Artificial Intelligence and Machine Learning

Project Documentation format

1. Introduction

• **Project Title:** Deep Learning Fundus Image Analysis for Early Detection of Diabetic Retinopathy

•Diabetic Retinopathy (DR) is a diabetes complication that affects the eyes. Early detection is critical to prevent blindness.

Team Members:

Team ID : LTVIP2026TMIDS83275

Team Member: Meher Divakar – Model Development

Team Member: __________ – Data Collection & Preprocessing

Team Member: __________ – Frontend Developer

Team Member: __________ – Backend Developer & Testing Lead

2. Project Overview

Purpose:

This project uses Deep Learning techniques to detect and classify Diabetic Retinopathy (DR) from retinal fundus images. The goal is to assist ophthalmologists in early diagnosis, reduce manual screening time, and prevent vision loss through early intervention.

Features:

1. Intelligent Image Classification

The system uses Convolutional Neural Networks (CNN) to classify retinal images into:

- No DR
- Mild DR
- Moderate DR
- Severe DR
- Proliferative DR

It converts complex retinal patterns into accurate predictions.

---

2. Robust Image Preprocessing Pipeline

Includes:

- Image resizing (224x224)

- Normalization
- Noise reduction
- Pixel scaling
- Data augmentation (optional)

Ensures high-quality model input and improved accuracy.

---

3. Performance Optimization & Hyperparameter Tuning

Uses:

- Adam Optimizer
- Learning rate tuning
- Dropout layers
- Early stopping
- Batch size optimization

Reduces overfitting and improves generalization.

---

4. Comprehensive Evaluation Metrics

Model evaluated using:

- Accuracy
- Precision
- Recall
- F1-Score
- Confusion Matrix
- ROC-AUC Curve

Ensures reliable and validated model performance.

---

5. Scalable & Modular Architecture

Designed to integrate with:

- Web applications
- Hospital systems
- Cloud platforms
- Mobile health apps

---

6. Medical Decision Support

Provides:

- DR Stage

- Confidence Score
- Risk Level Indicator

Supports clinical decision-making (not replacing doctors).

---

7. Real-World Healthcare Applicability

System can be used in:

- Rural health camps
- Hospitals
- Telemedicine platforms
- Government screening programs

---

8. Extensibility with Future Technologies

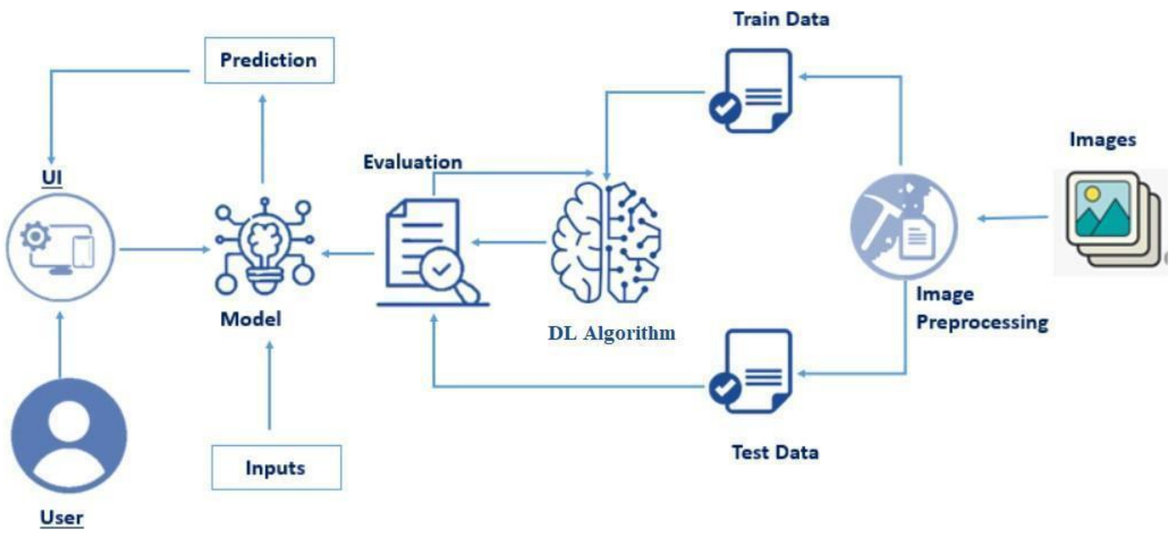
Can be enhanced with:

- Real-time camera scanning
- Multi-disease detection
- Cloud deployment
- AI-based treatment recommendation

---

3. Architecture

Fig.1: Technical Architecture



Overview of Technical Architecture

1. Data Layer

Retinal fundus images stored in dataset folder (Kaggle DR dataset).

