HematoVision: Advanced Blood Cell Classification Using Transfer Learning

**Team Information**

* Team ID:  LTVIP2025TMID34062
* Team Leader:  TATA.Vishnu Priya Kavya
* Team Members:  Vasimalla Sai Dheeraj, Konda Vishnu Priya, Thota Siva Rama Krishna Prasad

**Table of Contents**

1. Project Overview
2. System Architecture
3. Technology Stack
4. Project Structure
5. Implementation Details
6. Development Workflow
7. Setup and Installation
8. Features and Functionality
9. API Documentation
10. Screenshots and Results
11. Challenges and Solutions
12. Future Enhancements
13. Conclusion

**1. Project Overview**

**HematoVision is a deep learning-based web application for classifying blood cell images into four categories: Eosinophil, Lymphocyte, Monocyte, and Neutrophil. It uses transfer learning with MobileNetV2 and is deployed via a Flask web interface. This project aids medical diagnostics by providing automated, accurate blood cell classification.**

**2. System Architecture**

**The system follows a modular architecture with three main layers:**

**a. Frontend Layer**

* **HTML/CSS templates to enable user interaction.**
* **File input to upload images.**

**b. Backend Layer (Flask)**

* **Handles routing, image uploads, and rendering templates.**
* **Passes image data to the model for prediction.**

**c. Model Layer**

* **A Convolutional Neural Network (CNN) trained using transfer learning (e.g., MobileNet).**
* **Outputs the predicted blood cell based on image analysis.**

**Workflow:**

1. **User uploads an image via the frontend.**
2. **Flask receives the request and saves the image.**
3. **Image is preprocessed and fed into the model.**
4. **Model returns prediction.**
5. **Flask renders the result on the web page.**

**3. Technology Stack**

* **Language: Python**
* **Framework: Flask**
* **Deep Learning: TensorFlow, Keras**
* **Frontend: HTML5, CSS3**
* **Libraries:**
  + **Pandas**
  + **matplotlib**
  + **tensorflow.keras**
  + **numpy**

**4. Project Structure**

**HematoVision/**

**├── static/**

**│ ├── css/**

**│ │ └── style.css**

**│ └── uploads/**

**├── templates/**

**│ ├── home.html**

**│ └── result.html**

**├── app.py**

**├── train\_model.py**

**├── Blood Cell.h5**

**├── requirements.txt**

**└── README.md**

**5. Implementation Details**

* **The model is trained on a dataset.**
* **Images are resized to 224x224 and normalized.**
* **A MobileNet-based CNN is used for better generalization and fast predictions.**
* **The model is saved as Blood Cell.h5 after training.**
* **Flask loads the model at runtime and performs inference on uploaded images.**

**6. Development Workflow**

1. **Data collection and preprocessing.**
2. **Export trained model.**
3. **Build Flask app (app.py) to serve the model.**
4. **Create HTML templates for the frontend.**
5. **Link form submission to prediction logic.**
6. **Test locally and refine the user experience.**

**7. Setup and Installation**

**Requirements:**

* **Python 3.8+**
* **pip**

**Installation Steps:**

**# Step 1: Navigate to project directory**

**cd HematoVision**

**# Step 2: Install dependencies**

**pip install -r requirements.txt**

**# Step 3: Run the application**

**python app.py**

**Access:**

**Open your browser and go to: http://127.0.0.1:5000/**

**8. Features and Functionality**

* **Clean and intuitive user interface.**
* **Upload blood cell images in .jpg or .png format.**
* **Real-time predictions using a trained deep learning model.**
* **Display of predicted blood type along with uploaded image.**
* **Error handling for invalid uploads or unsupported files.**

**9. API Documentation**

**GET /**

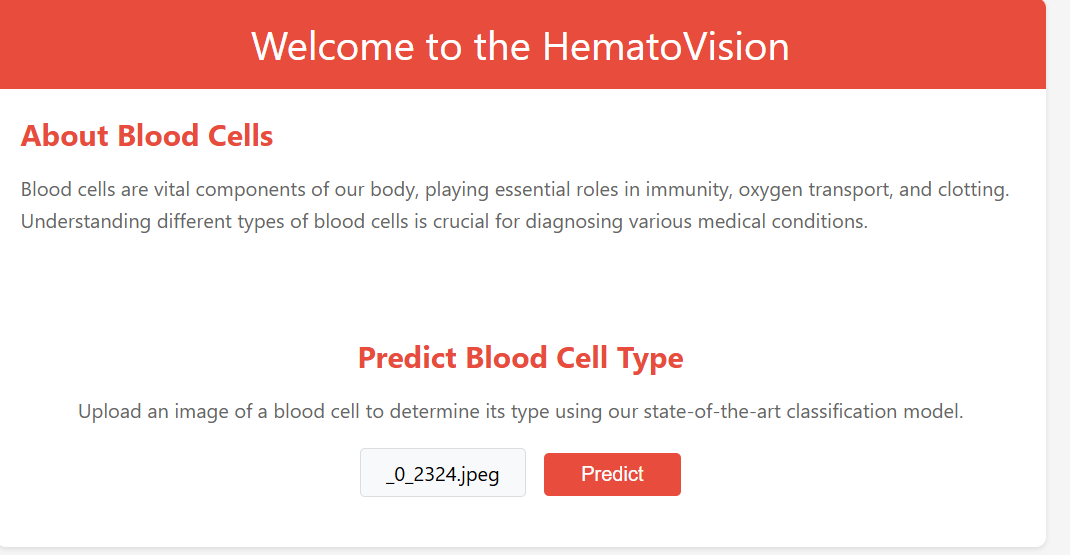
* **Loads the homepage.**
* **Returns the upload form.**

