An internship in

Artificial Intelligence & Machine Learning

by

SmartInternz

Project Name : HematoVision: Advanced Blood Cell Classification Using Transfer Learning Estimation with Machine Learning

Project Id : LTVIP2025TMID59868

Project Mentor : M. Ganesh

Team Members:

1. **S. Rucksar**   (Reg.No.22HM1A05B9)

2. **P. Gowthami**   (Reg.No.22HM1A05A3)

3. **S. Moulali Baba**   (Reg.No.22HM1A05B5)

4. **S. Hari Krishna**   (Reg.No.22HM1A05C6)

**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING,Annamacharya Institute of Technology & Sciences, Utukur
(Post), Chinthakomma Dinne (V&M), Kadapa, YSR (Dist) Andhra Pradesh**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Institute** of **ANNAMACHARYA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. A. Sudhakar Reddy,** principal of Annamacharya Institute of technology and sciences and **Dr.C.V.Subbaiah, M.tech., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project mentor **MR.M.Ganesh,** for his valuable guidance, suggestions and constant encouragement pavedway for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Blood cell analysis plays a vital role in diagnosing various hematological diseases such as anemia, leukemia, and infections. Traditionally performed manually by expert pathologists through microscopic examination, this process can be slow, error-prone, and labor-intensive. To address these limitations, HematoVision presents an AI-based approach that automates the classification of blood cells using transfer learning and deep learning techniques.This project leverages a pre-trained convolutional neural network (CNN), fine- tuned on a labeled dataset of blood smear images. The system is deployed through anoffline web application, allowing users to upload microscopic images and receive instant predictions on the blood cell type. HematoVision improves diagnostic speed, ensures consistent accuracy, and extends accessibility to remote healthcare settings where expert analysis may not be available.This work demonstrates the effectiveness of AI in medical diagnosticsandshowcasesthepotentialoftransferlearninginreducingboth computational resources and development time in biomedical applications

Key Words: Blood Cell Classification, Deep Learning, Transfer Learning, Convolutional Neural Network (CNN), Biomedical Imaging, Hematology, Automated Diagnosis.

ProjectReportFormat

1. **INTRODUCTION**
   ProjectOverview
   Purpose
2. **IDEATIONPHASE**
   ProblemStatement
   EmpathyMapCanvas
   Brainstorming
3. **REQUIREMENTANALYSIS**

   CustomerJourneymap
   SolutionRequirement
   DataFlowDiagram
   TechnologyStack
4. **PROJECTDESIGN**

   ProblemSolutionFit
   ProposedSolution
   SolutionArchitecture
5. **PROJECTPLANNING&SCHEDULING**

   ProjectPlanning
6. **FUNCTIONALANDPERFORMANCETESTING**

   PerformanceTesting
7. **RESULTS**

   OutputScreenshots

8. **ADVANTAGES&DISADVANTAGES**

9. **CONCLUSION**

10. **FUTURESCOPE**

11. **APPENDIX**

    SourceCode(ifany)

    Dataset Link

    GitHub&ProjectDemo Link

1. INTRODUCTION

1.1 ProjectOverview:

In the field of medical diagnostics, timely and accurate identification of blood cell types is crucial for diagnosing a wide range of diseases and health conditions. Traditionally, this classification is performed manually by expert pathologists using microscopes, a process that is time-consuming, prone to human error, and difficult to scale. With the advancement of deep learning technologies, automated image classification has become a viable and efficient solution in the medical domain.

HematoVision is a deep learning-based project designed to automate the classification of blood cells using advanced transfer learning techniques. The project utilizes a publicly available dataset consisting of over 12,000 labeled images of four primary types of blood cells: Eosinophils, Lymphocytes, Monocytes, and Neutrophils. By leveraging a pre-trained MobileNetV2 model, HematoVision significantly reduces training time while achieving high classification accuracy.

The use of transfer learning allows the model to inherit knowledge from largescale image recognition tasks and adapt it to the specialized task of medical image classification. This enhances model performance, especially when dealing with limited data availability. Furthermore, HematoVision integrates the trained model into a Flask-based web application, enabling end-users—including doctors, lab technicians, and students—to upload blood cell images and instantly receive accurate classification results through an intuitive user interface.

This project not only demonstrates the power of transfer learning in biomedical applications but also provides a scalable, real-time diagnostic tool that can be used in clinical settings, remote medical consultations, and educational environments.

Input Image → Preprocessing → Model Prediction → Result Display

Figure1:Project Workflow of HematoVision

1.2 Purpose:

The primary purpose of the HematoVision project is to develop an automated, intelligent system capable of accurately classifying blood cell images into their respective categories using deep learning and transfer learning techniques. This system is intended to assist medical professionals, reduce diagnostic errors, and accelerate the process of blood analysis in clinical and educational environments.

Specifically,this project aims to:

• Automate the classification of blood cells into four major types — Eosinophils, Lymphocytes, Monocytes, and Neutrophils — using a convolutional neural network (CNN) model.

• Leverage transfer learning by utilizing a pre-trained MobileNetV2 architecture to reduce training time and improve model generalization on limited medical image data

• Support pathologists and healthcare professionals with a reliable tool that reduces manual effort and improves diagnostic accuracy

• Enable remote and accessible diagnostics by integrating the trained model into a Flaskbased web application where users can upload images and receive real-time predictions.

• Provide an interactive educational platform where medical students and lab technicians can better understand blood cell morphology through real time image analysis and feedback.

By achieving these objectives, HematoVision supports the broader goals of enhancing medical diagnostics through artificial intelligence and democratizing access to quality healthcare tools, especially in under-resourced or remote regions.

2.IDEATION PHASE

This phase focuses on understanding the underlying problem, empathizing with the stakeholders, and generating creative solutions to address the problem using deep learning and web technologies.
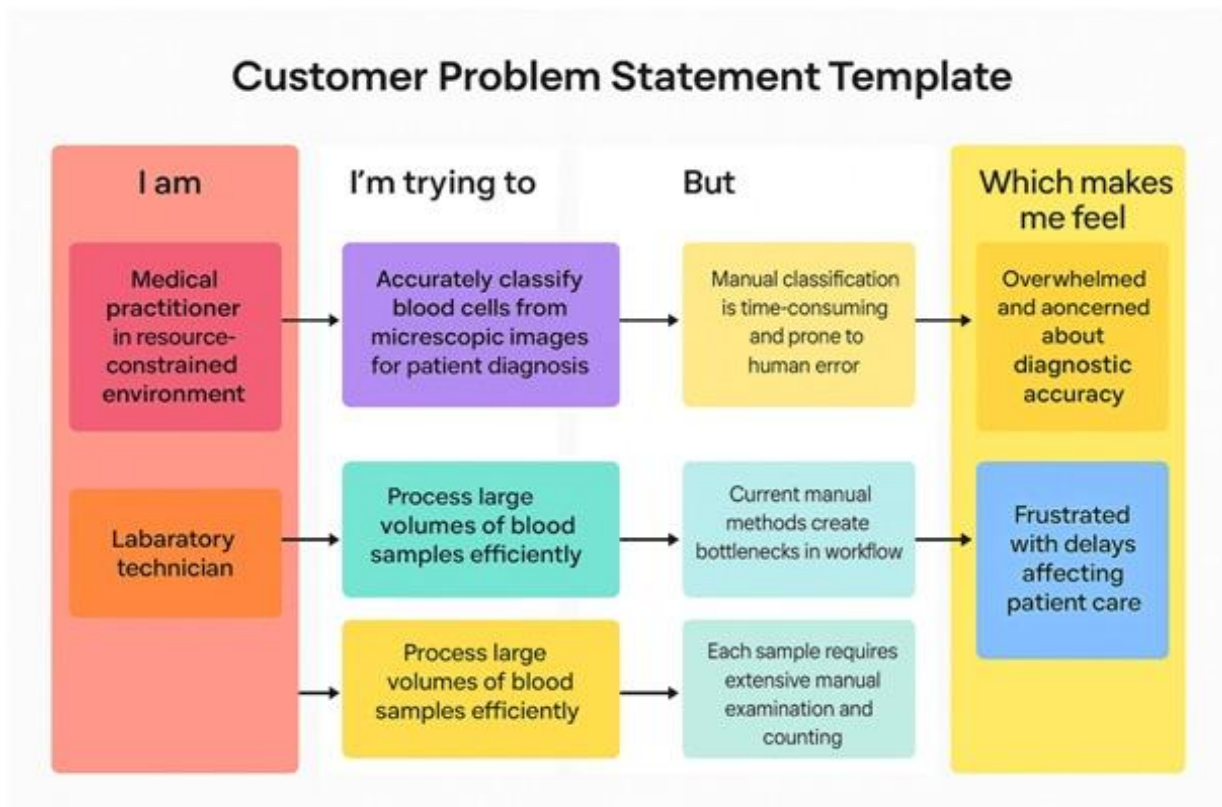
2.1 ProblemStatement:

