**Artificial Intelligence and MachineLearning**

**Project Documentation**

**HematoVision:** Advanced Blood Cell Classification Using Transfer Learning

Project Documentation

**1. Introduction**

Project Title: HematoVision: Advanced Blood Cell Classification Using Transfer Learning

Team Members:

**2. Project Overview**

Purpose: HematoVision aims to develop an accurate and efficient model for classifying blood cells by employing transfer learning techniques. It provides a reliable and scalable tool for pathologists and healthcare professionals, ensuring precise and efficient blood cell classification.

Features:

Classification of blood cells (eosinophils, lymphocytes, monocytes, and neutrophils) using transfer learning.

Leverages pre-trained Convolutional Neural Networks (CNNs) for expedited training and improved accuracy.

Integration into automated diagnostic systems for real-time blood analysis and report generation.

Application in telemedicine platforms for remote consultations and diagnostics.

Incorporation into educational tools for medical training, offering interactive learning and instant feedback.

**3. Architecture**

Frontend: The frontend is built using HTML pages (home.html and result.html), which provide the User Interface (UI) for image selection and displaying prediction results.

Backend: The backend is developed using Flask, a Python web framework. It handles the logic for receiving image uploads, processing them through the trained model, and returning predictions to the frontend.

Database: No explicit database is mentioned or used in the provided project description. The project focuses on model integration and serving predictions via a web application.

**4. Setup Instructions**

Prerequisites:

Anaconda Navigator (refer to the provided link for download).

Prior knowledge of Deep Learning Concepts (Neural Networks, Deep Learning Frameworks, Transfer Learning, VGG16, Convolutional Neural Networks (CNNs), Overfitting and Regularization, Optimizers).

Prior knowledge of Flask Basics.

Installation:

Open Anaconda Prompt as administrator.

Type pip install numpy and press Enter.

Type pip install pandas and press Enter.

Type pip install scikit-learn and press Enter.

Type pip install matplotlib and press Enter.

Type pip install scipy and press Enter.

Type pip install seaborn and press Enter.

Type pip install tensorflow and press Enter.

Type pip install Flask and press Enter.

**5. Folder Structure**

The project folder contains the following structure:

Project Folder/

├── app.py

├── Blood Cell.h5

└── templates/

├── home.html

└── result.html

app.py: The main Python script for the Flask application.

Blood Cell.h5: The saved pre-trained blood cell classification model.

templates/: A directory containing the HTML template files for the user interface.

6. Running the Application

To run the web application locally:

Open Anaconda Prompt from the start menu.

Navigate to the project folder where your Python script (app.py) is located.

Type the command python app.py and press Enter.

Open your web browser and navigate to the localhost URL: http://127.0.0.1:5000.

**7. API Documentation**

The Flask application exposes the following endpoints:

Endpoint: /

Method: GET

Description: Renders the home.html page, which is the main UI for uploading images.

Parameters: None

Example Response: Renders the HTML content of home.html.

Endpoint: /output

Method: POST

Description: Receives the uploaded image file, processes it using the loaded Blood Cell.h5 model, and returns the prediction result.

Parameters:

image\_file: The blood cell image uploaded by the user (via HTML form input).

Example Response: Renders the result.html page, displaying the predicted blood cell class (e.g., "Neutrophil", "Monocyte", "Lymphocyte", "Eosinophil").

**8. Authentication**

The provided project description does not include details about authentication or authorization mechanisms. The application appears to be designed for direct access without explicit user login or role-based access control.

**9. User Interface**

The application provides a simple web-based user interface:

Home Page (home.html):

Allows users to choose an image file to upload.

Features a "Predict" button to initiate the classification process.

(As described in the prompt) "By clicking on choose file it will ask us to upload the image, then by clicking on the predict button, it will take us to the result.html"

Result Page (result.html):

Displays the predicted classification of the uploaded blood cell image.

Examples:

Test For Class-1: Neutrophil

Test For Class-2: Monocyte

Test For Class-3: Lymphocyte

Test For Class-4: Eosinophil

**10. Testing**

The model was tested using the predict() function of the MobileNetV2 model. The evaluation involved predicting the class of various blood cell images (Neutrophil, Monocyte, Lymphocyte, Eosinophil) and showcasing the results on the UI. The model was trained for 5 epochs with callbacks like Model Checkpoint and Early Stopping to ensure optimal performance.

**11. Screenshots or Demo**

https://drive.google.com/file/d/1fm-U\_yhKdk9TwT8hzplfy4dX-T7ysyFM/view?usp=drivesdk

**12. Known Issues**

There are no known bugs or issues explicitly documented in the provided project description.

**13. Future Enhancements**

The provided project description does not explicitly outline potential future features or improvements. However, based on the project's scope, potential enhancements could include:

Implementing a user authentication system.

Adding a history of predictions for users.

Integrating with a database to store prediction logs and user data.

Improving the UI/UX for a more intuitive experience.

Expanding the model to classify more types of blood cells or detect anomalies.

Deploying the application to a cloud platform for wider accessibility.