Hematovision:

Advanced Blood Cell Classification using Transfer Learning

Project Documentation format

1. Introduction

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

- **Team Members:**
    1. Derangula Jahnavi[22HM1A0528]
    2. Balijabudda Prasad [22HM1A0507]
    3. Busireddy Swathi [22HM1A0520]
    4. Galam Tharunya [22HM1A0535]

. 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.

- **Goals:**

Specific Goals:

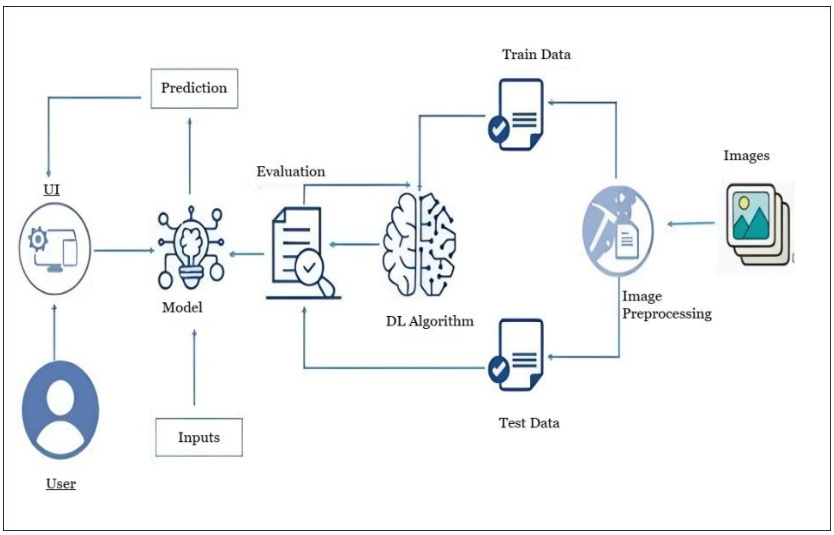
- To build a deep learning model using **transfer learning (MobileNetV2)** for accurate blood cell classification.
- To develop a **Flask-based web application** that allows users to upload an image and instantly receive the predicted cell type.
- To reduce the dependency on manual microscopic examination, thereby saving time and minimizing human error.
- To design a system that works efficiently on local machines **without the need for GPU or cloud resources**.

- To offer an **easy-to-use interface** for medical labs, students, or researchers working with blood cell data.
- To maintain **high accuracy and fast performance**, even on basic hardware setups.

- **Features:**

  - Classifies blood cell images into Neutrophil, Eosinophil, Monocyte, and Lymphocyte.
  - Uses MobileNetV2 transfer learning model for high accuracy.
  - Web interface built with Flask for image upload and prediction.
  - Provides instant prediction results with image preview.
  - Works offline without internet or GPU.
  - Lightweight and fast; suitable for low-end systems.
  - Saves the trained model as `.h5` for reuse.
  - Simple to set up and install using Python and pip.
  - Easily extendable for additional features or classes.

3. Architecture



4. Setup Instructions

- **Prerequisites**:

To complete this project, you must require the following software, concepts, and package.