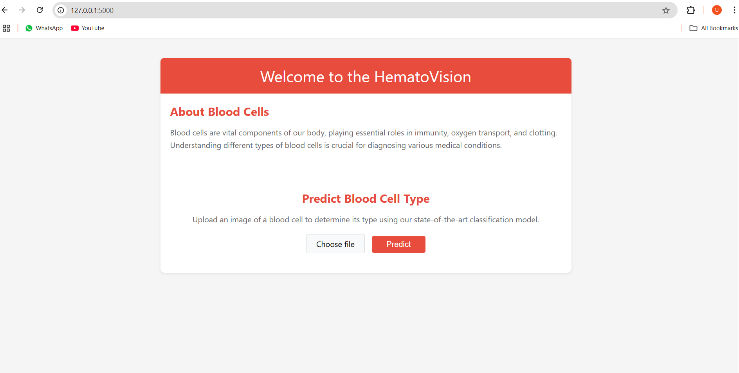
**POST /predict**

* **Accepts an image file.**
* **Preprocesses the image and predicts blood type.**
* **Returns the result page with the prediction and image.**

**10. Screenshots and Results**

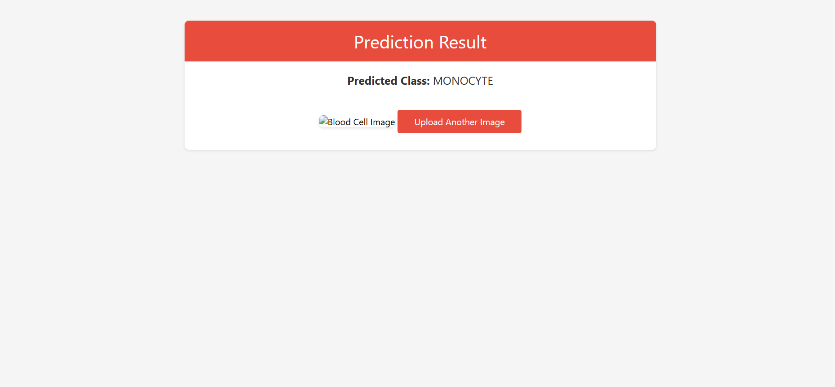
**Home Page:**

* **File input and upload button. **

****

**Result Page:**

* **Displays uploaded image.**
* **Shows predicted blood type (e.g., "monocyte").**

****

**11. Challenges and Solutions**

|  |  |
| --- | --- |
| **Challenge** | **Solution** |
| **Similar appearance across classes** | **Used transfer learning for robust feature learning** |
| **Large model size** | **Optimized with MobileNet to reduce size** |
| **UI responsiveness** | **Applied lightweight custom CSS** |
| **Handling unsupported file formats** | **Added file validation in Flask** |

**12. Future Enhancements**

* **Deploy on cloud (Render, AWS, or Heroku).**
* **Add mobile responsiveness to UI.**
* **Use cloud storage for uploaded images.**
* **Implement top-3 prediction output.**
* **Add multilingual support.**
* **Train on larger and more diverse blood cell datasets.**

1. **Conclusion**

**HematoVision successfully demonstrates the power of transfer learning in accurately classifying blood cells using microscopic images.** **By leveraging a large annotated dataset and MobileNetV2, the model achieves high precision with reduced training time and computational resources.** **Its integration into a Flask web application enables real-time and user-friendly diagnostics.** **The project holds strong potential in clinical settings for automating diagnostics and improving pathologist workflows.** **It also empowers remote healthcare delivery by enabling image-based consultations and analysis.** **Overall, HematoVision contributes significantly to modernizing blood analysis and improving global healthcare access.**

**Git Hub Link** : <https://github.com/Vishnupriyakavya/HematoVision-blood_cells->