---

2. Data Preprocessing Module

- Image resizing
- Normalization
- Data augmentation
- Train/Test split

---

3. Training and Testing Split

- Training Data (80%)
- Testing Data (20%)

Prevents overfitting and ensures proper evaluation.

---

4. Deep Learning Algorithm Layer

CNN Model:

- Convolution Layers
- Pooling Layers
- Dropout
- Fully Connected Layer
- Softmax Output

Model learns retinal lesion patterns.

---

5. Model Evaluation Module

Evaluates:

- Accuracy
- Confusion Matrix
- Classification Report
- Validation Loss

---

6. User Interface (UI) Layer

Web-based interface allows users to:

- Upload fundus image
- View prediction
- See probability score

---

7. Prediction Module

User uploads image → Model processes → DR stage predicted → Result displayed.

Architecture Flow:

Image Dataset → Preprocessing → Train/Test Split → CNN Model → Evaluation → Flask UI → Prediction

---

4. Setup Instructions

Prerequisites:

- VS Code
- Python 3.x

Installation:

Open terminal and install:

```
pip install numpy
pip install pandas
pip install tensorflow
pip install keras
pip install matplotlib
pip install seaborn
pip install scikit-learn
pip install opencv-python
pip install flask
pip install joblib
```

---

5. Folder Structure

Client (Frontend)

Templates Folder:

- index.html – Landing page
- predict.html – Image upload form
- results.html – Prediction results page
- adaptivity.html – Additional information page

Frontend collects image input and displays results.

Server (Backend)

- app.py – Flask application

- model.h5 – Trained CNN model
- requirements.txt
- static/
- templates/

Server Flow:

User Upload → Flask Backend → Image Preprocessing → CNN Model → Prediction → Result Display

---

6. Running the Application

1. Dataset Collection

Dataset downloaded from Kaggle DR dataset.

---

2. Exploratory Data Analysis (EDA)

Univariate Analysis

- Class distribution analysis
- DR severity frequency plot

Multivariate Analysis

- Severity vs Age
- Severity vs Image Features

---

3. Data Preprocessing

- Remove corrupted images
- Normalize pixel values
- Encode labels
- Split dataset

---

4. Model Building

CNN Model trained with:

- Adam Optimizer
- Categorical Crossentropy
- Epochs: 25
- Batch Size: 32

Evaluated using:

- Confusion Matrix

- Classification Report
- ROC Curve

---

5. Server-Side Application

Flask /predict route:

- Accept image upload
- Preprocess image
- Predict DR stage
- Return prediction + probability

---

7. API Documentation

Endpoint	Method	Description
/	GET	Loads main page
/predict	GET	Loads prediction form
/predict	POST	Returns prediction via form
/api/predict	POST	Returns prediction in JSON
/adaptivity	GET	Loads information page