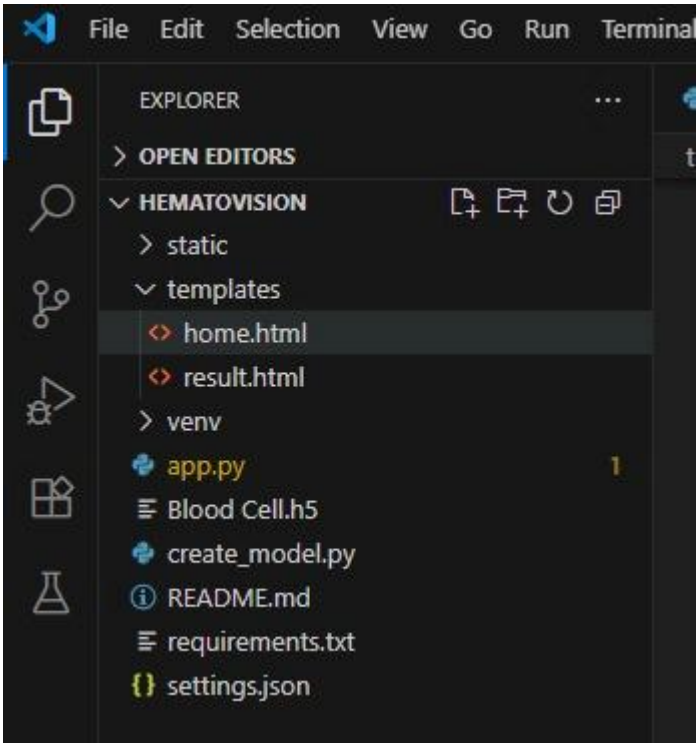
- Anaconda Navigator:
    - Refer to the link below to download Anaconda Navigator
- Python packages:
    - Open anaconda prompt as administrator
    - Type “pip install numpy” and click enter.
    - Type “pip install pandas” and click enter.
    - Type “pip install scikit-learn” and click enter.
    - Type ”pip install matplotlib” and click enter.
    - Type ”pip install scipy” and click enter.
    - Type ”pip install seaborn” and click enter.
    - Type ”pip install tenserflow” and click enter.
    - Type “pip install Flask” and click enter.

- **Installation:**

1. Create a Virtual Environment (Optional but Recommended):
   python -m venv .venv
   source .venv/Scripts/activate For Windows
   source .venv/bin/activate For Mac/Linux
2. Install Required Packages:
   Ensure you run:
   pip install numpy
   pip install tensorflow
   pip install Flask
   pip install Pillow
3. Run the Flask Web Application:
   python app.py
4. Access the Application:
   Open your browser and visit: [http://127.0.0.1:5050]

NOTE: After successful setup, you can upload blood cell images through the web interface to detect and classify the type of white blood cell (Neutrophil, Eosinophil, Monocyte, or Lymphocyte) using the trained deep learning model.

5.Folder Structure



6. Running the Application

A. Backend (Flask Application):

The Flask backend in the HematoVision project serves both the **prediction API** and the **frontend HTML templates** (index.html and result.html).

Steps to run the backend:

1. Open the terminal (inside your project folder).
2. Activate the virtual environment:
    - For **Windows**:

      ```
      venv\Scripts\activate
      ```

    - For **Mac/Linux**:

      ```
      source venv/bin/activate
      ```

3. Run the Flask application:

   ```
   python app.py
   ```

4. The server will start at:

   `http://127.0.0.1:5000`

A. Frontend:

The project uses Flask's **Jinja2 templates** (index.html, result.html) for the user interface. You do **not need to run any separate command** for the frontend. Once the Flask app is running, the UI will automatically load in your browser.

Access the web interface by opening:

`http://127.0.0.1:5000`

Flask Backend Endpoints for HematoVision:

- `/` → Home page with image upload form
- `/predict` → Handles the uploaded image and returns the predicted cell type
- `/result` → Displays the result (image + prediction)

7. API Documentation

a) Home Page

- **URL:** /
- **Method:** GET
- **Description:** Displays the landing page of the application.
- **Response:** Returns the index.html template.

b) Result Page

- **URL:** /about
- **Method:** GET
- **Description:** Displays the result after prediction is completed..
- **Response:** Returns `result.html` template with uploaded image and predicted class/

c) Image Prediction API

- **URL:** /predict
- **Method:** POST
- **Description:** Accepts a blood cell image, performs classification using the trained MobileNetV2 model, and returns the prediction.

Request Parameters:

Name	Type	Description
file	File	The image file to be uploaded (JPG, PNG, etc.).

