

# PROJECT\_SMART\_INTERNZ

## PROJECT TITLE:

**HematoVision: Advanced Blood Cell Classification Using  
Transfer Learning**

**TEAM ID : LTVIP2025TMID34658**

- ❖ Kondaveeti Manikanta Deepika (Team Leader)
- ❖ Nalla Naga Durga
- ❖ Mamidisetti Subrahmanyam
- ❖ Tanuku Bala Charan Teja

Google Drive Link:

Project\_Demo :

<https://drive.google.com/file/d/1CqX0mxl9w7ByJ4aUn9sl9871SpCs63qq/view?usp=drivesdk>

GitHub Link:

[https://github.com/subrahmanyam9617196/HematoVision\\_Advanced-Blood-Cell-Classification-Using-Transfer-Learning](https://github.com/subrahmanyam9617196/HematoVision_Advanced-Blood-Cell-Classification-Using-Transfer-Learning)

# **HematoVision: Advanced Blood Cell Classification Using Transfer Learning**

**Category:** Artificial Intelligence

**Skills Required:** Python, Deep Learning, Transfer Learning

## **Project Description:**

HematoVision aims to develop an accurate and efficient model for classifying blood cells by employing transfer learning techniques. Utilizing a dataset of 12,000 annotated blood cell images, categorized into distinct classes such as eosinophils, lymphocytes, monocytes, and neutrophils, the project leverages pre-trained convolutional neural networks (CNNs) to expedite training and improve classification accuracy. Transfer learning allows the model to benefit from pre-existing knowledge of image features, significantly enhancing its performance and reducing computational costs. This approach provides a reliable and scalable tool for pathologists and healthcare professionals, ensuring precise and efficient blood cell classification.

### **Scenario 1: Automated Diagnostic Systems for Healthcare**

Integrating HematoVision into automated diagnostic systems in clinical settings can revolutionize blood analysis. By using transfer learning, the system quickly adapts to the specifics of blood cell classification, capturing images of blood samples, classifying the cells in real-time, and generating detailed reports. This automation reduces the manual workload on pathologists, speeds up diagnostic processes, and ensures high accuracy in results, ultimately improving patient care and treatment efficiency.

### **Scenario 2: Remote Medical Consultations**

HematoVision can be employed in telemedicine platforms to enhance remote consultations and diagnostics. With transfer learning, the model's ability to accurately classify blood cells from diverse sources is improved, allowing healthcare providers to upload blood cell images for automated analysis. This enables timely and accurate assessments without the need for in-person visits, facilitating better access to specialized medical expertise and improving healthcare delivery in remote or underserved areas.

### **Scenario 3: Educational Tools for Medical Training**

HematoVision's transfer learning-based classification model can be integrated into educational tools for medical training. By incorporating this advanced technology into interactive learning platforms, students and laboratory technicians can upload and analyze blood cell images to receive instant feedback. This hands-on learning experience enhances their understanding of blood cell morphology and classification, providing practical skills and knowledge that are crucial for accurate diagnostic practice and medical training.

#### **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.

##### **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. HematoVision utilizes transfer learning to classify blood cell types from microscopic images accurately. By integrating AI into hematovision system enhances diagnostic speed, precision, and reliability, supporting early disease detection and improves patient outcomes.

##### **Proposed Solution:**

An AI-powered application that uses transfer learning to accurately classify blood cells from microscope images in real-time. It helps doctors by providing quick, reliable results, reducing manual effort and improving the speed and accuracy of diagnosis.

##### **Target Users:**

Pathologists and lab technicians who need quick and accurate blood cell analysis.

Hospitals and diagnostic centers aiming to automate blood smear classification.

Doctors offering remote consultations who require fast diagnostic support.

Medical students and trainees learning about blood cell identification.

### **Expected Outcome:**

A functional AI-powered application that accurately classifies blood cells from images using transfer learning.

Provides fast, reliable diagnostic support to improve efficiency in medical analysis and education.

## **Phase-2: Requirement Analysis**

### **Objective:**

Define the technical and functional requirements for the HematoVision application.

