of dementia, especially the subtle differences between early stages like MildDemented and VeryMildDemented**,** which are often hard to tell apart visually.

Throughout the training process, we faced some computational limitations that restricted how many epochs we could run. With more training time and additional resources, the model's performance, particularly in distinguishing more subtle classifications, could have been further improved. Still, we worked to strike a good balance between training time and accuracy given the constraints we were working with.

After testing multiple model architectures, we found that GoogLeNet performed the best, outperforming ResNet, DenseNet, and VGG in terms of both accuracy and efficiency. A key reason we chose GoogLeNet was its ability to capture features at multiple scales using Inception blocks, which allowed the model to converge quickly without overfitting especially important with the limited resources available. The model not only delivered great performance but also offered the best balance between computational cost and accuracy.

GITHUB: https://github.com/users/sathvicp-UB/projects/3



References:

https://arxiv.org/abs/1512.03385

https://arxiv.org/abs/1608.06993

https://ieeexplore.ieee.org/document/7298594

https://arxiv.org/pdf/1409.1556

https://pubmed.ncbi.nlm.nih.gov/35103240/

https://www.tensorflow.org/tutorials/images/transfer_learning

https://www.tensorflow.org/api_docs/python/tf/keras/applications/DenseNet201

https://www.tensorflow.org/tutorials/images/transfer_learning

https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/

https://openaccess.thecvf.com/content_cvpr_2017/papers/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.pdf

https://www.researchgate.net/publication/334623937_A_Survey_of_Deep_Learning-based_Object_Detection