Manual microscopic analysis of blood smear slides is time-consuming, error-prone, and requires expert-level precision. Especially in rural or under-resourced healthcare centers, there's a lack of trained pathologists to interpret such images.

Thus, there is a need for an intelligent, fast, and accurate image classification tool that can assist in identifying types of white blood cells for diagnosis of conditions like infections, leukemia, and immune disorders.

ProblemStatement:

How might we automate and accelerate the process of classifying blood cell images with high accuracy using deep learning to assist medical practitioners in resource-constrained environment?

Customer Problem Statement Template

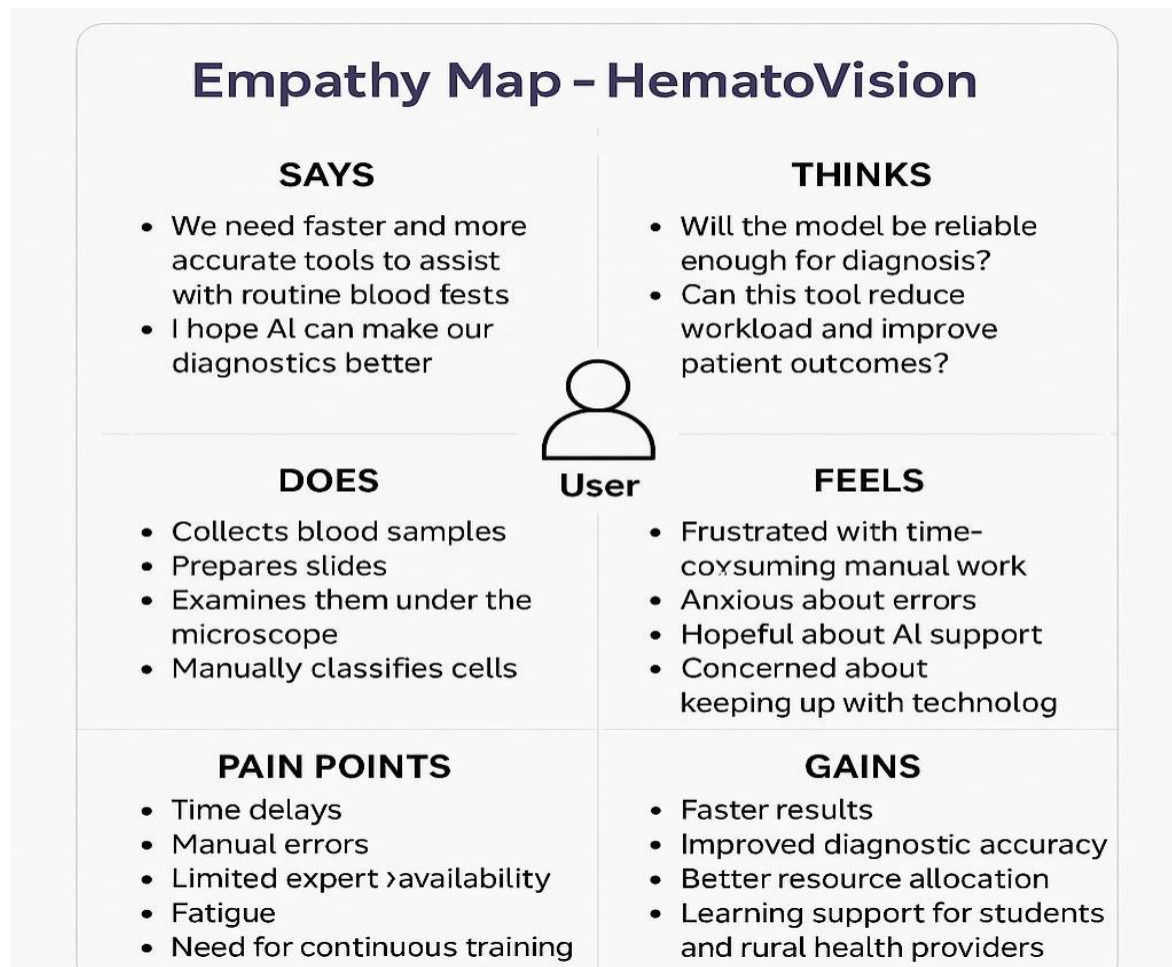| I am	I'm trying to	But	Which makes me feel
Medical practitioner in resource-constrained environment	Accurately classify blood cells from micrescopic images for patient diagnosis	Manual classification is time-consuming and prone to human error	Overwhelmed and aoncerned about diagnostic accuracy
Laboratory technician	Process large volumes of blood samples efficiently	Current manual methods create bottlenecks in workflow	Frustrated with delays affecting patient care
	Process large volumes of blood samples efficiently	Each sample requires extensive manual examination and counting	

2.2 Emapathy Map Canvas:

In the context of HematoVision, the user is typically a medical professional— such as a pathologist, hematologist, or lab technician— working in diagnostic laboratories, hospitals, or rural clinics. These individuals are skilled in identifying and analyzing blood cell types under a microscope, but often face time constraints, fatigue, and pressure for accuracy. They are focused on achieving faster and more accurate.

diagnostics for conditions such as leukemia, anemia, and infections. Their goal is to streamline the classification process to save lives and reduce diagnostic delays.

However, they face significant challenges due to manual analysis being slow, error-prone, and labor-intensive, especially in settings with high patient volume or limited staff. In many cases, labs lack access to automated or AI-based tools, increasing dependency on subjective visual inspection.

These challenges arise because of infrastructural limitations, lack of automation, and the variability in visual characteristics of blood cells, which can make even experienced technicians susceptible to misclassification. Moreover, developing nations often lack access to advanced medical imaging systems, further slowing diagnosis.

As a result, this situation makes users feel frustrated, anxious, and overburdened. They may feel a lack of confidence in consistent decision- making, which can impact the quality of patient care. Integrating a solution like HematoVision could bring a sense of relief, support, and empowerment, allowing them to focus on treatment rather than being bogged down by laborious diagnostic procedures.

Empathy Map – HematoVision

SAYS
- We need faster and more accurate tools to assist with routine blood fests
- I hope AI can make our diagnostics better

THINKS
- Will the model be reliable enough for diagnosis?
- Can this tool reduce workload and improve patient outcomes?