### **Key Points:**

### **Technical Requirements:**

- **ProgrammingLanguage:Python**
- **Python Packages: NumPy, Pandas, Scikit-learn, Matplotlib, SciPy,Seaborn, TensorFlow, Flask**
- **Frameworks: Flask for web integration, TensorFlow for deep learning**
- **Pre-trained Model: VGG16 (used for transfer learning)**
- **DevelopmentTools: Command Line (pip install)**

## Functional Requirements:

- Ability to upload microscopic blood cell images through the web interface.
- Classify blood cells into types like eosinophils, lymphocytes, monocytes, and neutrophils using a trained model.
- Display classification results along with prediction confidence.
- Provide an easy-to-use interface for doctors, students, and lab technicians.

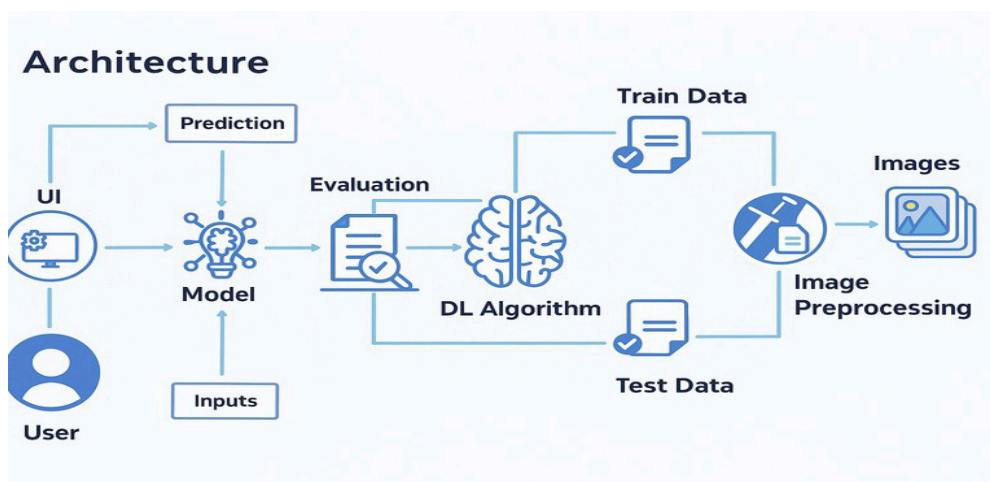
## Constraints & Challenges:

- 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.

## Phase-3: Project Design

### Objective:

Develop the architecture and user flow of the application.



## Key Points:

## System Architecture:

- ★ The user uploads a blood cell image through the web interface (UI).
- ★ The image is sent as input to the trained deep learning model.
- ★ The model uses a pre-trained CNN (like VGG16) to process the image and predict the blood cell type.
- ★ The prediction is evaluated and displayed back on the UI with confidence levels.
- ★ Initially, the model is built using preprocessed images from the dataset, split into training and testing sets.
- ★ The training data is used to fine-tune the DL algorithm using transfer learning, while test data is used for evaluation.

## User Flow:

- ★ Step 1: User opens the web application.
- ★ Step 2: Uploads a microscope image of a blood smear.
- ★ Step 3: The model processes the image and classifies the cell type.
- ★ Step 4: The predicted cell type and confidence score are displayed.
- ★ Step 5: User can repeat with other images or use the results for diagnostic insight.

## Phase-4: Project Planning

### Objective:

Breakdown development tasks for efficient completion.

### Sprint Planning with Priorities:

#### Sprint 1 - Setup & Preparation (Day 1)

##### ● High Priority

- Set up the environment using Anaconda Navigator.

- Install all required Python packages (TensorFlow, Flask, etc.).
- Collect and preprocess the blood cell image dataset.

## Sprint 2 – Model Development & Integration (Day 2)

### ● High Priority

- Build and train the blood cell classification model using transfer learning (e.g., VGG16).
- Integrate the trained model with the Flask web application.

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

### ● Medium Priority

- Test the app functionality, fix bugs, and improve UI responsiveness.

### ● Low Priority

- Finalize deployment and prepare presentation/demo materials.