Example Request using HTML Form:

```
<form method="POST" action="/predict" enctype="multipart/form-data">
<input type="file" name="image" required>
<button type="submit">Predict</button>
</form>
```

Example Response (Rendered on Inspect Page):

Upon successful prediction, the following details are displayed:

- Uploaded image preview
- Predicted blood cell type (e.g., “Lymphocyte”)
- Displayed on a new result page

Example Backend Response (if it were JSON API):

(Note: The current version displays output via HTML templates. JSON response is for future extensibility if needed.)

```
{
  "predicted_label": "Neutrophil",
  "confidence": 94.82,
  "image_path": "static/uploads/image123.jpg"
}
```

8. Authentication

The current version of the **HematoVision** project does **not** include any user authentication or authorization features. The application is designed as a **publicly accessible image classification tool** for educational and demonstration purposes. It allows any user to:

✔ Access the web interface

✔ Upload blood cell images for classification

✔ View prediction results without login or registration

Future Scope for Authentication (Optional Enhancements):

For improved security and controlled access in future versions, the following authentication and authorization methods can be implemented:

Method	Description
Session-Based Authentication	User credentials are verified on login, and a secure session is maintained using cookies.
Token-Based Authentication (JWT)	Issues a secure token on login, required for accessing protected routes or APIs.
Role-Based Access Control (RBAC)	Different user roles (e.g., Admin, Food Plant Head, Supermarket Manager) can be defined to restrict or grant access to specific features.

Recommended Future Features:

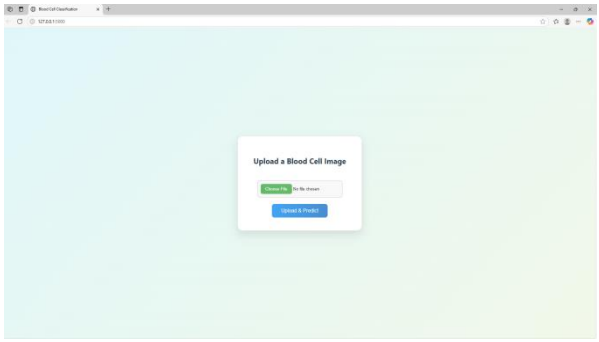
- **Admin Login:** Restrict access to sensitive features like retraining models or managing datasets.
- **User Dashboard:** Let registered users view previous predictions, upload history, and track results.
- **API Access Tokens:** Secure backend APIs for potential integration with mobile apps or external systems.

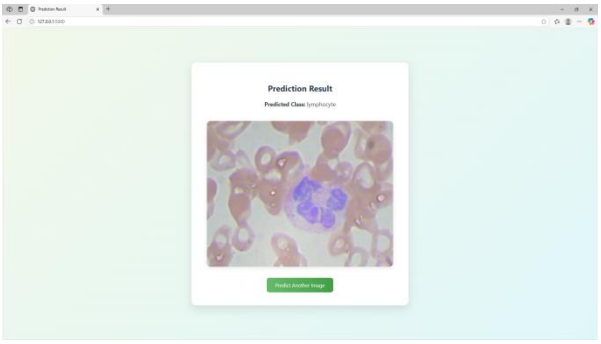
9. User Interface

The **HematoVision** project provides a simple and intuitive user interface built using **HTML**, **CSS**, and **Flask Jinja templates**. It is designed to ensure ease of use for all users, even those without technical knowledge.

UI Features:

- Minimalistic and responsive design
- Simple navigation and layout
- Image preview for confirmation
- Works on most browsers and devices





10. Testing

Testing Strategy

To ensure the accuracy and reliability of the **HematoVision** blood cell classification system, a combination of **manual and model-level testing** approaches was followed:

- **Model Evaluation Testing:**

  Evaluated the trained MobileNetV2 model using separate validation and test datasets.

  Generated performance metrics such as:

  - **Accuracy**
  - **Confusion Matrix**
  - **Classification Report**

  • Confirmed that the model performs well across all four cell types (Neutrophil, Eosinophil, Monocyte, Lymphocyte).

- **External Image Testing:**
  - Manually uploaded real blood smear images **not used during training**.