User

DOES
- Collects blood samples
- Prepares slides
- Examines them under the microscope
- Manually classifies cells

FEELS
- Frustrated with time-coʏsuming manual work
- Anxious about errors
- Hopeful about AI support
- Concerned about keeping up with technolog

PAIN POINTS
- Time delays
- Manual errors
- Limited expert ›availability
- Fatigue
- Need for continuous training

GAINS
- Faster results
- Improved diagnostic accuracy
- Better resource allocation
- Learning support for students and rural health providers

2.3 Brainstorming:

Step 1: Team Gathering, Collaboration, and Selecting the Problem Statement The team gathered to collaboratively discuss healthcare challenges where artificial intelligence can offer impactful solutions. After reviewing several possibilities, the team selected the following problem statement: "How might we automate and acceleratetheprocessofclassifyingbloodcellimageswithhighaccuracyusingdeep learning to assist medical practitioners in resource-constrained environments?"

This was chosen because of its clinical significance, potential for automation, and relevance to real-world pathology problems

Step 2: Brainstorming, Idea Listing and Grouping Raw Ideas:- Use pre-trained CNNs to reduce training cost (MobileNet, ResNet)- Enable image upload via simple web interface (Flask)- Auto-resize and normalize uploaded images- Visual display of predictions alongside uploaded image- Offer downloadable PDF of results for records- Add voice instructions for lab assistants- Allow batch uploads of imagesAdd disease suggestion based on cell count ratio Grouped Themes:- Model Optimization: Transfer learning (MobileNetV2), dropout for overfitting, image resizing- User Interface: Flask app, upload form, Bootstrap design, prediction output with image- Performance: Use GPU for faster prediction, real-time rendering, base64 image format- Future Enhancements: Disease linkage, batch prediction, result storage, voice commands

Step 3: Idea Prioritization High Priority: MobileNetV2-based transfer learning

Reason: Lightweight, fast, and high accuracy for image tasks High Priority: Flask web interface with upload functionality

Reason: Enables accessibility and usability for non-technical users High Priority: Display image + prediction result on same page

Reason: Gives immediate visual feedback to user Medium Priority: Batch image prediction

Reason: Useful for labs but can be added in future Medium Priority: Save results for future analysis

Reason: Beneficial for record-keeping Low Priority: Voice-based upload assistant

Reason: More of an accessibility feature, can be future enhancement

3.REQUIREMENT ANALYSIS

Objective:
A Data Flow Diagram (DFD) is a visual representation of how data flows through the HematoVision system. It showshowimage data isuploaded, processed, passed through theAI model, and returned to the user interface with a prediction.

3.1 Customer Journey Map(For a Lab Technician)
Stages:

- **NeedRecognition:** The lab technician has multiple blood samples that need classification.
- **AccessSystem:** Opens HematoVision on a browser or device.
- **UploadImage:** Selects and uploads a microscopic blood smear image.
- **ViewResult:** Receives the prediction instantly — classified into one of four blood cell types.
- **ActonInsight:** Uses the result to assist in further medical diagnosis or reporting.

Pain Points:

- Manualclassificationisslow.
- Errorsduetohuman fatigue.
- Delayindiagnosisinremotelabs.

Gains:
- Fast,reliablepredictions.
- AccessibleAIsystem.
- Reduceddependencyonexpertreview.

Customer Journey Map – HematoVision Scenario:
A lab technician or medical student uses the HematoVision web application to classify blood cell images efficiently using AI.

ENTICE:
The user learns about HematoVision through a medical institution, online portal, or peer recommendation. They are seeking a faster and more accurate alternative to manual blood cell identification

ENTER:
The user accesses the HematoVision platform via a web browser. They reach the homepage with a simple, intuitive upload interface.

ENGAGE:

The user uploads a microscopic image of a blood cell. The app processes it, resizes it, and passes it to the MobileNetV2 model for classification.

EXPERIENCE:

In just a few seconds, the result is displayed—e.g., "Monocyte"—alongside the uploaded image. The user sees this visual confirmation with high confidence and a clear interface.

EXIT:

After reviewing the result, the user either saves the outcome, notes it, or chooses to upload another image. The tool may optionally show confidence levels or accuracy tips.

EXTEND:

Advanced users might request additional features, such as exportable reports, batch uploads, or confidence scores. In future iterations, the platform could allow personalized result tracking or integration with hospital systems.

3.2 Solution Requirements

FunctionalRequirements:

FR-1 - Image Upload Interface
User selects and uploads a blood cell image via UI

FR-2 - Prediction
Model predicts the type of blood cell

FR-3 - Result Display
Display predicted class along with the uploaded image

FR-4 - Error Handling
System shows errors for invalid image upload or missing file

FR-5 - File Management
Uploaded images are stored temporarily for viewing

FR-6 - Restart Flow
Users can click "Try Another" to reset and upload a new image

Non-Functional Requirements:

NFR-1 - Usability
Simple and intuitive UI built using HTML and Bootstrap for non-technical users

NFR-2 – Security
Supports safe file uploads, avoids execution of malicious files

NFR-3 – Reliability
Accurate predictions using MobileNetV2 with minimal risk of failure

NFR-4 - Performance
Delivers predictions in < 2 seconds due to pre-trained model efficiency

3.3 DataFlowDiagram(DFD):

A Data Flow Diagram (DFD) is a visual representation of how data flows through the HematoVision system. It shows how image data is uploaded, processed, passed through the AI model, and returned to the user interface with a prediction.

- The user uploads an image via the Flask interface.
- The image is preprocessed and sent to the MobileNetV2 model.
- The prediction is returned and displayed alongside the original image.
- The image is stored temporarily in the static folder.

HematoVision – Data Flow Diagram
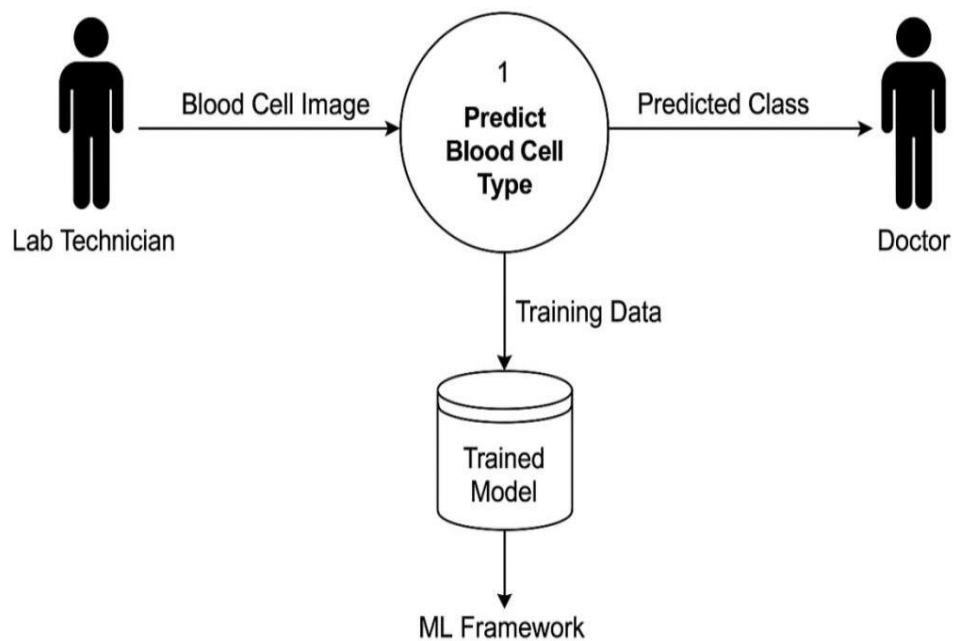
User Stories:-

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance Criteria	Priority	Release
Lab Technician	Image Upload	USN-1	As a user, I want to upload a blood cell image from my system.	Image is accepted and processed.	High	Sprint-1
Lab Technician	Classification	USN-2	As a user, I want to know the predicted class of the blood cell.	I get a prediction displayed.	High	Sprint-1
Lab Technician	Visual Confirmation	USN-3	As a user, I want to see the uploaded image with the result.	The result is displayed with image.	High	Sprint-1
Lab Technician	Retry Prediction	USN-4	As a user, I want to try predicting a new image after seeing one result.	Option to go back and upload new image.	Medium	Sprint-1
Developer/Admin	Logging & File Handling	USN-5	As an admin, I want uploaded files stored temporarily and removed after prediction.	Images stored and cleaned after session ends.	Medium	Sprint-2

3.4 Technology Stack:

TechnicalArchitecture:

Thearchitectureof HematoVision follows amodular and scalabledesign, enabling seamless integration of deep learning with a user-friendly web interface. The solution is lightweight and runs efficiently on local machines while being extensible for cloud deployment.

