High-Level Design (HLD) Document

Alzheimer Disease Classifier

Document Version: 1.0

Date: 27-02-2025

Author(s): Jatindra Paul

1. Document Revision History

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

2. Table of Contents

3. Introduction

3.1 Purpose

This document provides a high-level design for an Alzheimer Disease classifier that leverages a modified VGG16 network for binary classification (Alzheimer's Positive vs. Alzheimer's Negative). It outlines the system architecture, detailed module designs, interfaces, and operational considerations.

3.2 Scope

- **Application Domain:** Medical image analysis for Alzheimer's disease diagnosis.
- **System Functionality:** Ingest medical imaging data (e.g., MRI scans), preprocess images, classify images using a deep learning model, and deliver predictions via an API and/or user interface.
- **Target Audience:** Data scientists, software engineers, medical IT teams, and clinical researchers.

3.3 Definitions and Acronyms

- **VGG16:** A deep convolutional neural network model pre-trained on ImageNet.
- **HLD:** High-Level Design.
- **API:** Application Programming Interface.
- **MRI:** Magnetic Resonance Imaging.
- **CNN:** Convolutional Neural Network.

---

4. System Overview

4.1 System Description

The Alzheimer Disease classifier system is designed to:

- Process medical images using a standardized pipeline.
- Employ a VGG16-based model fine-tuned for binary classification.
- Provide real-time inference results through an API and a user-friendly interface.
- Log and monitor predictions for continuous model evaluation and retraining.