API JSON Example

Request:

```
{
  "image": "base64_encoded_image"
}
```

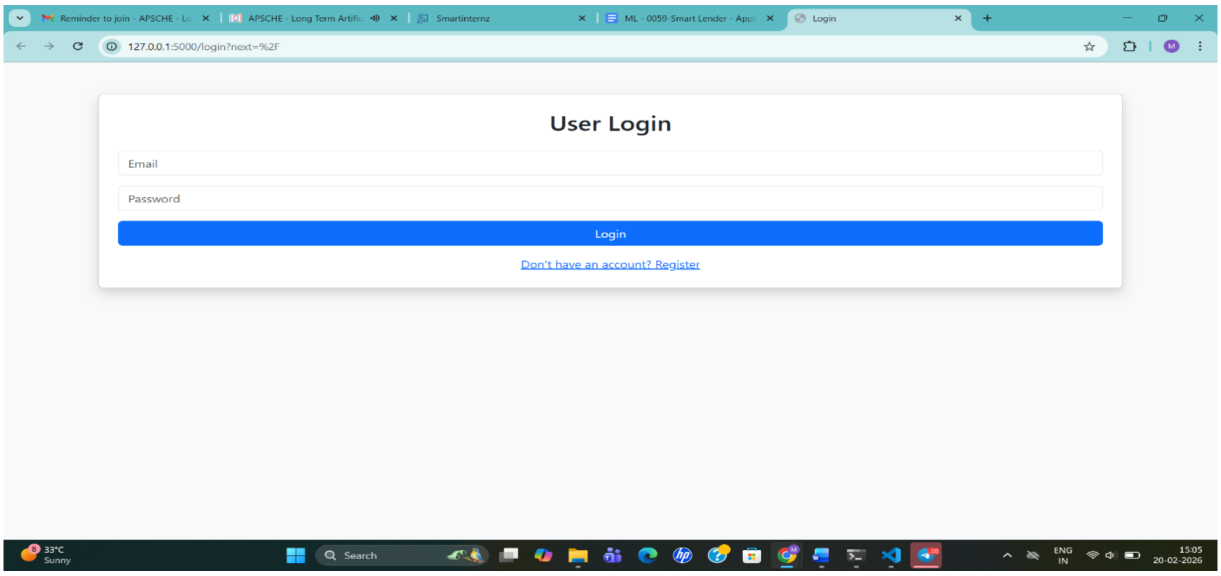
Response:

```
{
  "prediction": "Moderate DR",
  "confidence": 92.4
}
```

---

8. User Interface

Fig.2 – Home Page


Reminder to join - APSCHE - Lo
APSCHE - Long Term Artific
Smartinternz
ML - 0059-Smart Lender - Appl
Login
127.0.0.1:5000/login?next=%2F

User Login

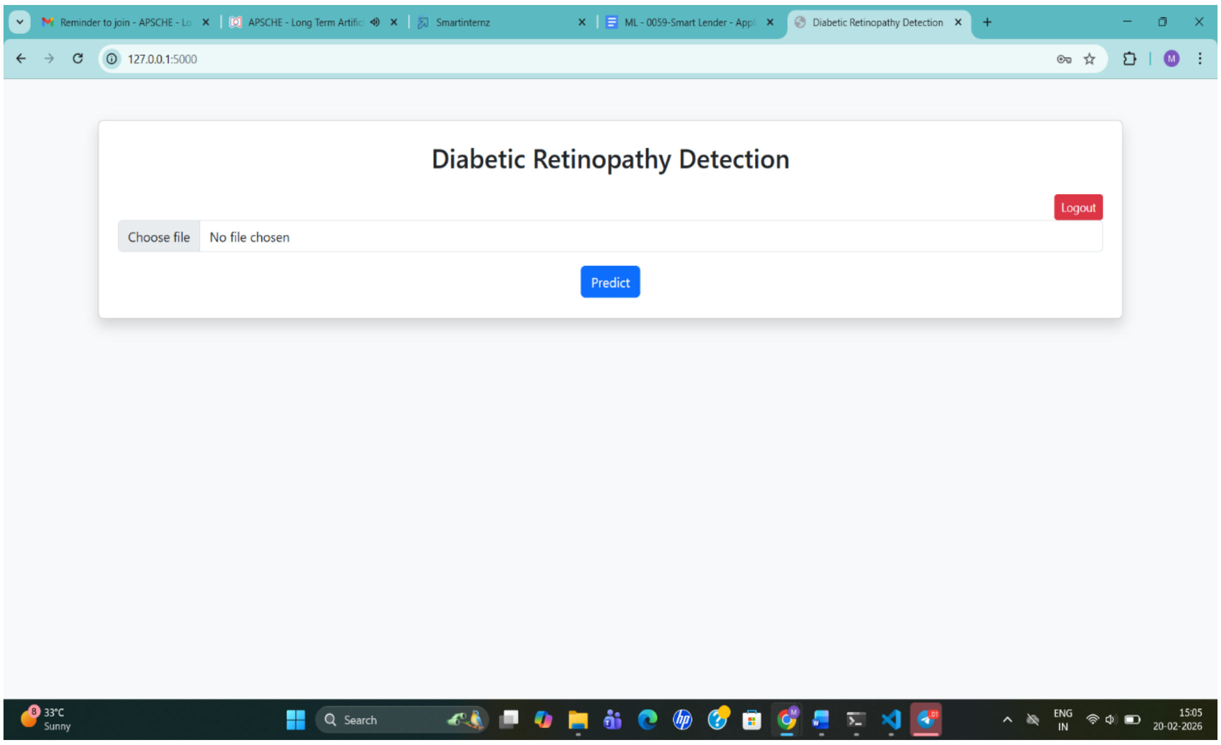
Email

Password

Login

Don't have an account? Register

Fig.3 – Image Upload Page


Reminder to join - APSCHE - Lo
APSCHE - Long Term Artific
Smartinternz
ML - 0059-Smart Lender - Appl
Diabetic Retinopathy Detection
127.0.0.1:5000

