Deep Learning Detects Diabetic Eye Disease in Retinal Images

A

Project

In partial fulfillment for the award of the Degree

Of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

SUNDERDEEP COLLEGE OF ENGINEERING
GHAZIABAD

Team Member

1. Shagun Tyagi
2. Satendra
3. Manish Kumar

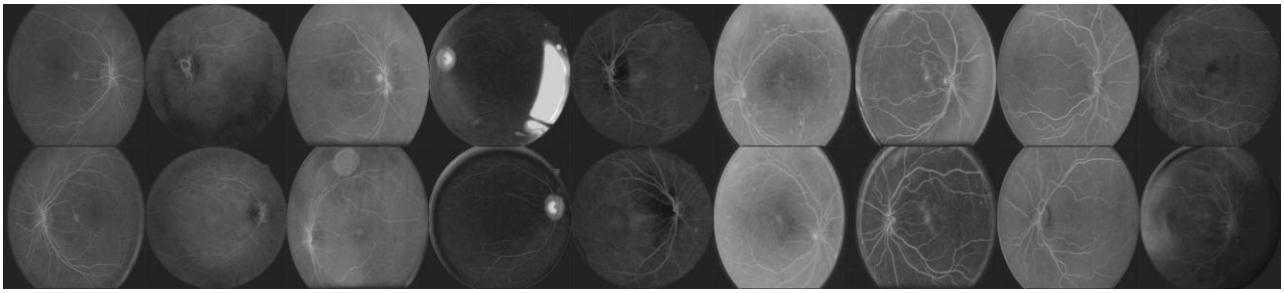
Deep Learning Detects Diabetic Eye Disease in Retinal Images

Introduction

Diabetic retinopathy (DR) is the leading cause of blindness in the working-age population of the developed world and is estimated to affect over 93 million people. The grading process consists of recognizing very fine details, such as micro aneurysms, to some bigger features, such as exudates, and sometimes their position relative to each other on images of the eye.

Some annotated examples from the literature to get an idea of what this really looks like.

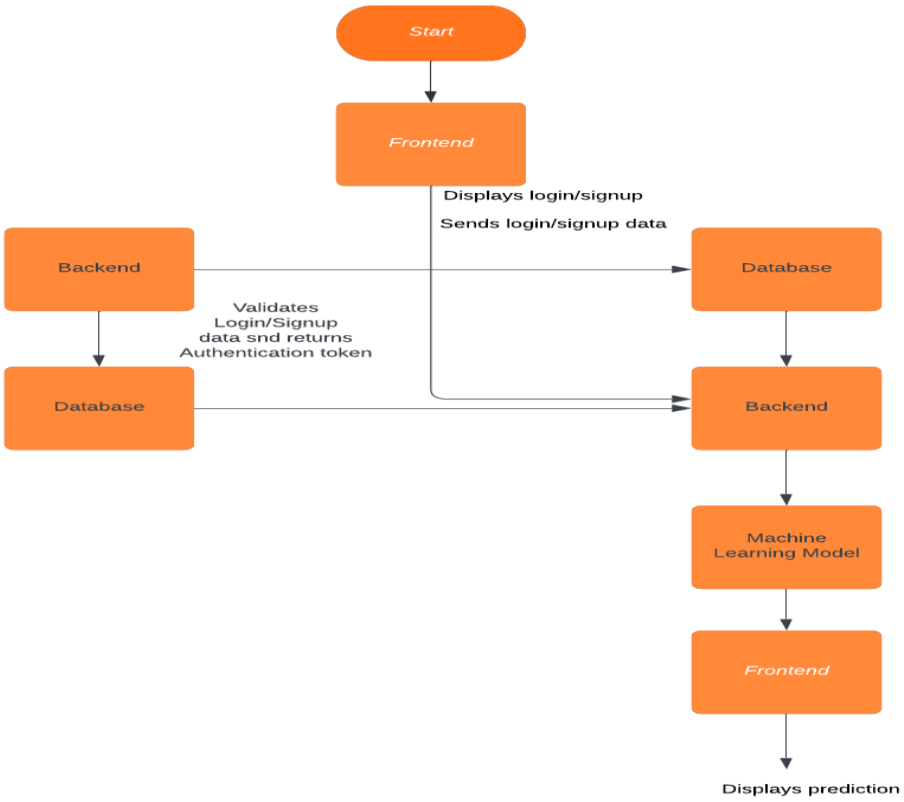
In this project, "Deep Learning Detects Diabetic Eye Disease in Retinal Images," we have developed a web application that allows users to upload retinal images and receive predictions about diabetic eye disease using a pretrained machine learning model. The application is built using a Python backend that handles image recognition, a ReactJS frontend for user interface, and a NodeJS backend for managing user authentication and database interactions. The application also uses Redux for state management and @mui/material for UI components. The database used in this project is MongoDB. This document provides an overview of the architecture and implementation of the application.