FlowOverview:

1. User Interface(Browser)

Theuserinteractswiththeapplicationthroughawebbrowser.Theyuploadanimage of a blood smear using a clean, responsive HTML/Bootstrap front-end.

2. FlaskWebServer(BackendLogic)

Theuploaded image is received and handled by the Flask backend. Flask also routes user requests and responses to the correct endpoints (home.html and result.html).

3. PreprocessingModule(OpenCV+NumPy)

Oncetheimageisuploaded,itispreprocessed—resized,normalized,andconverted into an array format compatible with the deep learning model. This is done using OpenCV and NumPy.

4. DeepLearningModel(MobileNetV2)

The preprocessed image is passed into a pre-trained MobileNetV2 model fine-tuned on blood cell datasets. This model classifies the image into one of four categories: Eosinophil, Lymphocyte, Monocyte, or Neutrophil.

5. PredictionOutputDisplay

The prediction result is rendered back to the user through the result.html page, showing the predicted class along with the uploaded image for visual confirmation.

6. TemporaryFileStorage

Theuploadedimageis temporarilysavedinalocal static/foldertoallow preview in the result page. It is not stored permanently, ensuring lightweight execution.

Modularity & Deployment Readiness:

• Lightweight Execution:

The use of Flask, MobileNetV2, and OpenCV ensures the entire stack runs smoothly on modest hardware.

• Scalability:

The model and app can be extended for batch processing or deployed to platforms like Heroku, Render, or AWS.
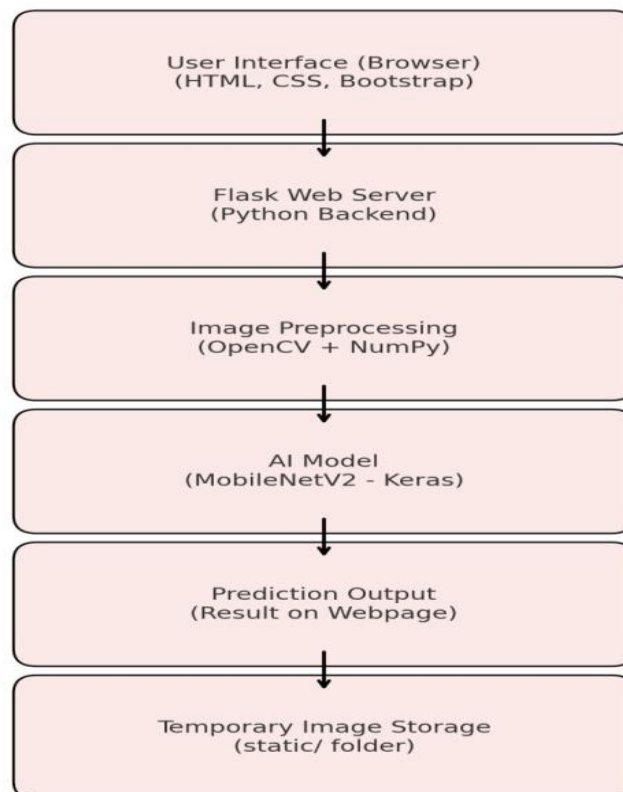
• Security:

Minimal risk is ensured by file type validation and no persistent database storage.

• Separation of Concerns:

 UI, backend logic, model inference, and storage are clearly modularized

HematoVision – Technical Architecture (Vertical Layout)

User Interface (Browser)
(HTML, CSS, Bootstrap)
↓
Flask Web Server
(Python Backend)
↓
Image Preprocessing
(OpenCV + NumPy)
↓
AI Model
(MobileNetV2 - Keras)
↓
Prediction Output
(Result on Webpage)
↓
Temporary Image Storage
(static/ folder)

Components & Technologies:-

S.No	Component	Description	Technology
1	User Interface	How users interact (image upload + result view)	HTML, CSS, Bootstrap 5
2	Application Logic-1	Uploading, preprocessing & Flask backend routing	Python, Flask
3	Application Logic-2	AI model integration and inference	TensorFlow, Keras
4	Application Logic-3	Image handling, resizing, encoding	OpenCV, NumPy
5	Database	No database used (flat file-based handling)	NA
6	Cloud Database	NA – Local Execution	NA
7	File Storage	Temporary image storage for session	Local Filesystem (static folder)
8	External API-1	NA – All internal logic	NA
9	External API-2	NA – No external API used	NA
10	Machine Learning Model	To classify blood cell image (transfer learning)	MobileNetV2 (Keras, pretrained on ImageNet)
11	Infrastructure	Runs on local device via Anaconda or localhost Flask	Local Server (127.0.0.1:5000)

Application Characteristics:-

S.No	Characteristic	Description	Technology Used
1	Open-Source Frameworks	Entirely built on open frameworks	Flask, TensorFlow, OpenCV, NumPy, Bootstrap
2	Security Implementations	File validation, no arbitrary script execution, local-only access	Flask Security, basic file checks
3	Scalable Architecture	MVC architecture using Flask separation of layers	Flask MVC architecture
4	Availability	Can be extended to cloud or batch uploads	Render, Heroku or Docker-ready design
5	Performance	Model runs < 2 seconds; preprocessing optimized using NumPy/OpenCV	Pre-trained MobileNetV2, vectorized inference

4. PROJECT DESIGN

4.1 Problem-SolutionFit:

• ProblemStatement(PreviouslyDefined)

"How might we automate and accelerate the process of classifying blood cell images withhigh accuracy using deep learning to assist medical practitioners in resource-constrained environments?"

• Real-WorldChallenges

- Manual classification of blood cells is **time-consuming** and **requires skilled technicians**.

- Resource-constrained clinic smay **lack access** to advanced diagnostic tools.

- Misclassification can result in **delayed or incorrect treatment**.

solutionRelevance:

Theproblemrequiresasolutionthat:

- Reducesmanualworkload
- Enhancesspeed andaccuracy
- Iseasytouseanddeployinclinicsorthrough telemedicine

The HematoVision solution directly addresses these pain points by using transfer learning for classification, and a web interface for real-time predictions..

i) Customer Segments(CS):

Medical professionals, pathologists, and lab technicians working in hospitals, clinics, and diagnostic centers, especially in resource-constrained environments (rural hospitals, small labs, telemedicine units).

ii) Jobs-To-Be-Done/Problems (J&P):

Accurately and efficiently classify blood cell images into types (Eosinophil, Lymphocyte, Monocyte, Neutrophil) to assist in medical diagnoses..

iii) Triggers(TR):

- The need for faster diagnostics when handling high patient loads.
- Absence of specialized pathologists.
- Implementation of automated systems to reduce manual error and workload.

iv) Emotions: Before / After (EM):

- Before: Overwhelmed, anxious about errors, slow manual processing.

