DIABETIC RETINOPATHY CLASSIFICATION USING DEEP LEARNING

1.Introduction

Diabetic Retinopathy (DR) is one of the most common complications of diabetes and is a leading cause of blindness among working-age adults worldwide. It occurs due to prolonged high blood sugar levels that damage the blood vessels in the retina. If not detected and treated early, it may result in permanent vision loss. According to global medical reports, millions of patients suffer from undiagnosed diabetic retinopathy due to lack of regular screening and shortage of ophthalmologists.

Traditional diagnosis of Diabetic Retinopathy requires retinal fundus imaging and manual examination by experienced ophthalmologists. This manual screening process is time-consuming, expensive, and prone to human error. In rural or underdeveloped regions, access to specialized eye care professionals is limited, leading to delayed detection.

With the rapid advancements in Artificial Intelligence and Deep Learning, medical image classification has become highly accurate and efficient. Convolutional Neural Networks (CNNs) have shown remarkable performance in detecting patterns in medical images. Transfer learning models like Xception, ResNet, and Inception have significantly improved classification accuracy even with limited datasets.

This project focuses on building a Deep Learning-based Diabetic Retinopathy Classification System using the Xception pre-trained model. The system analyzes retinal fundus images and classifies them into five stages:

1. No Diabetic Retinopathy
2. Mild DR
3. Moderate DR
4. Severe DR
5. Proliferative DR

Additionally, a web application is developed to allow users to upload retinal images and instantly receive classification results along with confidence scores. The application also includes secure user authentication using IBM Cloudant database.

This project bridges the gap between Artificial Intelligence and Healthcare by providing an automated screening system for Diabetic Retinopathy detection.

2. Project Overview

The primary objective of this project is to design and develop an AI-based web application that detects and classifies diabetic retinopathy from retinal images using Deep Learning techniques.

The system consists of two major components:

- Deep Learning Model
- Web Application Interface

The Deep Learning model is built using the Xception architecture, a powerful convolutional neural network pre-trained on ImageNet. The model is fine-tuned for retinal image classification and trained on a labeled dataset containing fundus images categorized into five DR stages.

The web application allows users to:

- Register and create an account
- Login securely
- Upload retinal images
- View prediction results
- See confidence percentage
- Logout securely

The system performs image preprocessing including resizing, normalization, and dimension expansion before feeding it into the trained model. The model outputs probabilities for each class, and the highest probability is selected as the predicted stage.

Key highlights of the project include:

- Use of Transfer Learning
- Secure User Authentication
- Cloudant NoSQL Database Integration
- Medical Dashboard Interface
- Retina Scanning Animation
- Confidence Bar Visualization
- Stage Auto Highlight

The system provides a practical solution for automated retinal disease detection and demonstrates the real-world application of AI in healthcare.

3. Architecture

The architecture of the Diabetic Retinopathy Classification System is divided into three main layers:

1. Frontend Layer
2. Backend Layer
3. Database Layer

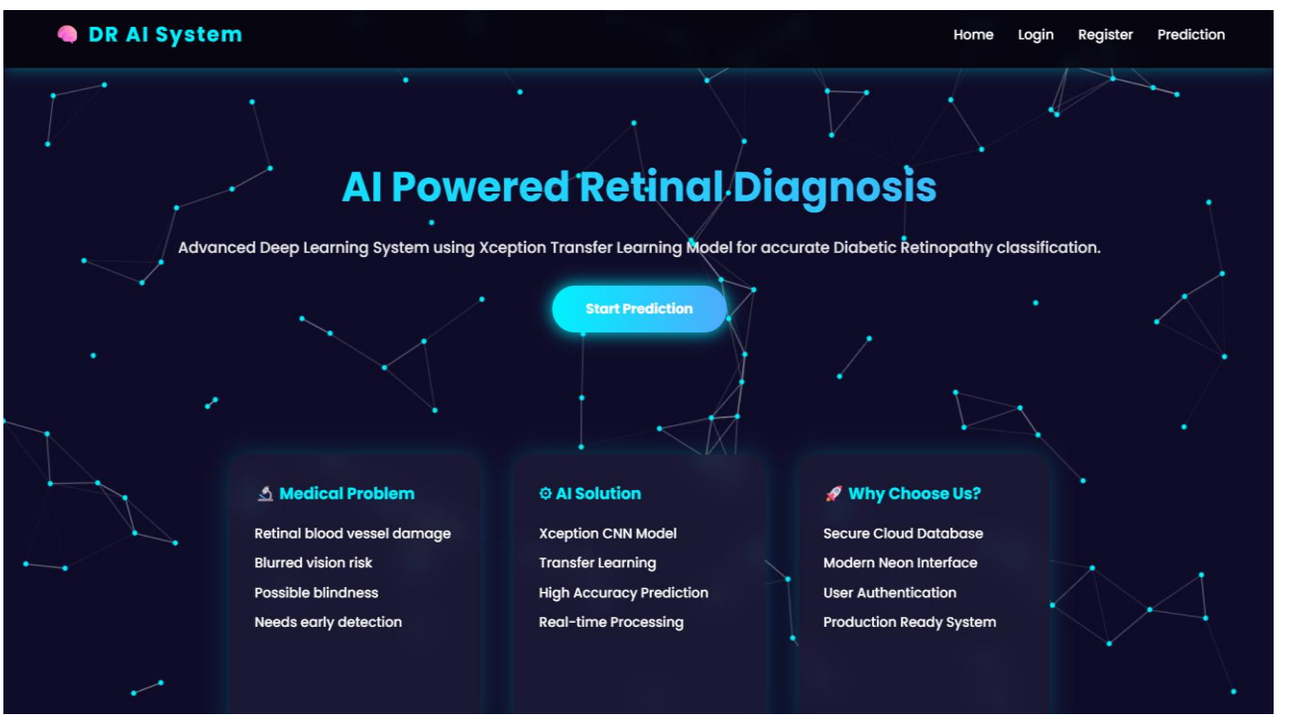
Frontend Architecture

The frontend is designed using HTML, CSS, and JavaScript. It provides a user-friendly and medical-themed interface. The UI includes animated elements such as:

- Moving gradient backgrounds
- Floating particles
- Retina scanning laser animation
- 3D hover effect cards
- Animated loading spinner
- Confidence percentage progress bar

The frontend handles user interactions such as registration, login, and image upload. Once the user uploads an image, it sends a POST request to the backend server for prediction.

The design follows a responsive layout and ensures compatibility across devices.



Backend Architecture

The backend is developed using Flask, a lightweight Python web framework. It performs the following tasks:

- Loads the trained Xception model
- Handles user authentication
- Processes image uploads
- Performs image preprocessing
- Runs model prediction
- Sends results to frontend

The backend performs the following preprocessing steps:

1. Load image using load_img()
2. Resize image to (299, 299)
3. Convert image to array
4. Expand dimensions
5. Normalize using preprocess_input()

The model outputs prediction probabilities. The numpy argmax function is used to determine the final predicted class.

app.py > ...

```python

@app.route("/")
def index():
    return render_template("index.html")


@app.route("/register")
def register():
    return render_template("register.html")


@app.route("/afterreg", methods=["POST"])
def afterreg():
    name = request.form["name"]
    userid = request.form["userid"]
    password = request.form["password"]

    query = {"_id": {"$eq": userid}}
    docs = db.get_query_result(query)

    if len(docs.all()) == 0:
        data = {
            "_id": userid,
            "name": name,
            "password": password
        }
        db.create_document(data)
        return render_template("register.html", pred="Registration Successful! Please Login.")
    else:
        return render_template("register.html", pred="User already exists!")


@app.route("/login")
```

Database Architecture

IBM Cloudant is used as the NoSQL database. It stores user credentials and manages authentication.

Database Schema:

```
{
  "_id": "user_email",
  "name": "User Name",
  "password": "User Password"
}
```

Operations performed:

- Create user document
- Query user during login
- Validate credentials
- Prevent duplicate registration

The database is connected using IAM authentication and secure API keys.

```
# -------- Cloudant DB --------
ACCOUNT_NAME = "b9a4698c-e779-4915-9de2-6ffde75c34cf-bluemix"
USERNAME = "b9a4698c-e779-4915-9de2-6ffde75c34cf-bluemix"
API_KEY = "Tp1KVRqBgKrAReRboeI-mnk4vU9O3sKnTWc8WPQ-hVdN"

client = Cloudant.iam(
    account_name=ACCOUNT_NAME,
    username=USERNAME,
    api_key=API_KEY,
    connect=True
)

db = client.create_database("my_database", throw_on_exists=False)
```

4. Setup Instructions

To successfully deploy and execute the Diabetic Retinopathy Classification System, a proper development environment must be configured. This section provides detailed steps required to install dependencies, configure services, and run the application smoothly.