- Checked if the model could **generalize** well to new, unseen images.
- Observed that the model maintained high accuracy on real-world images.

- **Web Application Functional Testing:**
  - Tested core web pages (Home, Predict, Result) for proper navigation and correct display.
  - Verified that image upload functionality worked as expected via the Flask interface.
  - Confirmed that predictions were displayed correctly along with the uploaded image.

Tools Used

Tool/Library	Purpose
Python 3.10+	Programming language used for model training and backend development
TensorFlow / Keras	Deep learning framework used to build and train the MobileNetV2 model
Flask	Lightweight Python web framework for building the user interface and backend
NumPy	For handling numerical operations and data preprocessing
Pillow (PIL)	For image loading and resizing during prediction

11. Screenshots or Demo

Screenshots

The complete execution of the **HematoVision** application is shown step-by-step in the screenshots below.

Step 1: Run the app.py script

Once the script is run, the terminal displays a link to access the web app locally at http://127.0.0.1:5000.

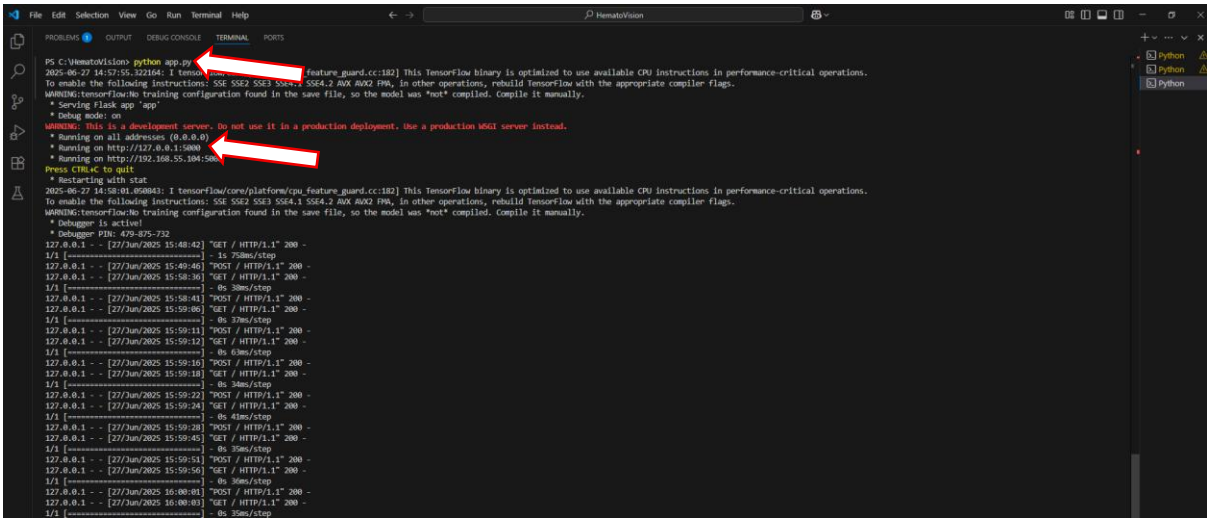


Fig 7.1.1: Code running in Terminal

*Step 2: **Access the Web Interface***

Clicking the local server link opens the HematoVision homepage in a browser.

.