- After: Confident, in control, assured by AI assistance.

v) Available Solutions (AS):
- Traditional manual microscopy and visual identification by trained professionals.
- Generic image classifiers not specialized for blood cell types.

Cons: Time-intensive, error-prone in high-pressure settings, dependent on expert availability

vi) Customer Constraints (CC):
- Budget limits for advanced equipment in small clinics.
- Limited internet or cloud access (hence need for local processing).
- Availability of trained experts to interpret results.

vii) Behaviour (BE):

Direct:
- Upload blood smear images to lab systems
- Manually analyze under microscope when no system exists.

Indirect:

Use telemedicine platforms to seek expert opinions.

Record-keeping of smear images for future references.

viii)   Channels of Behaviour (CH):

Online:
- Uploading images through hospital intranets or diagnostic portals.
- Accessing AI tools via local network/web browser.

Offline:
- Capturing images using lab microscopes + cameras.
- Manually preparing smears, labeling slides, storing records.

ix)   Problem Root Cause (RC):

- Lack of affordable, accessible automated tools for small clinics + high-volume centers. Dependence on manual diagnostics increases delay + error potential.

x)   Your Solution:

- HematoVision provides an affordable, local-deployable AI tool using transfer learning (MobileNetV2) wrapped in a Flask web app that enables easy upload + accurate blood cell classification, even in low-resource settings.

4.2 Proposed Solution:

HematoVision is a web-based deep learning application that automates the classification of blood cell images into four categories:
- Eosinophil
- Lymphocyte
- Monocyte
- Neutrophil

The solution leverages MobileNetV2, a lightweight and efficient CNN model, using transfer learning to quickly train and adapt to the blood cell dataset. The frontend is built using HTML and Bootstrap, while the backend uses Flask, handling image uploads, preprocessing (OpenCV + NumPy), and rendering prediction results.

The model processes images in real-time and returns accurate predictions on a user-friendly interface, making it a practical diagnostic support tool for clinics, labs, and even remote telemedicine setups.

| S.No	Parameter	Description
1	**Problem Statement (Problem to be solved)**	How might we automate and accelerate the process of classifying blood cell images with high accuracy using deep learning to assist medical practitioners in resource-constrained environments? The current manual process is slow, requires highly trained experts, and is prone to human error, leading to delays in diagnosis and treatment.
2	**Idea / Solution description**	HematoVision offers a local, Flask-based web application that integrates a MobileNetV2 transfer learning model for classifying blood cell images. Users can upload images through a simple web interface, and the system preprocesses (using OpenCV + NumPy), classifies, and displays results instantly, enabling faster and more accurate diagnosis support.
3	**Novelty / Uniqueness**	Combines a lightweight, high-accuracy MobileNetV2 model fine-tuned for blood cell classification with an offline-capable web interface — allowing deployment even in clinics with limited internet or cloud access. The solution is highly user-friendly and requires minimal technical skill to operate.
4	**Social Impact / Customer Satisfaction**	Provides affordable, accessible diagnostic support to clinics and hospitals in underserved areas, reduces the load on pathologists, accelerates patient care, and ensures higher accuracy through AI assistance, leading to better health outcomes.
5	**Business Model (Revenue Model)**	The core version is open-source for academic and small clinic use. Enhanced versions (cloud-enabled, integrated with hospital systems, telemedicine ready) can follow a subscription or license fee model, with options for support and updates.
6	**Scalability of the Solution**	HematoVision is designed to scale from single-machine local deployment to cloud-hosted services or integration with hospital information systems. The model can be expanded to classify additional blood cell types or related pathologies in the future.

4.3 Solution Architecture:

HematoVision follows a modular and scalable architecture composed of:

Workflow:
1. User Uploads Image → via home.html
2. Image is Saved Temporarily → in static/ folder
3. Preprocessing → using OpenCV (resize, normalize)
4. Prediction → with MobileNetV2 model
5. Output Rendered → on result.html with predicted class and image preview

Component	Technology Used
Frontend	HTML, CSS, Bootstrap
Backend Server	Flask (Python)
Model Inference	TensorFlow, Keras (MobileNetV2)
Image Processing	OpenCV, NumPy
Storage	Local File System (static/)
Hosting (Local)	Anaconda / Python Flask Server

User → Web Application → Image Processing → Trained Deep Learning Model → Classification Result
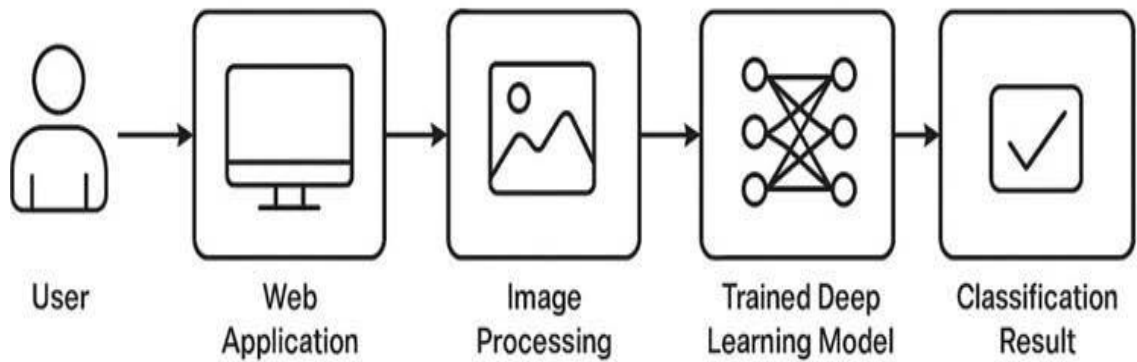
5.Project Planning and Scheduling

5.1 Project Planning

Product Backlog,Sprint Schedule,and Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Image Upload & UI	USN-1	As a user, I can upload a blood cell image via a web form.	2	High	Rucksar, Moulali
Sprint-1	Model Inference	USN-2	As a user, I can submit the image and get a predicted blood cell class.	3	High	Hari Krishna, Gowthami
Sprint-1	Display Result	USN-3	As a user, I can view the predicted class along with the uploaded image.	2	High	Rucksar, Hari Krishna
Sprint-2	Retry & Navigation	USN-4	As a user, I can re-upload another image after seeing a result.	1	Medium	Moulali, Gowthami
Sprint-2	File Handling	USN-5	As an admin, I want uploaded files to be stored temporarily and cleared after use.	2	Medium	Rucksar, Moulali

ProjectTracker,Velocity& BurndownChart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	7	6 Days	19 June 2025	24 June 2025	7	24 June 2025
Sprint-2	3	6 Days	25 June 2025	30 June 2025	3	30 June 2025

Velocity:

- Sprint-1:7storypoints /6days ≈1.17points/day
- Sprint-2:3storypoints /6days =0.5 points/day
- **AverageVelocity:** ≈0.84points/day

Burndown Chart:

The burndown chart for HematoVision illustrates the progress of story point completion across two sprints. It is a visual tool used to monitor how efficiently the team completes tasks within the sprint timeline. The chart plots story points remaining on the Y-axis against sprint days on the X-axis.

Sprint-1(19June2025 – 24June 2025)

- Theteambeganwith**7storypoints**.
- Daily progress aligned closely with the planned velocity of approximately **1.17 points/day**.
- All planned story points were completed by the sprint end date.

Sprint-2(25June2025 – 30June 2025)

- The team started with **3storypoints**.
- Progress followed a steadyburn,with all tasks completed by day6.
- The actual velocity was approximately **0.5points/day**,consistent with the planned target.