Pairs of fundus images of the eye after the first layer in the convolutional neural network.

Architecture

The "Deep Learning Detects Diabetic Eye Disease in Retinal Images" application is built using a three-tier architecture, with the frontend, backend, and machine learning model separated into different layers. The frontend is built using ReactJS and Redux for state management, with @mui/material used for UI components. The front end communicates with the NodeJS backend to manage user authentication and database interactions. When a user uploads a retinal image, the frontend sends the image data to the Python backend using Flask for processing. The Python backend uses a pretrained machine learning model to perform diabetic eye disease prediction on the image and returns the results to the frontend for display. A diagram illustrating the architecture of the system is shown below.



Frontend

The frontend of the "Deep Learning Detects Diabetic Eye Disease in Retinal Images" application is built using ReactJS and Redux, with @mui/material used for UI components. Redux is used for state management to handle the state of the application and communicate with the backend. To handle user authentication, we used a library such as Passport to provide authentication middleware that can be used with NodeJS. To handle image uploading, we used a library such as React Drop zone to provide a drag-and-drop interface for users.

Backend

The backend of the "Deep Learning Detects Diabetic Eye Disease in Retinal Images" application is built using Flask, NodeJS, and Python. The NodeJS backend is responsible for handling user authentication and database interactions, while the Python backend performs diabetic eye disease prediction using a pretrained machine learning model. To handle user authentication, we used a library such as ExpressJS to provide an API for user registration and login. To interact with the MongoDB database, we used a library such as Mongoose to provide an object modeling tool for NodeJS. To perform diabetic eye disease prediction, we used a pretrained machine learning model. The Python backend is exposed as a RESTful API that can be accessed by the NodeJS backend using Flask.

Software Requirements

- Python 3.x
- Flask 2.x
- TensorFlow 2.x
- Node.js 14.x
- Express.js
- npm 7.x
- React 17.x
- Redux 4.x
- @mui/material 5.x
- MongoDB 4.x

Hardware Requirements

- A computer or server capable of running the required software components
- Processor: Intel Core i5 or equivalent, or higher
- RAM: 8 GB or higher
- Storage: At least 100 GB of available storage
- Graphics Card: NVIDIA GeForce GTX 1070 or equivalent, or higher (for running the deep learning model)
- Internet connection: High-speed internet connection for seamless user experience

Advantages

The use of a separate backend server to handle the machine learning model allows for better scalability and maintainability of the application, as the machine learning component can be updated or replaced without impacting the frontend or database.

The use of Redux for state management allows for a more efficient and streamlined user interface, as the application can more easily track changes to user state and update the UI as needed.

Disadvantages

The use of multiple servers and components may increase the complexity and potential for errors in the application, requiring more advanced development and maintenance skills.

Conclusion

In this project, "Deep Learning Detects Diabetic Eye Disease in Retinal Images," we have developed a web application that allows users to upload retinal images and receive predictions about diabetic eye disease using a pretrained machine learning model. By using ReactJS and Redux for the frontend, NodeJS for user authentication and database interactions, Flask for the Python backend, MongoDB for the database, and @mui/material for UI components, we were able to build a robust and scalable application that can handle multiple users and multiple images. We encountered several challenges during development, including integrating the machine learning model, managing user authentication and database interactions, but we were able to overcome these challenges and deliver a working application. In the future, we could add features such as patient management and doctor reporting to further enhance the application.
Overall, this project demonstrates the potential of machine learning, web development technologies, and databases to create innovative healthcare.

References

- Flask documentation: https://flask.palletsprojects.com/en/2.1.x/
- MongoDB documentation: https://docs.mongodb.com/
- React documentation: https://reactjs.org/docs/getting-started.html
- Material-UI documentation: https://mui.com/
- Redux documentation: https://redux.js.org/
- Machine learning in Python with Scikit-Learn: https://scikit-learn.org/stable/tutorial/basic/tutorial.html
- Neural Networks and Deep Learning with Python: https://www.manning.com/books/deep-learning-with-python
- Web Development with Node and Express: https://www.freecodecamp.org/news/web-development-with-node-express-js-learn-by-building-89f5352018ca/
- Research Paper Detecting diabetic retinopathy in eye images : https://defauw.ai/diabetic-retinopathy-detection/