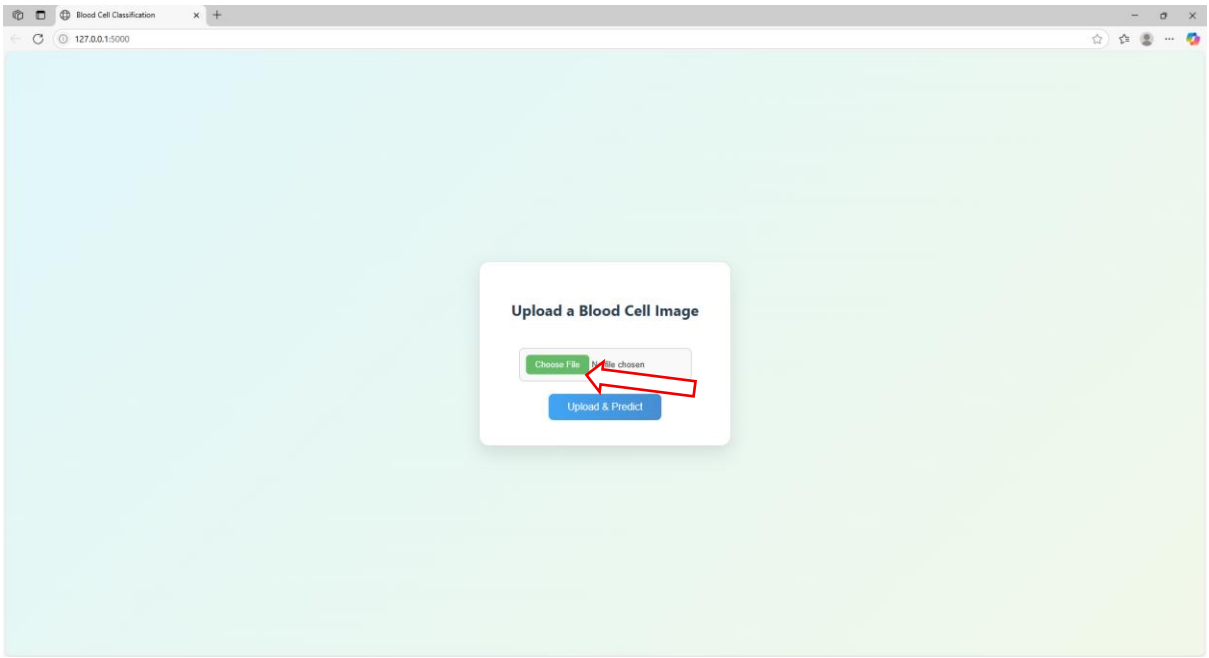


Fig 7.1.2: HematoVision Home Page

Step 3: Upload Image

Click “Choose File” to upload a blood cell image from your computer.

