HematoVision: Advanced Blood Cell Classification Using Transfer Learning

Team Information

Team ID: LTVIP2025TMID40243
Team Members:
- Jangam Shreya (Team Leader)
- Gundrathi Sai Sravanthi
- Hanchate Sai Manikanta
- Guni Reddy Charan Kumar Reddy

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.

Problem Statement:
Manual microscopic analysis is critical but time-consuming and prone to error. HematoVision addresses this by automating classification using AI and transfer learning for faster and more accurate diagnostics.

Proposed Solution:
A web-based AI application using VGG16 model to classify blood cells in real time, providing quick and reliable insights.

Target Users:
- Pathologists and lab technicians
- Hospitals and diagnostic labs
- Doctors offering remote consultations
- Medical students and trainees

Expected Outcome:
A reliable application that classifies blood cells with high accuracy and confidence scores, enhancing diagnostic workflows.

Phase 2: Requirement Analysis

Technical Requirements:
- Language: Python

- Libraries: NumPy, Pandas, Matplotlib, TensorFlow, Flask, etc.
- Frameworks: Flask (web), TensorFlow (deep learning)
- Pre-trained Model: VGG16
- Tools: Command Line, Anaconda

Functional Requirements:
- Upload microscopic blood cell images
- Predict cell type and show confidence
- Easy-to-use responsive UI

Constraints & Challenges:
- Imbalanced classes and poor-quality images
- Optimizing for performance and deployment
- UI responsiveness on all devices

Phase 3: Project Design

System Architecture:
1. User uploads blood cell image
2. Flask backend sends image to VGG16 model
3. Model predicts cell type
4. UI shows predicted class with confidence score

User Flow:
Step 1: Launch web app
Step 2: Upload image
Step 3: Model processes input
Step 4: Result is shown to user
Step 5: Repeat as needed

Architecture Diagram:

[Insert architecture diagram here]

Phase 4: Project Planning

Sprint 1 (Day 1):
- Setup environment & packages
- Dataset collection and preprocessing

Sprint 2 (Day 2):
- Build and train model with transfer learning
- Integrate Flask web interface

Sprint 3 (Day 2):
- Test, debug, improve responsiveness
- Final deployment and presentation prep

Phase 5: Project Development

Technology Stack:
- Frontend: HTML + Flask Templates
- Backend: Flask
- Deep Learning: TensorFlow + VGG16
- Language: Python

Development Steps:
- Trained classifier on 12,000 image dataset
- Integrated model into Flask app
- Developed image upload and prediction UI

Challenges:
- Overfitting resolved with dropout & augmentation
- Slow predictions optimized by saving in .h5
- Added preprocessing for blurry/low-res images

Phase 6: Testing & Performance

Test Cases Summary:
- TC001: Eosinophil image - Passed
- TC002: Mixed WBC/RBC image - Passed
- TC003: Poor lighting - Fixed
- TC004: Mobile display - Failed
- TC005: Performance (<2s) - Needs Optimization
- TC006: Remote server access - Deployed

Overall, HematoVision achieved successful predictions and accuracy across most scenarios, with continued improvement scope for UI responsiveness and performance.