# Eye Disease Detection Using Deep Learning

**Project Title:** Eye_Disease_Detection_Using-Deep_Learning

**Author:** Veera Venkata Sesha Sai Mohan

**Date:** July 25, 2025

## 1. Project Overview

### 1.1. Introduction & Vision

This document outlines the plan for the **AI Eye Disease Detection** project. The vision is to create an accessible, end-to-end deep learning solution that can classify major eye diseases from retinal fundus images. The system will not only provide a diagnosis but also enhance trust and usability through model explainability (Grad-CAM) and professional reporting (PDF generation). This tool is intended to serve as a powerful aid for preliminary screening, making ophthalmic diagnostics faster and more accessible

### 1.2. Problem Statement

Millions of people suffer from preventable vision loss due to diseases like cataracts, glaucoma, and diabetic retinopathy. Early detection is critical but is often hindered by a lack of access to ophthalmologists and diagnostic equipment, especially in remote areas. An automated, reliable, and easy-to-use system can serve as a first-line screening tool to identify at-risk individuals, prompting timely medical consultation.

### 1.3. Goals and Objectives

- **Primary Goal:** To develop and deploy a highly accurate deep learning model that classifies retinal images into four categories: Cataract, Diabetic Retinopathy, Glaucoma, and Normal.
- **Objective 1:** Build a robust image classification model using TensorFlow/Keras and the Xception transfer learning architecture.
- **Objective 2:** Develop a user-friendly web interface using Flask that allows users to easily upload an image and receive a diagnosis.
- **Objective 3:** Implement Grad-CAM to provide a visual heatmap, explaining which parts of the image the model focused on for its prediction.
- **Objective 4:** Integrate a feature to auto-generate and download a professional PDF

# Eye Disease Detection Using Deep Learning

## 2. Scope and Features

### 2.1. In-Scope Features

- **Multi-Class Classification:** The model will classify images into four distinct classes.
- **Image Upload & Preview:** Users can select a JPG or PNG file and see a preview before submitting.
- **Prediction Dashboard:** A clean UI will display the predicted disease, a confidence score (as a percentage), the original image, and the Grad-CAM heatmap.
- **PDF Report Generation:** A downloadable PDF report will be generated with all relevant diagnostic information.
- **Flask Web Application:** The entire system will be accessible through a web browser.

### 2.2. Out-of-Scope Features (Future Work)

- User accounts and historical data storage.
- Real-time analysis via a live camera feed.
- Batch processing of multiple images at once.
- Integration with electronic health record (EHR) systems.
- Deployment to a cloud platform (e.g., AWS, Google Cloud).

## 3. Technical Architecture & Stack

This project is divided into three main components: the deep learning model, the backend server, and the frontend interface.

- **Programming Language:** Python 3.12
- **Deep Learning Framework:** TensorFlow / Keras
- **Web Framework:** Flask
- **Image Processing:** OpenCV, Pillow
- **Model Explainability:** tf-keras-vis (for Grad-CAM)
- **PDF Generation:** ReportLab
- **Frontend:** HTML5, CSS3, JavaScript

### 3.1. Deep Learning Model

- **Model Architecture:** Xception (a powerful convolutional neural network).

# Eye Disease Detection Using Deep Learning

- **Technique:** Transfer Learning, using pre-trained weights from the ImageNet dataset.
- **Input Data:** Retinal fundus images, resized to 299x299 pixels.
- **Training:** The model will be trained on a labeled dataset with image augmentation techniques (rotation, zoom, flips) to improve robustness. Early stopping will be used to prevent overfitting.

### 3.2. Backend (Flask Application)

- **app.py:** The main application file will handle routing, business logic, and communication between the model and the frontend.
- **API Endpoint (/predict):** This endpoint will accept an image file via a POST request.
- **Workflow:**
  a. Receive the uploaded image.
  b. Preprocess the image to match the model's input requirements.
  c. Pass the image to the loaded Keras model for prediction.
  d. Generate the Grad-CAM heatmap for the prediction.
  e. Generate the PDF report.
  f. Render a results page displaying all the information.

### 3.3. Frontend (HTML/CSS/JS)

- **index.html:** The main landing page with the file upload form and image preview functionality.
- **result.html:** The page to display the prediction results, including the heatmap and a link to download the PDF report.
- **styles.css:** Custom CSS for a modern, responsive, and user-friendly design.

## 4. Project Plan & Milestones

| Phase | Milestone | Key Activities | Estimated Timeline |
|---|---|---|---|
| **1. Research & Setup** | Project Initialized | - Define project scope & requirements.<br>- Set up Git repository.<br>- Create virtual environment & install dependencies. | 1 Day |
| **2. Model Development** | Trained Model (`.h5`) | - Data collection & preprocessing.<br>- Build and train the Xception model.<br>- | 4 Days |

# Eye Disease Detection Using Deep Learning

| | | | |
|---|---|---|---|
| | | Evaluate model accuracy and performance. | |
| **3. Backend Development** | Functional API | - Set up Flask application structure.<br>- Implement image upload and prediction logic.<br>- Integrate the trained model. | 3 Days |
| **4. Feature Integration** | Core Features Complete | - Implement Grad-CAM generation.<br>- Implement PDF report generation logic. | 2 Days |
| **5. Frontend Development** | UI Complete | - Design and code index.html and result.html.<br>- Style the application with CSS for a polished look. | 2 Days |
| **6. Testing & Deployment** | Project Complete | - End-to-end testing of the user flow.<br>- Bug fixing and refinement.<br>- Write README.md and finalize documentation. | |

## 5. Future Recommendations & Best Practices

- ⚙️ Continuous Model Improvement: Integrate more data and retrain the model regularly to enhance its generalization.
- ☁️ Deployment: Consider deploying the application on cloud services like AWS/GCP with GPU acceleration for faster inference.
- 🔐 Security: Implement image validation and size checks to prevent misuse.
- 📈 Analytics: Add tracking for number of diagnoses performed, user location (anonymized), and model performance metrics.
- 🧪 Testing: Add unit and integration tests for each major component.
- 📚 Version Control: Use GitHub with Git LFS for large files and maintain a clear changelog.

## 6. For questions or suggestions

**Name**: Veera Venkata Sesha Sai Mohan
**GitHub**: @vvssmohan
**Email**: ramanammohan12@gmail.com