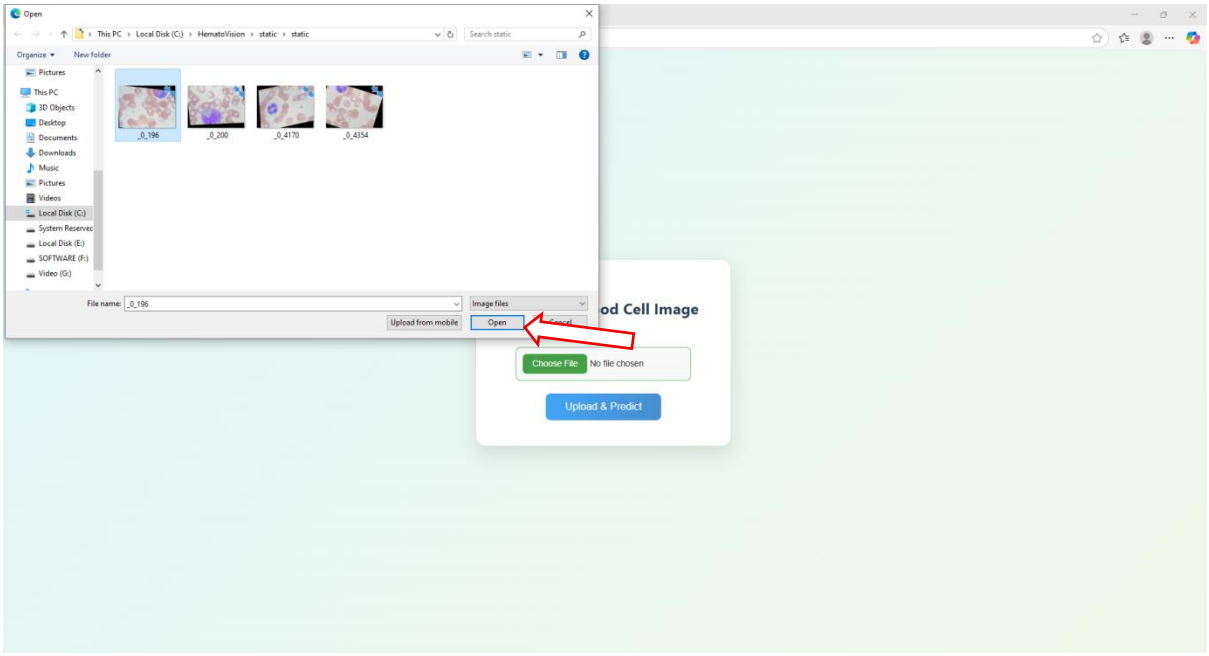


Fig 7.1.3: File upload page

Step 4: Click "Predict"

After selecting an image, click the **Predict** button to classify the cell type.

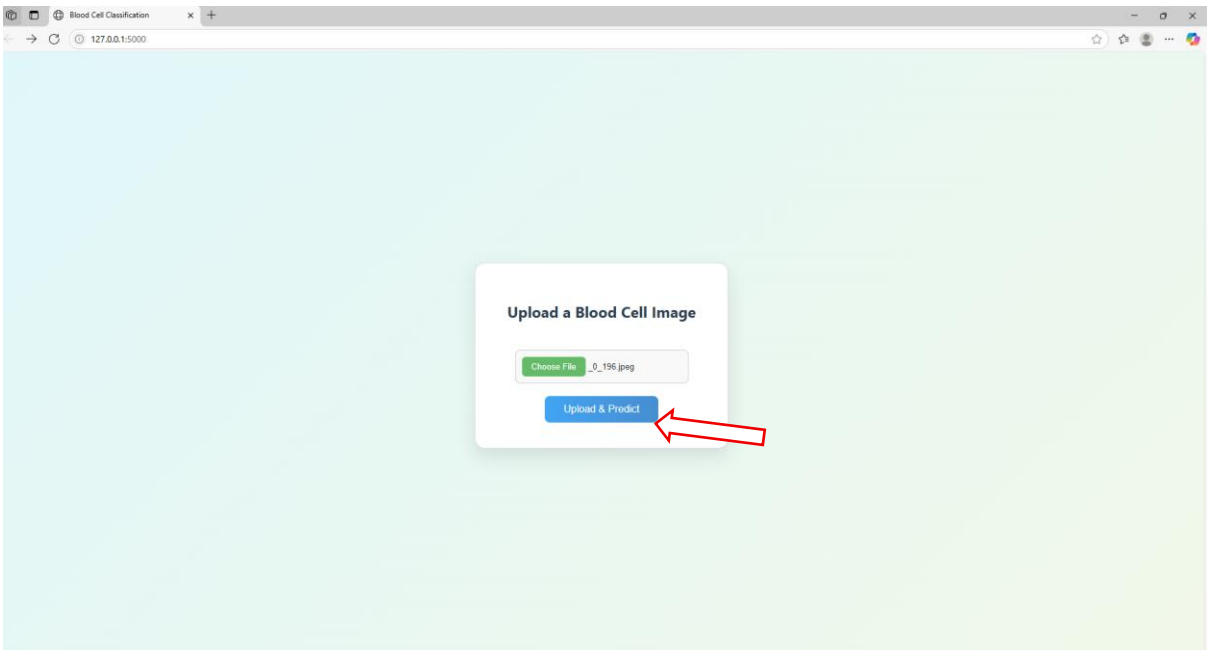


Fig 7.1.4: Selected image displayed on prediction page

Step 5: View Prediction Results

The application will display the uploaded image and the predicted cell type.