4.1 Prerequisites

Before setting up the project, ensure the following software and tools are installed:

- Python 3.8 or higher
- pip (Python package manager)
- Virtual Environment (recommended)
- TensorFlow
- Flask
- IBM Cloudant account
- Web browser (Chrome / Edge / Firefox)

For model training :

- GPU-enabled system (NVIDIA CUDA supported)

4.2 Installation Steps

Step 1: Clone the Project Repository

Download or clone the project source code from the repository.

git clone <repository-link>
cd diabetic-retinopathy-project

Step 2: Create and Activate Virtual Environment

It is recommended to use a virtual environment to isolate project dependencies.

python -m venv venv
venv\Scripts\activate (Windows)

Step 3: Install Required Dependencies

Install all required Python packages:

pip install flask
pip install tensorflow
pip install numpy
pip install cloudant
pip install pillow

Alternatively:

pip install -r requirements.txt

Step 4: Add the Trained Model

Place the trained model file:

Updated-Xception-diabetic-retinopathy.h5

inside the root project directory.

Step 5: Configure Cloudant Credentials

In app.py, update the following:

ACCOUNT_NAME = "your_account_name"
USERNAME = "your_username"
API_KEY = "your_api_key"

4.3 Environment Configuration

Ensure:

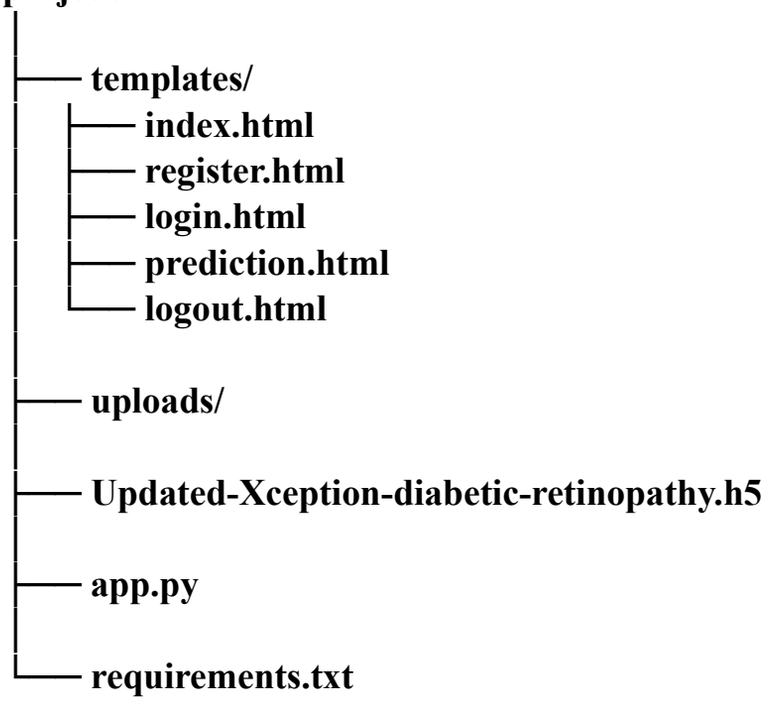
- Internet connection is active (for Cloudant DB)
- Model file path is correct
- Upload folder exists

Once setup is complete, the application can be executed locally.

5. Folder Structure

A well-organized folder structure ensures maintainability, scalability, and modular development. The project follows a clean architecture separating frontend templates, backend logic, and machine learning components.

project/



Explanation of Folders

templates/

Contains all frontend HTML pages rendered using Flask's render_template() function.

- index.html → Landing page
- register.html → User registration
- login.html → User login
- prediction.html → Dashboard & result display
- logout.html → Logout confirmation

uploads/

Stores temporarily uploaded retinal images for prediction processing.