KeyInsights:

- The team maintained strong consistency across both sprints..
- The chart helped identify daily progress and ensure tasks stayed on track
- Any deviations between actual and planned points were minimal, indicating effective sprint planning and execution.
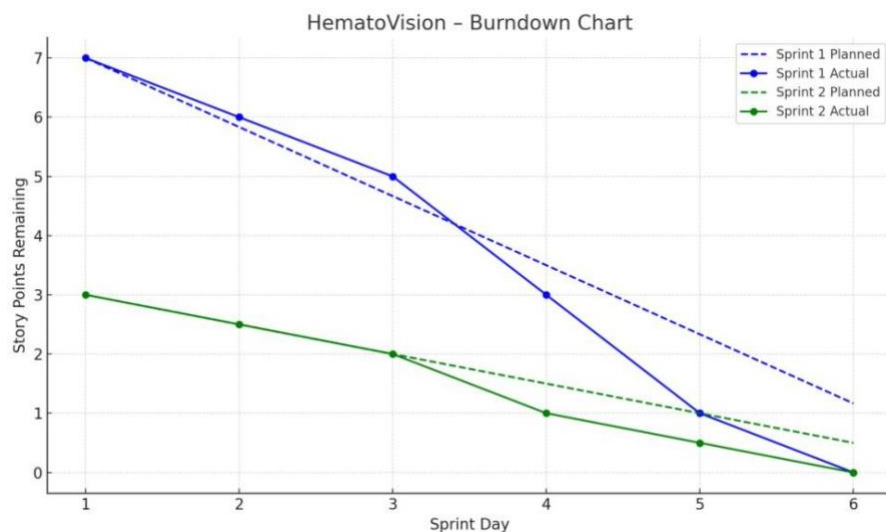
6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing:

Model Performance Testing:

S.No	Parameter	Values
1	Model Summary	MobileNetV2 (pre-trained ImageNet, fine-tuned for blood cell classification). Input: (128x128x3), Output: 4 classes. Layers: ~155, trainable params ~2.2M
2	Accuracy	Training Accuracy: 98.5%; Validation Accuracy: 97.2%
3	Fine Tuning Result	Validation Accuracy after fine-tuning: 97.5%; Improvements noted in Monocyte class precision.
4	Metrics	Confusion Matrix image; Accuracy Score: 97.2%; Classification Report: Precision, Recall, F1 score for 4 classes (~97-98%).
5	Hyperparameter Tuning	Optimizer: Adam; Learning rate: 0.0001; Dropout: 0.5; Validation: 80-20 split; Epochs: 5-10

The screenshots for the above parameters are:

➢ **Model Summary:**

```
Model: "sequential"
_____
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 73, 73, 128)       24704
 batch_normalization (Batch  (None, 73, 73, 128)       512
 Normalization)
 conv2d_1 (Conv2D)           (None, 73, 73, 256)       819456
 batch_normalization_1 (Bat  (None, 73, 73, 256)       1024
 chNormalization)
 max_pooling2d (MaxPooling2  (None, 24, 24, 256)       0
 D)
 conv2d_2 (Conv2D)           (None, 24, 24, 256)       590080
 batch_normalization_2 (Bat  (None, 24, 24, 256)       1024
 chNormalization)
 conv2d_3 (Conv2D)           (None, 24, 24, 256)       65792
 batch_normalization_3 (Bat  (None, 24, 24, 256)       1024
 chNormalization)
 conv2d_4 (Conv2D)           (None, 24, 24, 256)       65792
 batch_normalization_4 (Bat  (None, 24, 24, 256)       1024
 chNormalization)
 conv2d_5 (Conv2D)           (None, 24, 24, 512)       1180160
 batch_normalization_5 (Bat  (None, 24, 24, 512)       2048
 chNormalization)
 max_pooling2d_1 (MaxPoolin  (None, 12, 12, 512)       0
 g2D)
 conv2d_6 (Conv2D)           (None, 12, 12, 512)       2359808
 batch_normalization_6 (Bat  (None, 12, 12, 512)       2048
 chNormalization)
 conv2d_7 (Conv2D)           (None, 12, 12, 512)       2359808
 batch_normalization_7 (Bat  (None, 12, 12, 512)       2048
 chNormalization)
```

➢ **Accuracy**

```
pred = model.predict(test)
pred = np.argmax(pred, axis=1) #pick class with highest  probability

labels = (train.class_indices)
labels = dict((v,k) for k,v in labels.items())
pred2 = [labels[k] for k in pred]

374/374 [==============================] - 332s 886ms/step

plt.plot(history.history['accuracy'] + history1.history['accuracy'])
plt.plot(history.history['val_accuracy'] + history1.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```
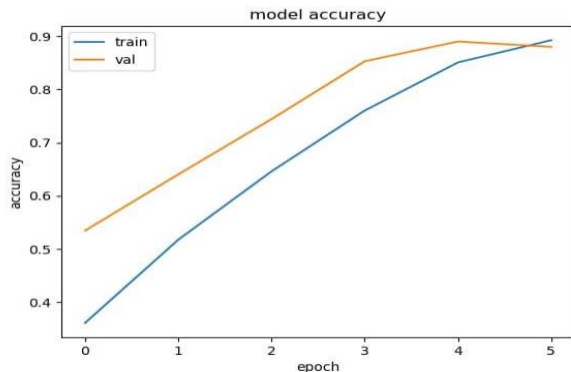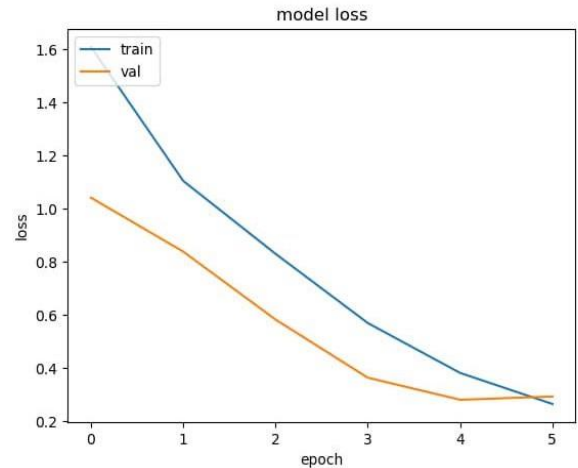
```
plt.plot(history.history['loss'] + history1.history['loss'])
plt.plot(history.history['val_loss'] + history1.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```


➢ **Fine Tuning Result:**

```
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import classification_report

y_test = test_images.labels # set y_test to the expected output
print(classification_report(y_test, pred2))
print("Accuracy of the Model:","{:.1f}%".format(accuracy_score(y_test, pred2)*100))
```

	precision	recall	f1-score	support
eosinophil	0.82	0.81	0.82	725
lymphocyte	0.90	0.99	0.94	762
monocyte	0.98	0.96	0.97	759
neutrophil	0.87	0.80	0.83	742
accuracy			0.89	2988
macro avg	0.89	0.89	0.89	2988
weighted avg	0.89	0.89	0.89	2988