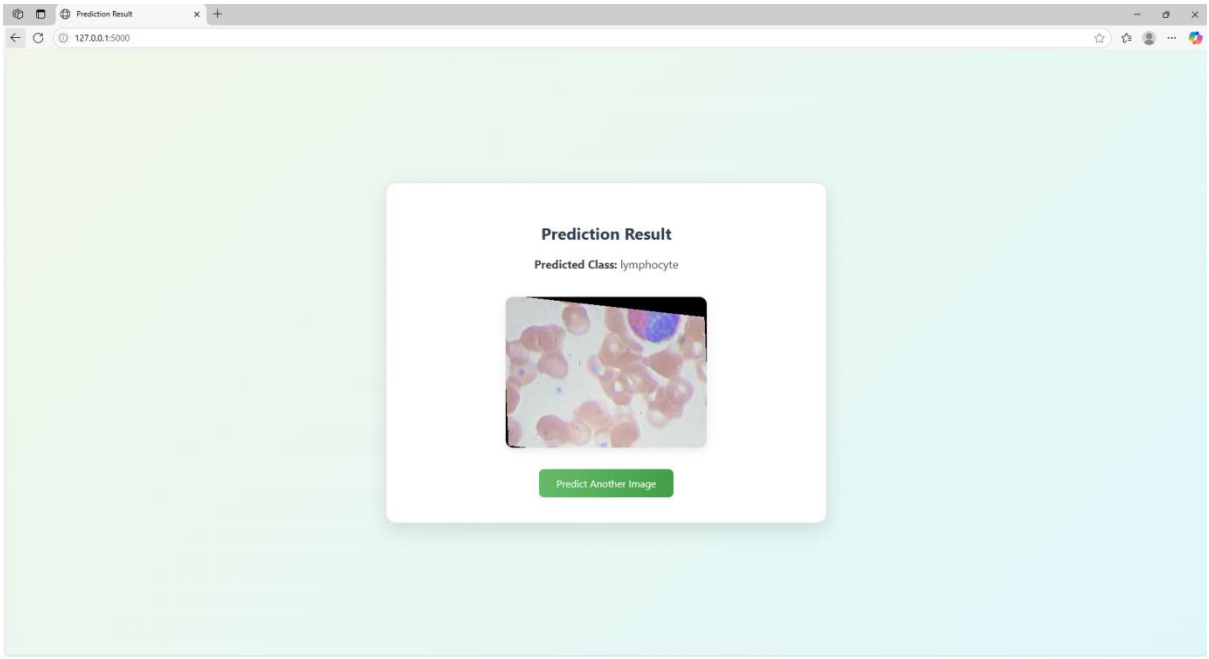


Fig 7.1.5: Prediction result with image preview

Step 6: After clicking the **Predict** button, the model processes the uploaded image and classifies the type of white blood cell (e.g., Lymphocyte, Neutrophil, etc.).
Below are some example outputs from real test images, showcasing accurate predictions and model confidence..