Diabetic Retinopathy Detection

Logout

Choose file No file chosen

Predict

9. Testing

Testing Strategy

Functional Testing

- Verified routes
- Image upload
- Prediction output

API Testing

- JSON image input
- Error handling

Model Validation

- Checked class balance
- Verified prediction consistency

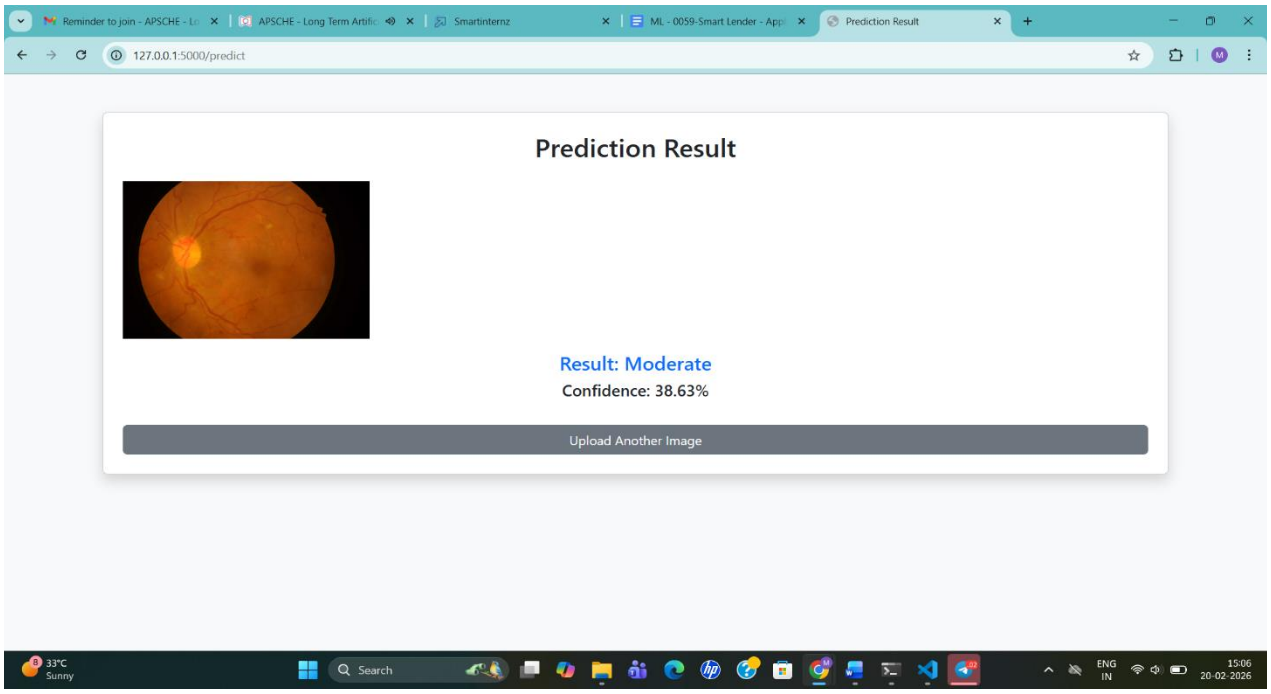
Tools Used

- Manual Browser Testing
- Postman
- Flask Debug Mode
- NumPy & TensorFlow

---

10. Screenshots / Demo

Fig.4 – Results Page



Prediction Result

Result: Moderate

Confidence: 38.63%

---

11. Known Issues

- Limited dataset diversity
- Slight overfitting risk
- Large image upload slows response
- Model interpretability limited (black-box issue)

---

12. Future Enhancements

- Add Grad-CAM visualization
- Deploy on cloud
- Add PDF medical report download
- Integrate doctor dashboard
- Multi-disease detection