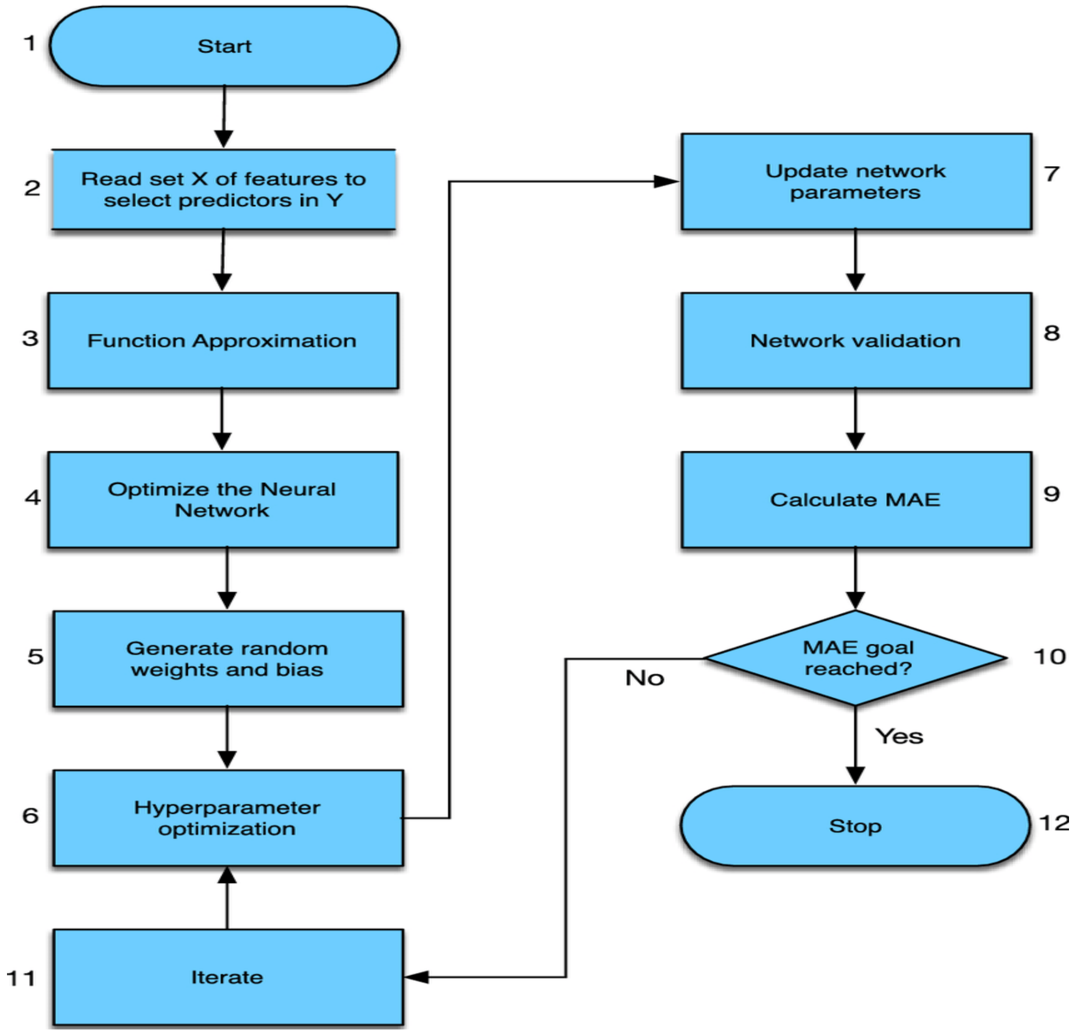
4.2 High-Level Requirements

- **Functional Requirements:**
  - Accept and preprocess medical images.
  - Classify images into two classes: Alzheimer's Positive and Alzheimer's Negative.
  - Return prediction probabilities with confidence metrics.
  - Maintain logs for each prediction.
- **Non-Functional Requirements:**
  - Ensure compliance with data privacy standards (e.g., HIPAA).
  - Achieve near real-time inference with minimal latency.
  - Scale to handle a growing volume of images and concurrent requests.

---

5. System Architecture

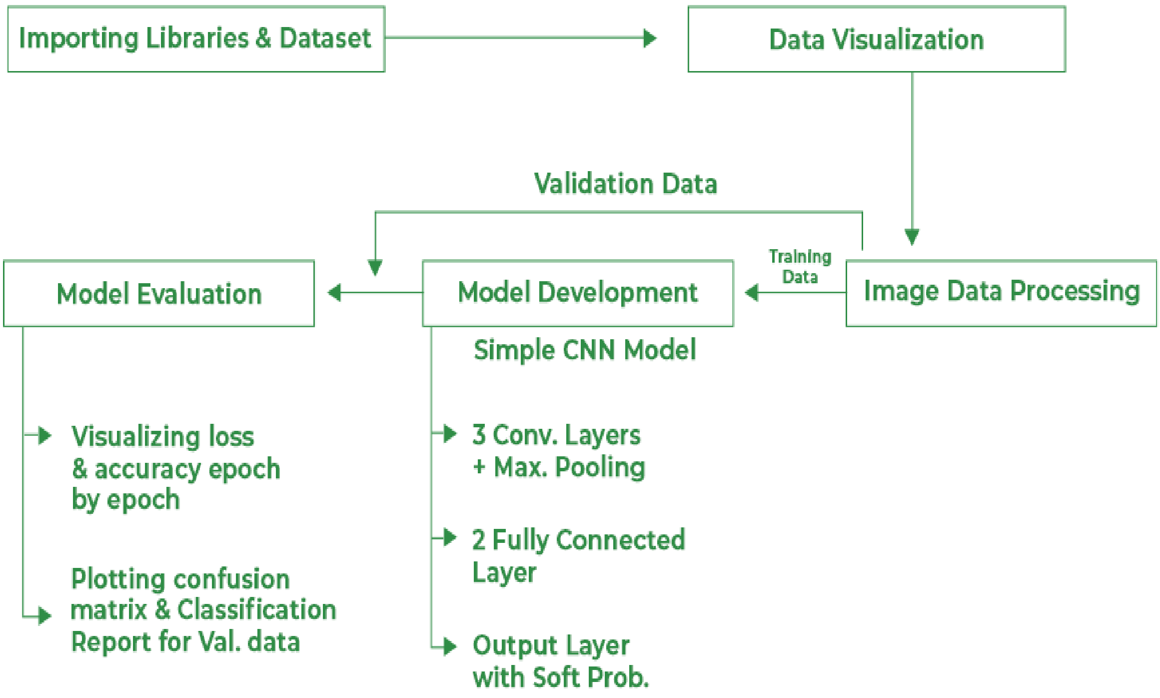
5.1 Architectural Diagram



5.2 Data Flow

1. **Data Ingestion:** Medical images are collected and securely transmitted to the system.
2. **Pre-processing:** Images are normalized, resized (e.g., to 224x224 pixels), and augmented.
3. **Model Inference:** The processed images are fed into the VGG16-based classifier.
4. **Output Generation:** The inference engine outputs prediction probabilities and binary class labels.