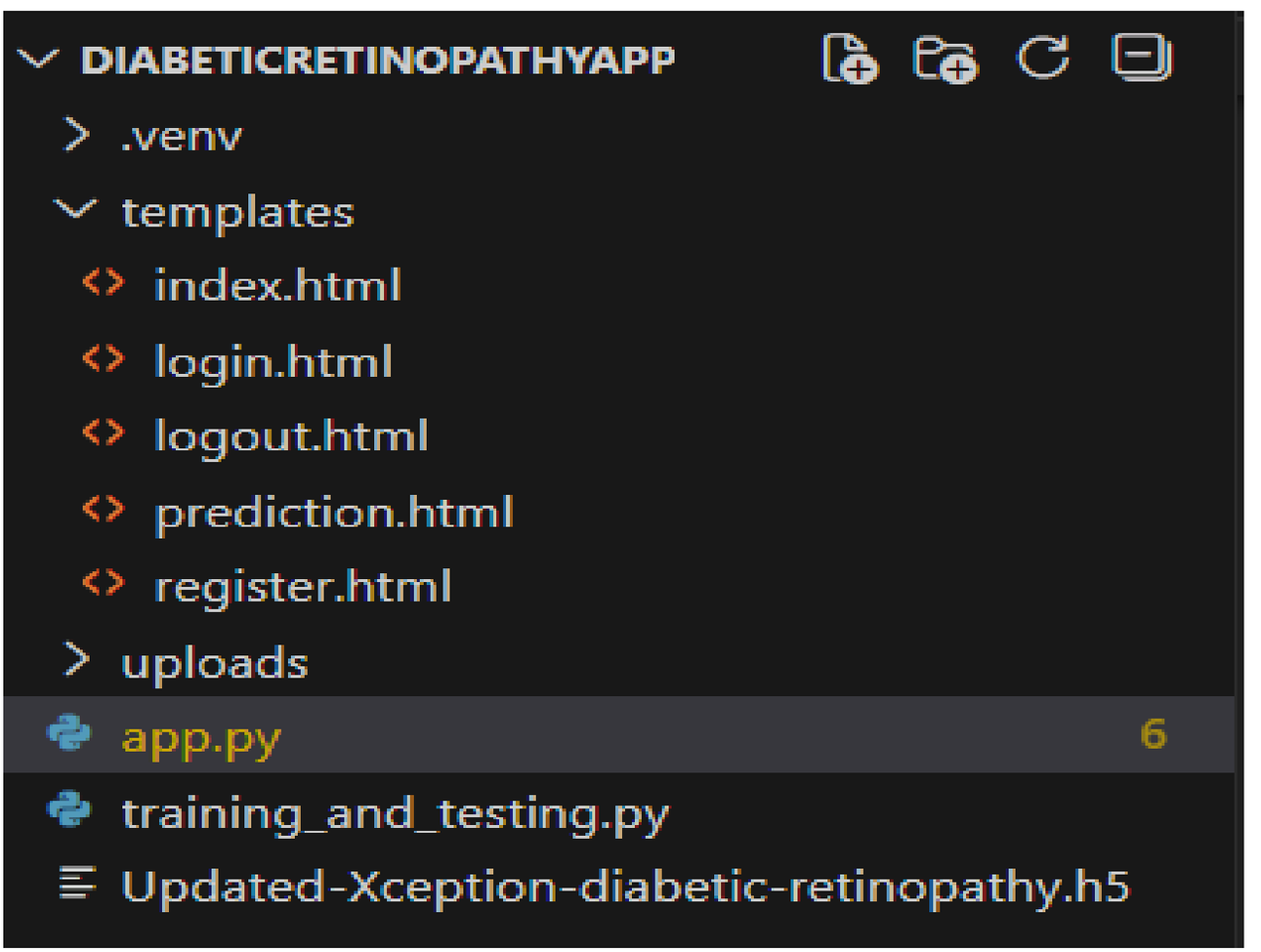
app.py

Main backend file that:

- Loads the trained model
- Connects to Cloudant database
- Handles authentication
- Performs prediction
- Manages sessions

Model File (.h5)

Contains trained Xception CNN weights and architecture.



6. Running the Application

Once setup is complete, the application can be executed locally.

Step 1: Open Terminal

Navigate to the project directory:

cd diabetic-retinopathy-project

Step 2: Activate Virtual Environment

venv\Scripts\activate

Step 3: Run Flask Application

python app.py

Step 4: Access Application

Open browser and enter:

http://127.0.0.1:5000

The application will load the homepage. Users can:

- Register
- Login
- Upload retinal image
- View prediction result

The application runs in debug mode for development. For production deployment, debug mode should be disabled.

