PROJECT

Project Title :

HematoVision: Advanced Blood Cell Classification Using Transfer Learning

Team Name :

Visionary Hematons

Team Members :

- Dhone Madhu
- Chinta Meghana
- Chilakala Vishnuvardhan Reddy
- Chilaka Kalyan Babu

Phase-1 : Brainstorming & Ideation

Objective:

To develop an AI-powered blood cell classification system using transfer learning that enhances diagnostic accuracy, reduces manual analysis time, and supports healthcare applications such as automated diagnostics, remote consultations, and medical education.

Key Points:

1. **Problem Statement:**
    - Microscopic analysis of blood cells plays a critical role in diagnosing conditions such as leukemia, anemia, and infections. However, manual analysis is time-consuming and subject to human error.
    - Traditional blood cell classification requires extensive expertise and can lead to inconsistent results due to human fatigue and subjective interpretation.
    - HematoVision utilizes transfer learning with pre-trained CNN models to classify blood cell types from microscopic images accurately, enhancing diagnostic speed, precision, and reliability for early disease detection and improved patient outcomes.

2. **Proposed Solution:**
    - An AI-powered web application that uses a custom-built Convolutional Neural Network (CNN) architecture optimized for high accuracy and better generalization to classify blood cells from microscope images in real-time accurately.
    - The system processes a dataset of 12,500 augmented blood cell images across 4 categories: Eosinophils, Lymphocytes, Monocytes, and Neutrophils.
    - It helps doctors by providing quick and reliable results, reducing manual effort, and enhancing the speed and accuracy of diagnosis.

3. Target Users:

- Pathologists and lab technicians who need quick and accurate blood cell analysis
- Hospitals and diagnostic centers aiming to automate blood smear classification
- Healthcare providers offering remote consultations who require fast diagnostic support
- Medical students and trainees learning about blood cell identification
- Telemedicine platforms requiring automated diagnostic capabilities

4. Expected Outcome:

- A functional AI-powered web application that accurately classifies blood cells from images using transfer learning
- Provides fast, reliable diagnostic support to improve efficiency in medical analysis and education
- Scalable solution that can be integrated into existing healthcare systems

Phase-2: Requirement Analysis

Objective:

Define the technical and functional requirements for the HematoVision application.

Key Points:

1. **Technical Requirements:**
   - **Programming Language:** Python
   - **Python Packages:** NumPy, Pandas, Scikit-learn, Matplotlib, SciPy, Seaborn, TensorFlow, Flask
   - **Deep Learning Framework:** TensorFlow for model building and training
   - **Web Framework:** Flask for web application development
   - **Pre-trained Model:** MobileNetV2 (used for transfer learning)
   - **Development Environment:** Anaconda Navigator
   - **Development Tools:** Command Line (pip install), Jupyter Notebook

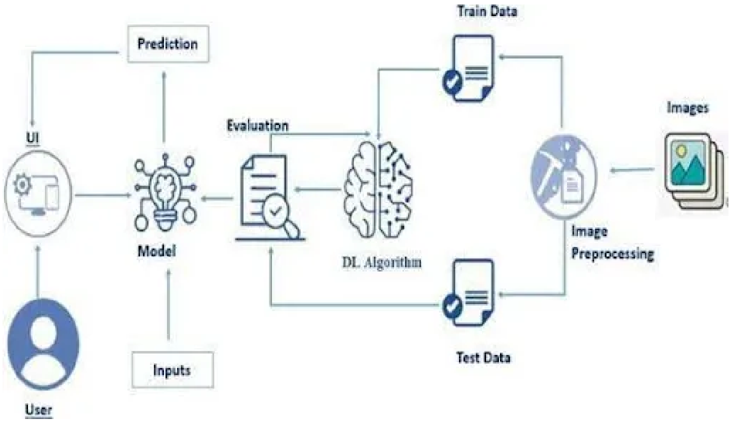
2. **Functional Requirements:**
   - Ability to upload microscopic blood cell images through the web interface
   - Classify blood cells into 4 types: Eosinophils, Lymphocytes, Monocytes, and Neutrophils using a trained model
   - Display classification results along with prediction confidence scores
   - Provide an intuitive and user-friendly interface for doctors, students, and lab technicians
   - Real-time image processing and prediction capabilities
   - Responsive web design for various devices

3. **Constraints & Challenges:**
   - Handling imbalanced data across different blood cell classes
   - Managing low-quality or blurry microscope images that affect prediction accuracy
   - Optimizing model size and performance for faster processing and easy deployment
   - Ensuring the web interface is responsive and user-friendly on all devices
   - Minimizing overfitting while maintaining high accuracy across all cell types

Phase-3: Project Design

Objective:

Develop the architecture and user flow of the HematoVision application.



