Project Documentation format

1. Introduction

Project Title: Hematovision: Advanced Blood Cell Classification using Transfer Learning

Team Members: Team Leader : Derangula Jahnavi
Team member : Balijabudda Prasad
Team member : Busireddy Swathi
Team member : Galam Tharunya

2. Project Overview

Purpose:

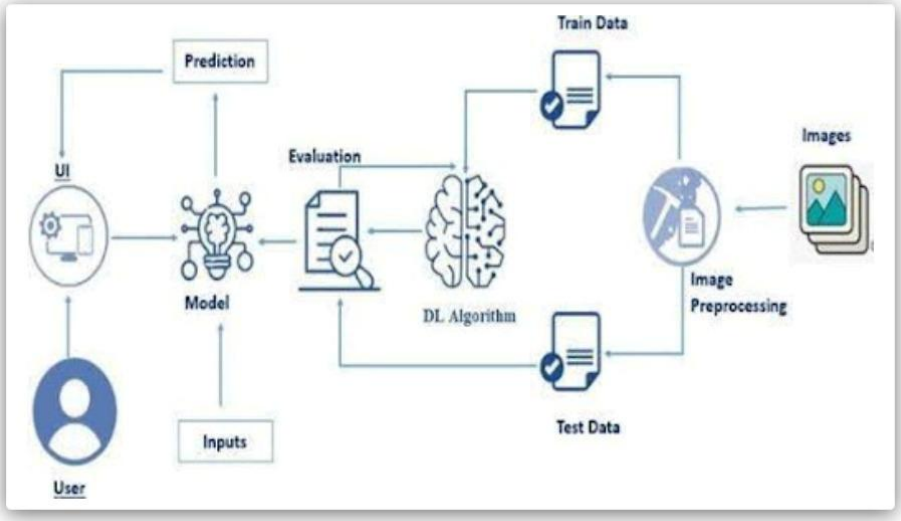
The purpose of Hematovision is to develop an intelligent, automated system that leverages transfer learning to accurately classify blood cells from microscopic images. This project aims to reduce the time, effort, and subjectivity involved in manual blood smear analysis, thereby assisting medical professionals in making faster and more reliable diagnoses of hematological disorders. By using pre-trained deep learning models, Hematovision ensures high accuracy even with limited medical datasets, making it an effective solution for both well-equipped labs and resource-constrained healthcare settings.

Features:

Hematovision offers a range of powerful features designed to streamline and enhance the process of blood cell classification. At its core, it uses transfer learning with pre-trained deep learning models like ResNet or VGG to accurately classify various types of blood cells, including RBCs, WBCs, and platelets. The system ensures high accuracy, speed, and consistency, reducing human error in manual analysis

3. Architecture:

Architecture:



4. Setup Instructions

Prerequisites:

