LOW LEVEL DESIGN (LLD)

Alzheimer Disease Classifier

Document Version: 1.0

Last Revised Date: 27-02-2025

Written By: Jatindra Paul

Document Control

Change Record

Version	Date	Author	Comments
1.0	27-02-2025	Jatindra Paul	Initial version of the document

Table of Contents

1. Introduction

1.1 What is a Low-Level Design Document?

The **Low-Level Design (LLD)** document provides detailed information about the internal structure of the **Alzheimer Disease Classifier** system. It describes class structures, module functionalities, and the relationships between various components. The purpose of this document is to provide a blueprint for implementing the classifier using **VGG16** for **binary classification** (Alzheimer's Positive vs. Negative).

1.2 Scope

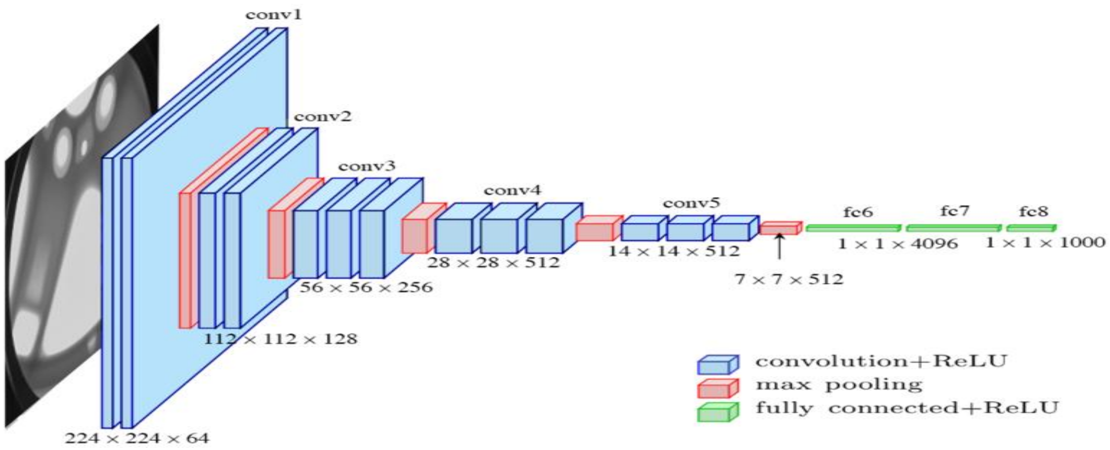
The **LLD** includes:

- Data ingestion and preprocessing steps.
- Model architecture using **VGG16** and fine-tuned layers.
- Training methodologies and evaluation metrics.
- Model inference and deployment strategy.
- Unit test cases to validate each module.

2. Architecture

Workflow Overview

1. **Data Acquisition**
    - Medical images (MRI scans) are collected.
    - Data is stored securely for further processing.
2. **Pre-processing & Augmentation**
    - Image resizing, normalization, and augmentation.
3. **Model Training (VGG16-based)**
    - Train model with labelled Alzheimer's images.
    - Use transfer learning with fine-tuned layers.
4. **Model Evaluation**
    - Validate performance using metrics (Accuracy, AUC, Precision, Recall).
5. **Model Inference & Deployment**
    - Deploy model via API.
    - Accept image input and return classification results.



3. Architecture Description

3.1 Data Description

- **Dataset:** MRI scan images labelled for Alzheimer's detection.
- **Format:** Images stored in **JPEG/PNG** format.
- **Classes:** Binary classification – *Alzheimer's Positive* and *Alzheimer's Negative*.

3.2 Data Pre-processing

- **Image Resizing:** Convert images to **224x224 pixels** for VGG16.
- **Normalization:** Scale pixel values between **0 and 1**.
- **Augmentation:** Apply techniques like **rotation, flipping, and brightness adjustment** to improve model robustness.
- **Dataset Splitting:**
  - **Training Set:** 80%
  - **Validation Set:** 10%
  - **Test Set:** 10%

3.3 Model Training

- **Base Model:** VGG16 pre-trained on **ImageNet**.
- **Fine-Tuning:**

  - Freeze convolutional layers to preserve pre-trained features.
  - Add new dense layers with dropout for regularization.
- **Loss Function:** Binary Cross-Entropy.
- **Optimizer:** Adam.
- **Metrics:**
  - Accuracy
  - AUC-ROC
  - Precision/Recall
- **Training Method:**
  - **Epochs:** 20-50 (based on early stopping).
  - **Batch Size:** 32 or 64.

3.4 Model Evaluation

- **Validation Metrics:**
  - **Confusion Matrix:** Evaluates TP, FP, FN, TN.
  - **AUC-ROC Curve:** Measures model discrimination.
  - **Precision-Recall Curve:** Assesses model effectiveness.
- **Cross-validation:** Used to prevent overfitting.

3.5 Model Inference and Deployment

- **Inference Pipeline:**
  - Accepts images via API.
  - Applies pre-processing (resizing, normalization).
  - Runs inference using trained model.
  - Returns probability scores and classification labels.
- **Deployment Strategy:**
  - Model hosted using **Flask / FastAPI**.
  - Containerized using **Docker**.
  - Deployed on **AWS / GCP / Azure** for scalability.
- **Security Considerations:**

- Data encryption in transit and at rest.
- User authentication for API access.

4. Unit Test Cases

Test Case	Pre-Requisite	Expected Result
Verify image upload functionality	API is running	Image should be successfully uploaded
Verify image pre-processing	Image is uploaded	Image should be resized and normalized
Verify model inference	Pre-processed image is ready	Model should return classification output
Verify API response time	API is running	Response time should be < 500ms
Verify model accuracy	Trained model is available	Accuracy should be above 85%

Test Case	Pre-Requisite	Expected Result
Verify security authentication	API requires authentication	Only authenticated users can access results

Conclusion

This **Low-Level Design (LLD)** document provides a structured approach to developing the **Alzheimer Disease Classifier** using a **VGG16-based model**. It outlines data processing, model training, inference, deployment, and security considerations, ensuring a scalable and efficient classification system.