Artificial Intelligence & Machine Learning

Project Documentation format

1. Introduction

• **Project Title:** HematoVision – Advanced Blood Cell Classification Using Transfer Learning
• **Team Members:**
BAITAPALLI KISHOR-Model Development, Flask Integration, Performance Testing, Deployment
Kenguva Pavan-Dataset Collection, Data Preprocessing, UI Design, Testing & Documentation Support

2. Project Overview

• **Purpose:** The purpose of HematoVision is to automate blood cell classification using deep learning. The system reduces manual effort in microscopic blood analysis and improves diagnostic efficiency**.**
•**Features:**

- Image upload functionality
- Automated blood cell classification
- Prediction confidence display
- Invalid image detection
- Web-based interface using Flask
- MobileNetV2 Transfer Learning Model

3. Architecture

Frontend:

- HTML
- CSS
- Bootstrap
- Jinja2 Templates

Backend:

- Python
- Flask Framework
- REST-based routing
- Model integration using TensorFlow/Keras

Database:

- No external database used
- Image dataset stored in local directory
- Model stored as .h5 file

4. Setup Instructions

Prerequisites:

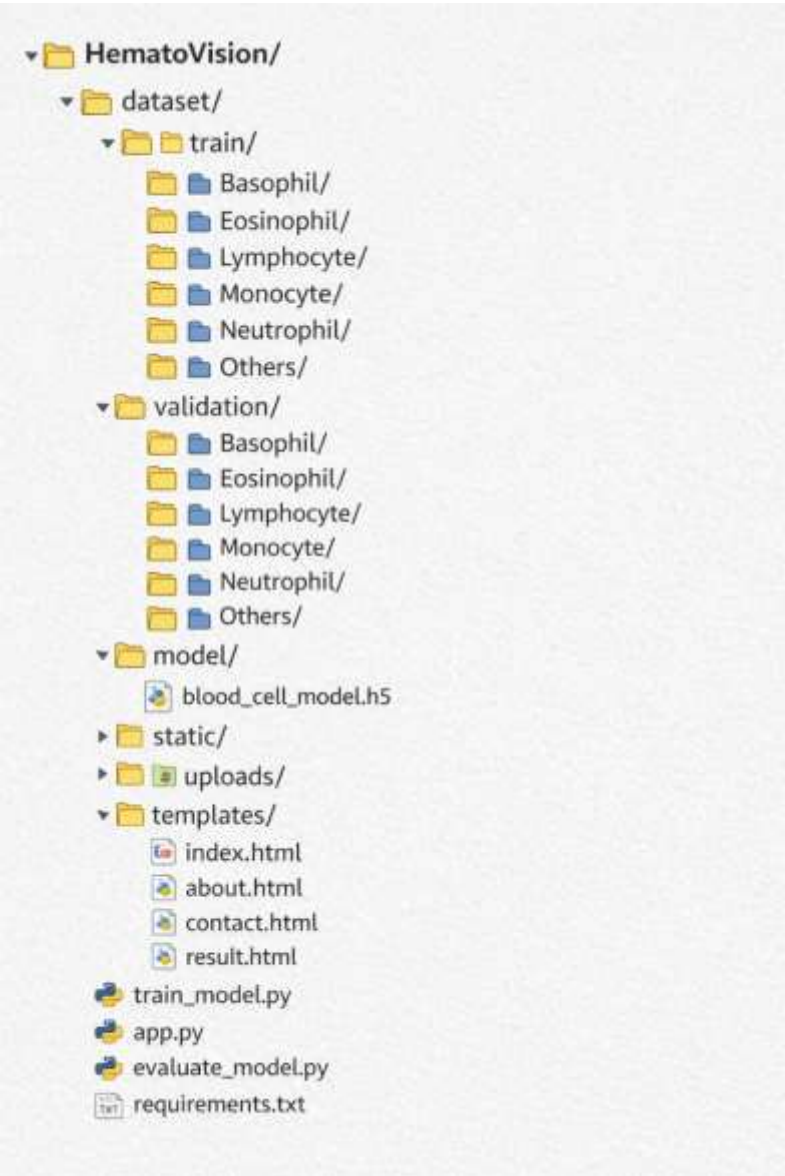
- Python 3.10+
- TensorFlow
- Flask
- scikit-learn
- matplotlib

Installation:

1. git clone https://github.com/Baitapallikishor/HematoVision.git
2. cd HematoVision
3. pip install -r requirements.txt
4. python train_model.py
5. python app.py

5. Folder Structure

The HematoVision project follows a Python Flask-based architecture. The folder structure is organized as follows:



Description:

- **dataset/** – Contains training and validation image data.
- **model/** – Stores the trained deep learning model file (.h5).
- **static/** – Stores uploaded images and static assets.
- **templates/** – Contains HTML pages for the web interface.
- **train_model.py** – Script used for training the MobileNetV2 model.
- **app.py** – Flask backend application for prediction and routing.
- **requirements.txt** – Lists required Python dependencies.

6. Running the Application

Step 1: Train the Model

```
python train_model.py
```

Step 2: Run the Flask Application

```
python app.py
```

Step 3: Open in Browser

http://127.0.0.1:5000

7. API Documentation

POST /predict

Description:
Accepts image file and returns prediction result.

Request:
Form-data image upload

Response:
Predicted class and confidence percentage.

8. Authentication

• No authentication system implemented.
• Application is publicly accessible for testing.

9. User Interface

Include screenshots of:

- Home Page
- Upload Section
- Prediction Result Page
- Confusion Matrix

10. Testing

- Functional Testing
- Performance Testing
- Confusion Matrix Analysis
- Classification Report

11. Screenshots or Demo

Include:

- Accuracy Graph
- Confusion Matrix
- Classification Report
- Flask Prediction Output

12. Known Issues

- Minor misclassification between Monocyte and Eosinophil
- Performance depends on system hardware

13. Future Enhancements

- Add database for storing prediction history
- Improve accuracy using more dataset
- Deploy on cloud server
- Add user authentication
- Add support for additional blood disorders