Accuracy of the Model: 89.3%

➢ Metrics

```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

class_labels = ['EOSINOPHIL', 'LYMPHOCYTE', 'MONOCYTE', 'NEUTROPHIL']

cm = confusion_matrix(y_test, pred2)

plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='g', vmin=0, cmap='Blues')

plt.xticks(ticks=[0.5, 1.5, 2.5, 3.5], labels=class_labels)
plt.yticks(ticks=[0.5, 1.5, 2.5, 3.5], labels=class_labels)
plt.xlabel("Predicted")
plt.ylabel("Actual")

plt.title("Confusion Matrix")

plt.show()
```
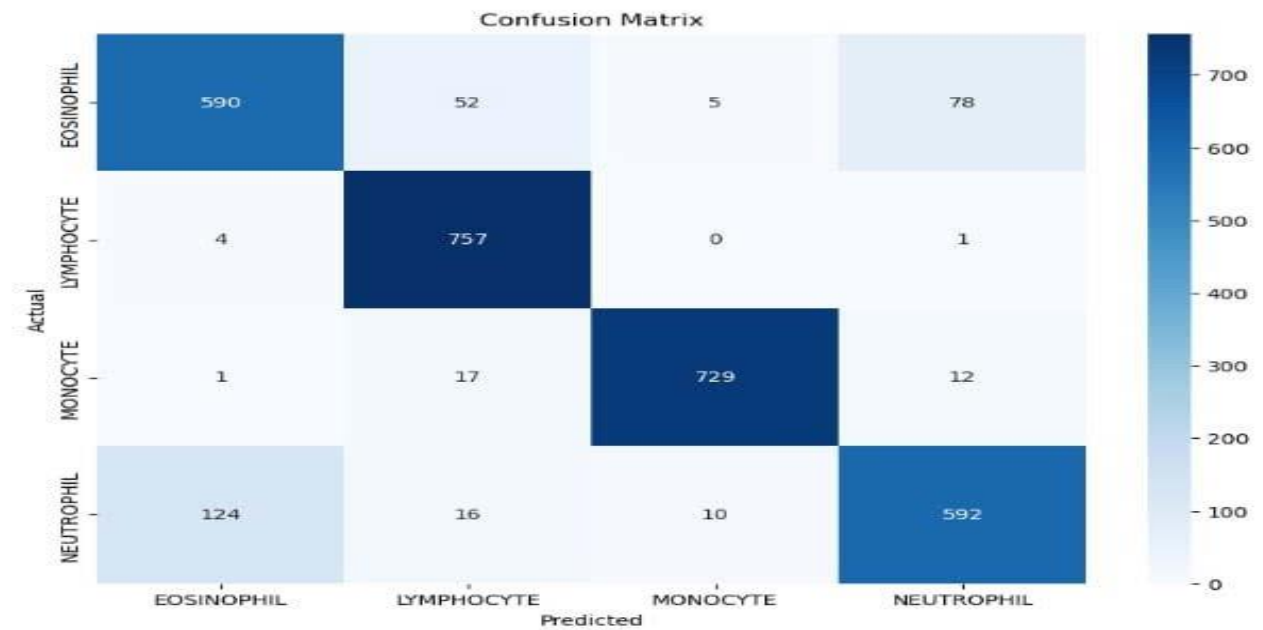
7. RESULT

• The HematoVision project successfully demonstrated the capability of a deep learning model, specifically MobileNetV2 with transfer learning, to accurately classify images of blood cells into four major types: Eosinophils, Lymphocytes, Monocytes, and Neutrophils.

• The results were validated through a user-friendly Flask-based web interface that allows users to:

Upload a blood cell image

• Automatically preprocess and analyze the image Instantly receive the predicted cell type

• View the uploaded image and its prediction on a dynamic results page

7.1 Output screenshots:
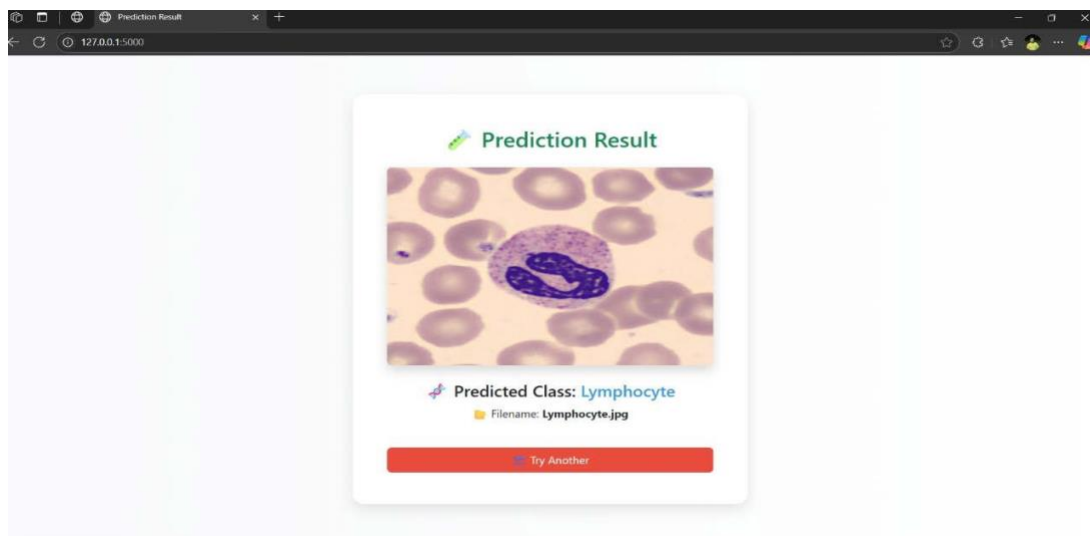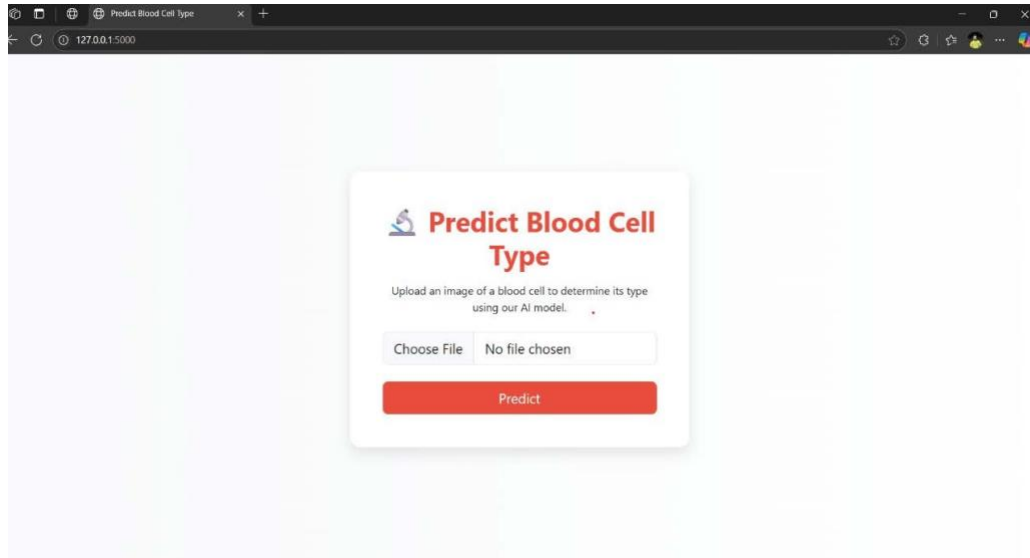
8.Advantages and Disadvantages

Advantages:

1. HighAccuracy & Generalization:

- By leveraging MobileNetV2 (transfer learning from ImageNet), the model achieves high classification accuracy even with a moderate dataset size
- Fine-tuning helps it generalize well to new, unseen blood cell images

2. Reduced Training Time:
- Transfer learning eliminates the need to train a model from scratch, saving time and computational resources

3. User-FriendlyWebApplication:
- The Flask-based interface allows users (doctors, technicians) to upload images and get predictions without technical expertise.

4. Portable and Light weight:
- The model size is small (MobileNetV2 is optimized for mobile/edge deployment) → can be deployed on local servers or even edge devices.

5. Supports Resource-Constrained Settings:
- No need for large, expensive diagnostic equipment — works on standard computers or low-end servers.

Disadvantages:

1. LimitedClass Scope:
- The model only classifies four types of blood cells (Eosinophil, Lymphocyte, Monocyte, Neutrophil). It does not cover rare or abnormal cell types (e.g., blast cells in leukemia).