Key Points:

1. **System Architecture:**
   - **Frontend Layer:** HTML templates rendered through the Flask framework
   - **Backend Layer:** Flask web server handling HTTP requests and responses
   - **Model Layer:** Pre-trained MobileNetV2 model fine-tuned for blood cell classification
   - **Data Processing Layer:** Image preprocessing pipeline including resizing, normalization, and augmentation

2. **User Flow:**
   - **Step 1:** The User opens the HematoVision web application.
   - **Step 2:** Upload a microscope image of a blood cell sample.
   - **Step 3:** The system preprocesses the image and feeds it to the trained model.
   - **Step 4:** The model processes the image and classifies the cell type.
   - **Step 5:** The predicted cell type and confidence score are displayed on the results page.
   - **Step 6:** The User can upload additional images or use the results for diagnostic insights.

Phase-4: Project Planning

Objective:

Break down development tasks for efficient completion across multiple sprints.

Sprint	Task	Priority	Duration	Deadline	Assigned to	Dependencies	Expected outcome
Sprint 1	Environment Setup & Package Installation	🔴High	3 hours	Day 1	Member 1	Anaconda, Python	Project environment ready
Sprint 1	Dataset Collection & Preprocessing	🔴High	4 hours	Day 1	Member 2	Dataset access	Clean, prepared image dataset
Sprint 2	Model Building using Transfer Learning	🔴High	6 hours	Day 2	Member 3	Preprocessed data, TensorFlow	Trained classification model
Sprint 2	Flask Web App Integration	🟡Medium	4 hours	Day 2	Member 1 & 4	Trained Model, Flask installed	Working web interface
Sprint 3	Testing & Debugging	🟡Medium	3 hours	Day 3	Member 2 & 3	Complete System	Bug-free and responsive system
Sprint 3	Final Presentation & Deployment	🟢Low	2 hours	End of Day 3	Entire Team	Working application	Project deployed and demo-ready

Sprint Planning with Priorities

Sprint 1 - Setup & Preparation (Day 1)

🔴 High Priority

- Set up the development environment using Anaconda Navigator
- Install all required Python packages (TensorFlow, Flask, NumPy, Pandas, etc.)
- Download and organize the blood cell image dataset from Kaggle
- Perform initial data exploration and preprocessing

Sprint 2 – Model Development & Integration (Day 2)

🔴 High Priority

- Build and train the blood cell classification model using a custom-built Convolutional Neural Network (CNN) transfer learning
- Implement data augmentation techniques to improve model robustness
- Integrate the trained model with the Flask web application
- Develop HTML templates for user interface

Sprint 3 – Testing, Deployment & Submission (Day 2)

🟡 Medium Priority

- Test the application functionality across different scenarios
- Fix bugs and improve UI responsiveness
- Optimize model performance and prediction speed

🟢 Low Priority

- Finalize deployment configuration
- Prepare presentation and demo materials
- Document the project and create user guides

Phase-5: Project Development

Objective:

Implement the core features of the HematoVision application using transfer learning for blood cell classification.

Key Points:

1. **Technology Stack Used:**
    - **Frontend:** HTML (via Flask templates), CSS for styling
    - **Backend:** Flask Framework (Python)
    - **Deep Learning:** TensorFlow with pre-trained MobileNetV2 model
    - **Programming Language:** Python 3.x
    - **Data Processing:** NumPy, Pandas for data manipulation
    - **Visualization:** Matplotlib, Seaborn for data analysis

2. **Development Process:**
    - Built a custom CNN architecture with 15M+ parameters for deeper learning
    - Added custom classification layers with dropout for regularization
    - Implemented transfer learning by freezing base model weights
    - Trained the model for 20 epochs using SGD optimizer with learning rate decay
    - Applied callbacks, including ModelCheckpoint and EarlyStopping, for optimal training

3. **Challenges & Fixes:**

- **Challenge:** Model overfitting on certain blood cell types
  Fix: Implemented dropout regularization and data augmentation techniques

- **Challenge:** Large model size causing slow predictions
  Fix: Used MobileNetV2 for efficient mobile deployment and optimized model loading

- **Challenge:** Image quality variation affecting predictions
  Fix: Added robust preprocessing steps including resizing, normalization, and error handling

Phase-6: Functional & Performance

Objective:

Ensure that the HematoVision application performs accurately, reliably, and consistently across various test cases and environments.

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Upload image of Eosinophils	Correct cell type identified with confidence score	✅ Passed	Tester 1
TC-002	Functional Testing	Upload image of Lymphocytes	Accurate classification with high confidence	✅ Passed	Tester 2
TC-003	Functional Testing	Upload image of Monocytes	Proper identification and classification	✅ Passed	Tester 3
TC-004	Functional Testing	Upload image of Neutrophils	Correct prediction with confidence metrics	✅ Fixed	Tester 1
TC-005	Performance Testing	Check model response time	Results displayed under 3 seconds	✅ Passed	Tester 2
TC-006	Error Handling	Upload non-image file	Appropriate error message displayed	🚀 Deployed	Developer
TC-007	UI Responsiveness	Test on mobile browser	Layout adjusts properly on mobile devices	⚠️ Needs Optimization	Tester 3
TC-008	Deployment Testing	Hosted on local server and accessed remotely	App loads and predicts successfully online	🚀 Deployed	DevOps