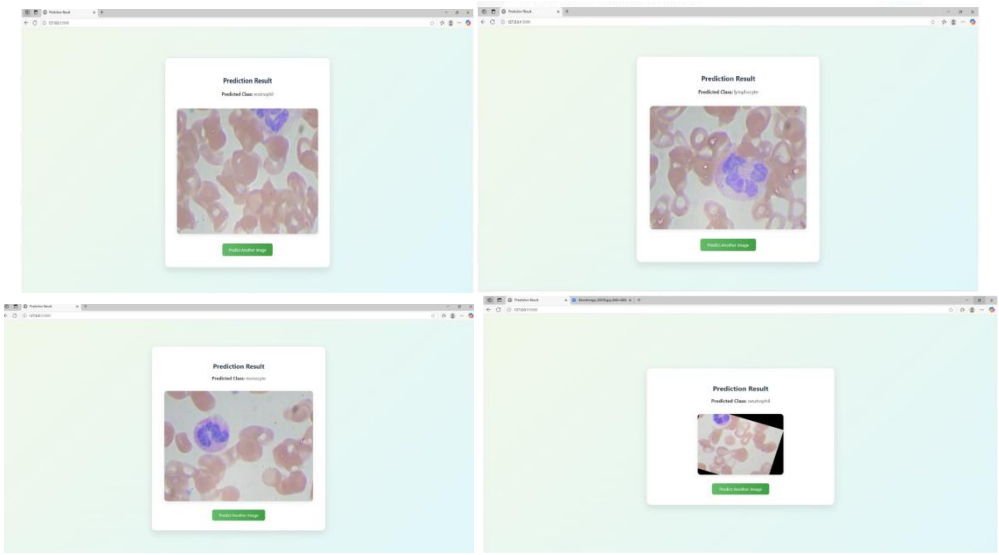


Fig 7.1.6: Prediction output for various test inputs with high accuracyAfter the Images

All uploaded images are stored in the static/uploads/ directory for reference or reuse.

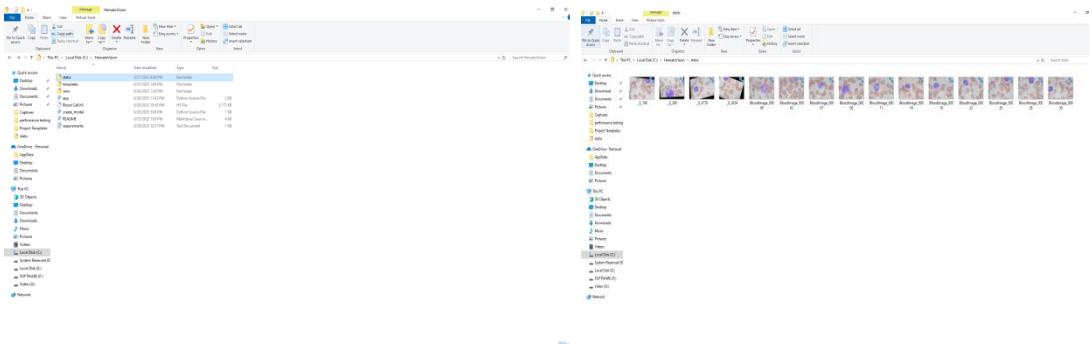


Fig 7.1.7: Folder structure showing stored uploaded images

Project Demo Link:

https://drive.google.com/file/d/1jkXonfIR4F6SJooRcLCZzL83cJE3ErhH/view?usp=drivesdk

12. Known Issues

Functional Issues

Model Misclassification:

 The model may misclassify images with poor focus, overlapping cells, or inconsistent staining.

- **Confidence Variability:**

Some external images (unseen during training) yield low confidence scores, even when predictions are correct.

File Handling Issues

- **Duplicate File Names:**
  If a user uploads a file with an already existing name, it may overwrite the previous image in static/uploads/.
- **UI/UX Issues**
- **Image Display Delay:**
  Larger image files may take a few seconds to display after prediction.
- **Security Issues**

No File Type Validation:

- The app does not strictly check image file formats, which could allow unsupported or potentially harmful files.

13. Future Enhancements

Model Improvements

- **Increase Accuracy**:

Train with more diverse blood smear datasets to improve accuracy.

- **Disease Detection**:

Extend the model to identify blood-related abnormalities like leukemia, anemia, or infection indicators.

Web Application Features

- **Image Preview Before Submission:**
- **Drag and Drop Uploads:**.
- **Batch Prediction:**

File Management

- **Auto-Renaming Uploaded Files**:

Add timestamps to filenames to prevent overwrites.

- **Prediction History Page**:

Let users view past predictions with image and result logs.

Security Enhancements

- **Strict File Type Checks**:

Allow only .jpg, .jpeg, .png uploads to prevent script injection.

- **User Login and Access Control**:

Only allow authorized users to access admin tools (e.g., retraining).

Reporting & Analytics

- **Prediction Report Download (PDF/CSV)**
- **Usage Dashboard** showing number of uploads, prediction types, accuracy trends

Deployment & Scalability

- **Cloud Hosting (Heroku, AWS, etc.)**
- **Mobile-Friendly UI** for phone or tablet access