7. API Documentation

The backend provides several REST-like routes handled by Flask.

1. Home Route

GET /

Description: Renders the homepage.

2. Register Route

GET /register
POST /afterreg

Parameters:

- name
- userid
- password

Response:

- Success message
- User already exists message

3. Login Route

GET /login
POST /afterlogin

Parameters:

- userid
- password

Response:

- Redirect to prediction page
- Invalid credentials message

4. Prediction Route

GET /prediction
POST /prediction

Parameters:

- image (file upload)

Response:

- Predicted class
- Confidence percentage
- Highlighted stage in UI

5. Logout Route

GET /logout

Description:

- Clears user session
- Redirects to logout page

8. Authentication

The system implements session-based authentication using Flask.

Registration Process

- User data stored in Cloudant DB
- Duplicate user prevention implemented

Login Process

- User credentials validated from database
- Session created:

session["user"] = userid

Access Control

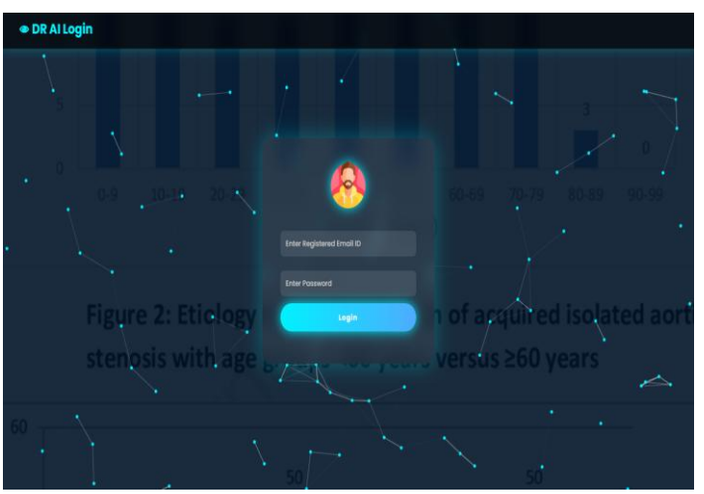
- Prediction page accessible only if logged in
- Unauthorized users redirected to login

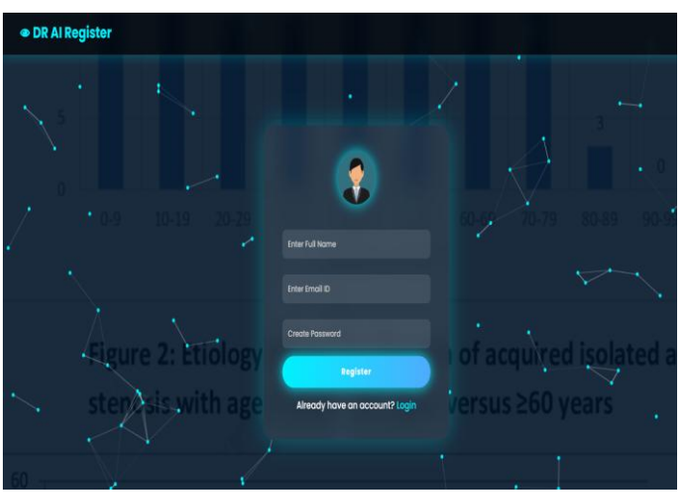
Logout

- Session cleared using:

session.clear()

This ensures secure access and prevents unauthorized prediction access.