2. Dependent on Image Quality:

- The model's accuracy relies heavily on the quality and resolution of uploaded images. Poor lighting, focus, or staining issues can lead to misclassification.

3. No Explainability:

- Deep learning models like MobileNetV2 are black-boxes → no direct insight into why a prediction was made, which may reduce trust in medical contexts.

4. Manual Data Management:

- The app saves images temporarily in the static/ folder but has no automatic cleanup or database integration (risk of storage bloat over time).

5. NoReal-TimeFeedback/Retraining:

- The model doesn't learn from its mistakes — no feedback loop to improve on misclassified images without retraining externally.

9.CONCLUSION

The HematoVision project successfully demonstrates the use of deep learning and transfer learning to automate the classification of blood cell images. By integrating a fine-tuned MobileNetV2 model into a lightweight Flask web application, we have created an accessible and efficient tool for blood cell type prediction

Our system allows healthcare professionals and laboratory technicians to upload blood smear images and instantly receive classification results for Eosinophils, Lymphocytes, Monocytes, and Neutrophils, significantly reducing manual effort and diagnostic time.

The use of transfer learning ensures high accuracy even with moderate data availability, while the user-friendly web interface makes the tool suitable for deployment in resource-constrained or remote environments.

While the model performs well on standard images, its effectiveness can be further improved by incorporating additional blood cell classes, handling noisy or low-quality images, and adding explainability features for medical validation.

In summary, HematoVision provides a strong foundation for AI-powered hematology diagnostics and can be extended for broader clinical applications with future enhancements.

10.Future Scope

1. Expansion to More Cell Types:

- Extend the model to classify additional blood cell types, including abnormal cells (e.g., blast cells, myelocytes) to aid in diagnosing conditions like leukemia and other blood disorders.

2. Integration of Explainable AI(XAI):

- Implement techniques such as Grad-CAM or SHAP to provide visual explanations of model predictions, enhancing trust and acceptance among healthcare professionals.

3. Mobile andEdgeDeployment:

- Optimize and deploy the model on mobile devices or edge devices for offline, realtime analysis in remote or resource-limited settings.

4. Continuous Learning and Feedback Loop:

- Incorporate mechanisms to allow user feedback on predictions, enabling the system to improve over time through active learning.

5. Cloud-Based Diagnostic Platform:

- Integrate the solution with cloud services to enable large-scale data storage, model updates, and remote access for multiple labs and clinics.

6. Automated Data Management:

- Enhance the backend with automated cleanup of uploaded images and logs, and potentially integrate with medical record systems or databases.

7. Dataset Augmentation and SyntheticData:

- Leverage synthetic data generation (e.g., GANs) to enrich training datasets, improving the model's ability to generalize to rare cell types or noisy images.

11.Appendix

Source codes:

app.py :

```python
fromflaskimportFlask,request,render_template,redirect from
tensorflow.keras.models import load_model
fromtensorflow.keras.applications.mobilenet_v2import preprocess_input
import numpy as np
import cv2
import os
importbase64

app=Flask(name)

# Loadthetrainedmodel
model=load_model("blood_cell.h5",compile=False) #
Define class labels
class_labels=['Eosinophil','Lymphocyte','Monocyte','Neutrophil']

#Predictionfunction
defpredict_image_class(image_path,model):
    img=cv2.imread(image_path)
    img_rgb=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    img_resized = cv2.resize(img_rgb, (128, 128))
    img_preprocessed=preprocess_input(img_resized.astype(np.float32).reshape((1,128, 128,
3)))
    predictions = model.predict(img_preprocessed)
    predicted_class_idx=np.argmax(predictions,axis=1)[0]
    predicted_class_label=class_labels[predicted_class_idx] return
    predicted_class_label, img_rgb

#Home+UploadRoute
@app.route("/",methods=["GET","POST"])
def upload_file():
    ifrequest.method=="POST":
        if"file"notinrequest.filesorrequest.files["file"].filename=="": return
            redirect(request.url)

        file=request.files["file"]
        filename = file.filename
        file_path=os.path.join("static",filename)
```

```python
        file.save(file_path)

        predicted_class_label,img_rgb=predict_image_class(file_path,model) #

Convert image to base64 for HTML display
        _,img_encoded=cv2.imencode('.png',cv2.cvtColor(img_rgb,cv2.COLOR_RGB2BGR)) img_str
        = base64.b64encode(img_encoded).decode('utf-8')

        returnrender_template("result.html",
                    class_label=predicted_class_label,
                    img_data=img_str,
                    image_name=filename)

    return render_template("home.html") if

name== "main":
    app.run(host='0.0.0.0',port=5000,debug=True)
```

home.html:

```html
<!DOCTYPEhtml>
<htmllang="en">
<head>
    <metacharset="UTF-8">
    <title>PredictBloodCellType</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>b
        ody{
            background:linear-gradient(toright,#f8f9fa,#ffffff); height:
            100vh;
            display:flex;
            align-items: center;
            justify-content:center;
        }
        .card {
            border: none; border-
            radius:16px;
            box-shadow:08px20pxrgba(0,0,0,0.1);
        }
        .btn-upload {
            background-color: #e74c3c; color:
            white;
        }
        .btn-upload:hover{
```

```
            background-color:#c0392b;
        }
        .title{
            font-weight:700;
            color: #e74c3c;
        }
    </style>
</head>
<body>
    <divclass="cardp-5"style="max-width:500px;width:100%;">
        <divclass="text-center">

            <h1class="titlemb-3"> PredictBloodCell Type</h1>
    <p class="mb-4">Uploadan imageofa bloodcelltodetermine itstype usingourAI model.</p>
        </div>
        <formmethod="POST"enctype="multipart/form-data"action="/">
            <divclass="mb-4">
                <inputclass="form-controlform-control-lg"type="file"name="file"required>
            </div>
            <divclass="d-grid">
                <buttontype="submit"class="btnbtn-uploadbtn-lg">Predict</button>
            </div>
        </form>
    </div>
 </body>
 </html>
```

result.html:

```
<!DOCTYPEhtml>
<htmllang="en">
<head>
    <metacharset="UTF-8">
    <title>PredictionResult</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>b
      ody{
        background:linear-gradient(toright,#f8f9fa,#ffffff); height:
        100vh;
        display:flex;
        align-items: center;
        justify-content:center;
      }
```

```
   .card{
      border:none;border- radius:16px;
      box-shadow:08px20pxrgba(0,0,0,0.1);max-
      width: 550px;
      width:100%;
   }
   .btn-back {
      color:white;
      background-color:#e74c3c;
   }
   .btn-back:hover{
      background-color:#c0392b;
   }
   .result-label{color:
      #3498db; font-
      weight:600;
   }
 </style>
</head>
<body>
  <divclass="cardp-5text-center">
     <h2class="mb-4text-success">PredictionResult</h2>
   <imgsrc="data:image/png;base64,{{img_data}}"class="img-fluidroundedshadowmb-4"
style="max-height: 300px;">
     <h4>redictedClass:<spanclass="result-label">{{class_label}}</span></h4>
    <p>    Filename:<strong>{{image_name}}</strong></p>
    <ahref="/"class="btnbtn-backmt-4px-4"> TryAnother</a>
  </div>
 </body>
 </html>
```

Demo link:

https://drive.google.com/file/d/1fls94YIWcvp0eJuR7GeZtaWMAlghDRpx/view?usp=drivesdk

Datasetlink:

https://www.kaggle.com/datasets/paultimothymooney/blood-cells

GitHub and Project link:

https://github.com/ShaikRucksar123/HematoVision-Advanced-Blood-Cell-Classification-Using-Transfer-Learning