Deep Learning Frameworks: TensorFlow / Keras or PyTorch

Image handling: OpenCV, PIL

Data handling: NumPy, Pandas

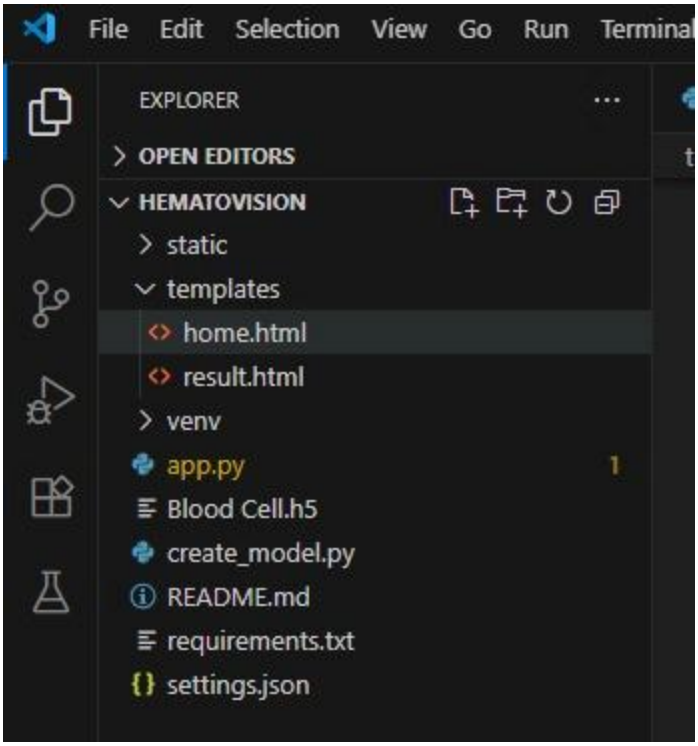
Visualization: Matplotlib,Seaborn

Installation:

1. Install Python
   Ensure that Python (version 3.8 or higher) is installed on your computer. You can download it from the official Python website.
2. Set Up a Virtual Environment (Optional)
   Creating a virtual environment is recommended to keep the project dependencies isolated from other projects.
3. Install Required Libraries
   Install all necessary Python libraries, including TensorFlow or PyTorch, Keras, OpenCV, NumPy, Pandas, Matplotlib, Seaborn, and Scikit-learn.
4. Download the Project Files
   Obtain the Hematovision project files by downloading them from a repository or shared location.
5. Prepare the Dataset
   Download a blood cell image dataset (such as the BCCD dataset). Organize the dataset into folders based on cell types (e.g., RBC, WBC, Platelets) and place it in the appropriate data directory within the project.
6. Train the Model
   Use the provided scripts or notebook to preprocess the data and train the model using transfer learning techniques on the labeled dataset.
7. Run the Classifier
   After training, use the classification tool or interface provided to test the system on new images.
8. (Optional) Launch the Web Interface
   If the project includes a web-based user interface (e.g., built with Flask), you can launch it to interact with the model through a browser.

.

5. Folder Structure



6. Running the Application:

Backend (Server-side):

The backend handles the core logic, including:

- Loading the trained machine learning model
- Processing uploaded images
- Running inference (prediction) to classify blood cells
- Sending results to the frontend

Frontend (User Interface)

The frontend is the interface that allows users to:

- Upload blood smear images
- View classification results (e.g., cell type, count, prediction confidence)
- Interact with the application through buttons or forms

7. API Documentation

POST /predict:

This endpoint receives an uploaded blood smear image and returns the predicted blood cell classification.

Sample Request Using HTML Form:

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
<meta charset="UTF-8">
<title>Hematovision - Blood Cell Classifier</title>
</head>
<body>
<h2>Upload Blood Smear Image</h2>
<form action="/predict" method="POST" enctype="multipart/form-data">
<label>Select Image:</label>
<input type="file" name="image" accept="image/*" required><br><br>
<input type="submit" value="Classify">
 </form>
  </body>
    </html>
```

Example API Response:

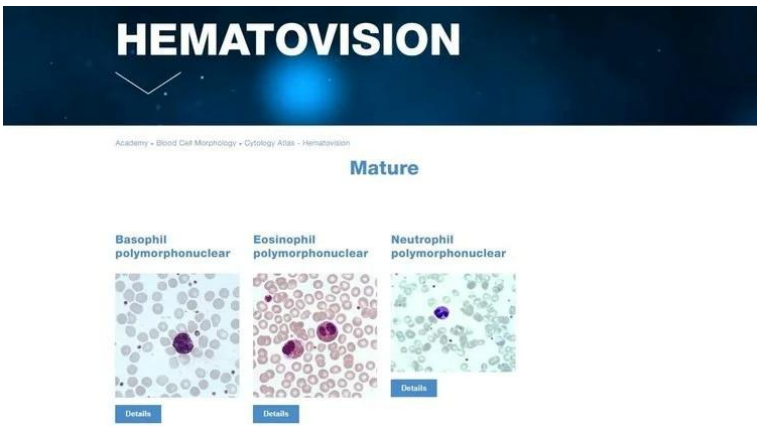
```
  {
 "status": "success",
 "prediction": {
  "cell_type": "Neutrophil",
  "confidence": 0.9654
 },
 "message": "Prediction completed successfully."
}
```

.

8. Authentication:

To ensure secure and restricted access to the Hematovision application, an authentication system is implemented. This helps protect sensitive medical data and ensures that only authorized users—such as doctors, lab technicians, or researchers—can access the classification services. The system may use token-based authentication, where users log in and receive a secure token that must be included in each API request. Alternatively, session-based or basic authentication can be used for simpler interfaces. This layer of security prevents unauthorized access and supports safe usage of the application in both clinical and research environments.

User Interface:

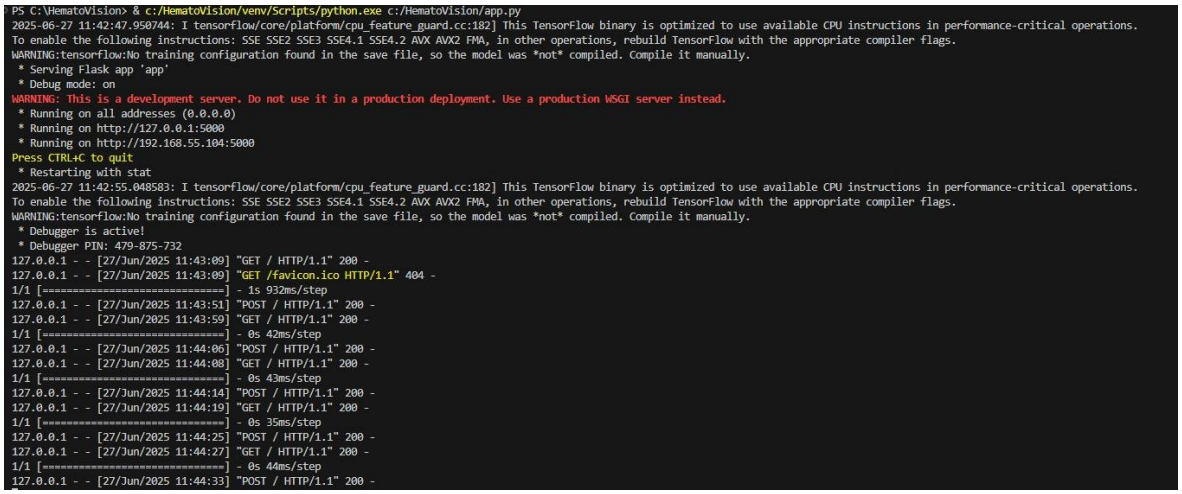


10. Testing :

Testing is a crucial phase in the Hematovision project to ensure the accuracy, reliability, and performance of the blood cell classification system. The model is tested using a separate set of labeled blood smear images that were not used during training, allowing for an unbiased evaluation. Key performance metrics such as accuracy, precision, recall, F1-score, and confusion matrix are used to assess how well the model identifies various blood cell types.

11.Screenshots or Demo:

The complete execution of the Hematovision application is shown below through a series of step-by-step screenshots:



12.Known Issues:

- Image Quality Dependency: Performance drops with low-resolution, blurry, or poorly stained images.
- Limited Dataset Diversity: May not accurately classify rare or uncommon blood cell types due to limited training data.
- Overlapping Cells: Difficulty in handling overlapping or clustered cells in microscopic images.
- Hardware Variation: Model performance may vary across labs due to differences in microscopes and imaging settings.
- No Batch Processing: Currently supports only single-image classification at a time.
- Lack of Abnormality Detection: Does not yet identify abnormal cells such as blast cells or cancerous forms.
- Generalization Limitations: Model may need retraining or fine-tuning when applied to new datasets or environments.
- Limited Clinical Validation: The system requires further testing and validation in real clinical settings for full reliability.