5. **Result Delivery:** Predictions are returned through API endpoints or displayed on a user interface.



6. Detailed Design

6.1 Data Acquisition and Pre-processing Module

- **Input:** MRI scans and other medical images labeled for Alzheimer's diagnosis.
- **Processing Steps:**
    - **Normalization:** Rescale pixel values.
    - **Resizing:** Convert images to 224x224 pixels (suitable for VGG16).
    - **Augmentation:** Apply techniques (rotation, flip, brightness adjustments) to improve model robustness.
- **Output:** Preprocessed image tensors ready for training and inference.

6.2 Model Training Module

- **Base Model:** VGG16 pre-trained on ImageNet.
- **Customization:**
    - Freeze convolutional layers to leverage pre-trained features.
    - Replace the final classification layers with a custom fully connected network:
        - Global Average Pooling / Flatten layer.
        - One or more Dense layers with dropout for regularization.
        - A final Dense layer with sigmoid activation for binary classification.
- **Training Setup:**
    - **Loss Function:** Binary Cross-Entropy.
    - **Optimizer:** Adam (or equivalent).
    - **Metrics:** Accuracy, AUC, Precision, Recall.
    - **Hyperparameters:** Epochs, batch size, learning rate, etc.

6.3 Inference Engine

- **Function:** Serve the trained model for real-time image classification.
- **Implementation:**
    - Mirror the pre-processing pipeline used during training.

- Integrate the model using a server framework (e.g., Flask, FastAPI, or TensorFlow Serving).
- Return prediction outputs as probability scores along with binary classification.

6.4 API / User Interface Module

- **API Endpoints:**
  - **/predict:** Accepts image uploads and returns classification results.
  - **/train:** Returns service status.
- **User Interface:**
  - Web-based interface for clinicians to upload images and view classification outcomes.
  - Visualization components to display prediction confidence and historical data.

7. Security and Compliance

- **Data Privacy:** Ensure that all medical data is encrypted in transit and at rest. Follow HIPAA and other relevant standards.
- **Access Control:** Implement user authentication and authorization for API access.
- **Audit Logging:** Maintain logs for every prediction request and response for compliance and auditing purposes.

---

8. Testing and Validation

- **Unit Testing:** Validate individual modules (pre-processing, model inference, API endpoints).
- **Integration Testing:** Ensure all components interact correctly.
- **Performance Testing:** Measure inference latency and system throughput.
- **Clinical Validation:** Collaborate with domain experts to evaluate prediction accuracy using real-world data.

---

9. Deployment Considerations

- **Containerization:** Use Docker to package the application and manage dependencies.
- **Orchestration:** Deploy on Kubernetes for scalability and reliability.
- **Continuous Integration/Continuous Deployment (CI/CD):** Set up automated pipelines for testing, deployment, and monitoring.
- **Environment:** Deploy in AWS.

---

10. Maintenance and Monitoring

- **Monitoring:** Use tools such as Prometheus and Grafana to track system performance, latency, and model drift.
- **Model Updates:** Schedule periodic retraining with new data to maintain and improve accuracy.

- **Issue Resolution:** Implement a ticketing system to manage bug reports and system improvements.

---

11. Appendices

11.1 Glossary

- **CNN:** Convolutional Neural Network.
- **API:** Application Programming Interface.
- **MRI:** Magnetic Resonance Imaging.
- **HIPAA:** Health Insurance Portability and Accountability Act.

11.2 References

- VGG16 Model Architecture – [Original Paper/Documentation]
- HIPAA Compliance Guidelines – [Relevant Regulatory Body]