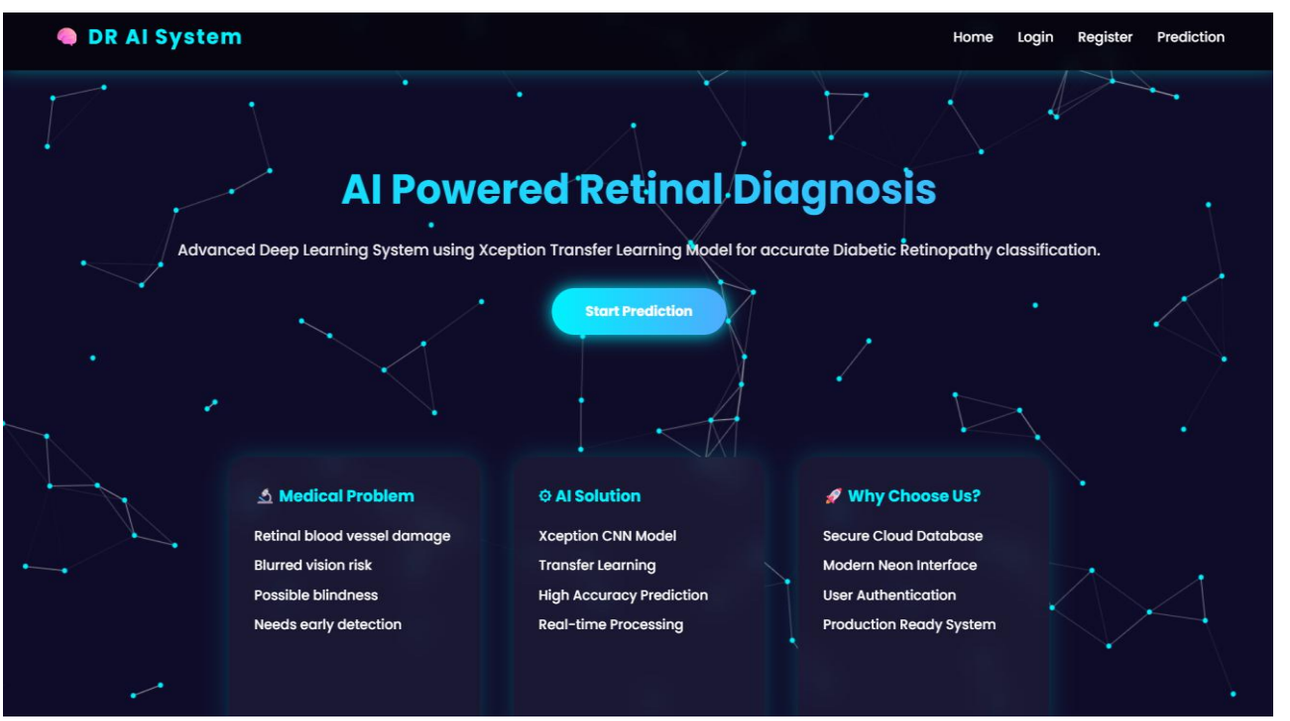
9. User Interface

The user interface is designed to provide a professional medical dashboard experience.

UI Highlights

- Retina-themed background
- Neon medical theme
- Animated particles
- Retina scanning laser effect
- 3D hover stage cards
- Confidence progress bar
- Responsive layout

The interface improves user engagement and enhances usability for both medical professionals and general users.



10. Testing

The system was tested in multiple stages:

Functional Testing

- Registration and login validation
- Duplicate user handling
- Session management
- Image upload validation

Model Testing

- Training accuracy monitoring
- Validation accuracy evaluation
- Overfitting analysis

Performance Testing

- Response time measurement
- Prediction time evaluation

Edge Case Testing

- Invalid file uploads
- Empty input fields
- Incorrect credentials

The system performed reliably under normal conditions.

11. Known Issues

- Model training requires GPU for faster computation
- Performance depends on retinal image quality
- Local deployment only (not yet cloud-hosted)
- Cloudant DB requires internet connectivity
- No password encryption implemented (basic storage)

Future versions may address these limitations.

12. Future Enhancements

Several improvements can enhance the system:

- Deploy on AWS / Azure cloud
- Add password hashing (bcrypt)
- Implement Grad-CAM heatmaps
- Add patient history tracking
- Real-time camera retina scanning
- Multi-disease detection (Glaucoma, Cataract)
- Doctor dashboard analytics
- Mobile application version
- PDF medical report generation

13. Conclusion

The Diabetic Retinopathy Classification System successfully demonstrates the integration of Deep Learning with web technologies for medical image classification.

Using the Xception CNN model, the system accurately classifies retinal images into five severity levels. The integration with Flask and Cloudant provides a secure and interactive web application for users.

This project highlights the real-world application of AI in healthcare and presents a scalable solution for automated retinal disease detection. With further improvements and cloud deployment, the system can be adapted for hospital-level usage.