Sprint	Task	Priority	Duration	Deadline	Assigned to	Dependencies	Expected outcome
Sprint 1	Environment Setup & Package Installation	● High	3hours	Day 1	Member 1	Anaconda , Python	Project environment ready
Sprint 1	Dataset Collection & Preprocessing	● High	4hours	Day 1	Member 2	Dataset access	Clean,prepared image dataset
Sprint 2	Model Building using Transfer Learning	● High	5hours	Day 2	Member 3	Preprocessed data, TensorFlow	Trained classification model
Sprint 2	Flask Web App Integration	● Medium	3hours	Day 2	Member 1 & 4	Trained Model,Flask installed	Working web interface
Sprint 3	Testing & Debugging	● Medium	2 hours	Day 2	Member 2 & 3	Complete System	Bug-free and responsive system
Sprint 3	Final Presentation & Deployment	● Low	1hour	End of Day 2	Entire Team	Working application	Project deployed and demo-ready

## **Phase-5: Project Development**

### **Objective:**

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

### **Key Points:**

#### **Technology Stack Used:**

- **Frontend:** HTML (via Flask templates)
- **Backend:** Flask Framework
- **Deep Learning:** TensorFlow with pre-trained VGG16 model
- **Programming Language:** Python

#### **Development Process:**

- **Built and trained a blood cell classification model using transfer learning (VGG16).**
- **Preprocessed a dataset of 12,000 labeled blood cell images.**
- **Integrated the trained model into a Flask web application.**
- **Developed a user interface to upload images and display classification results with prediction confidence.**

#### **Challenges & Fixes:**

- **Challenge:** Model overfitting on certain blood cell types.
- **Fix:** Used data augmentation and dropout regularization.
- **Challenge:** Large model size caused slow predictions.
- **Fix:** Applied model optimization and saved in.h5 format for faster loading.
- **Challenge:** Image quality variation.
- **Fix:** Added preprocessing steps like resizing and normalization before prediction.



## 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 mixed WBC/RBC image	Multi-class classification displayed correctly	✓ Passed	Tester 2
TC-003	Performance Testing	Check model response time	Results displayed under 2 secods	⚠ Needs Optimization	Tester 3
TC-004	Bug Fix Validation	Image with poor lighting	System still makes reasonable prediction	✓ Fixed	Developer
TC-005	UI Responsiveness	Test on mobile browser	Layout adjusts properly on mobile	✗ Failed	Tester 2
TC-006	Deployment Testing	Hosted on local server and accessed remotely	App loads and predicts successfully online	🚀 Deployed	DevOps

# Phase-7: Predict & Output



## Project Overview: HematoVision – Blood Cell Classification

```
1 <!-- templates/home.html -->
2 <!DOCTYPE html>
3 <html lang="en">
4 <head>
5   <meta charset="UTF-8">
6   <title>Blood cell Classification</title>
7   <style>
8     body {
9       font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
10      background: linear-gradient(135deg, #e0f7fa, #e0e0e0);
11      color: #333;
12      display: flex;
13      justify-content: center;
14      align-items: center;
15      height: 100vh;
16      margin: 0;
17    }
18
19    .upload-box {
20      background: #ffffff;
21      border-radius: 16px;
22      box-shadow: 0 12px 28px #000000;
23      padding: 40px 50px;
24      text-align: center;
25      transition: transform 0.3s ease;
26      border: 1px solid #e0e0e0;
27    }
28
29    .upload-box:hover {
30      transform: scale(1.015);
31    }
32  </style>
33 </head>
34 <body>
35 </body>
36 </html>
```

```
1 import os
2 import numpy as np
3 import cv2
4 from flask import Flask, request, render_template, redirect, url_for
5 from tensorflow.keras.models import load_model
6 from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
7 import matplotlib.pyplot as plt
8 import io
9 import base64
10 app = Flask(__name__)
11 model = load_model("Blood Cell.h5")
12 class_labels = ['eosinophil', 'lymphocyte', 'monocyte', 'neutrophil']
13
14 def predict_image_class(image_path, model):
15     img = cv2.imread(image_path)
16     img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
17     img_resized = cv2.resize(img_rgb, (224, 224))
18     img_preprocessed = preprocess_input(img_resized.reshape((1, 224, 224, 3)))
19     predictions = model.predict(img_preprocessed)
20     predicted_class_idx = np.argmax(predictions, axis=-1)[0]
21     predicted_class_label = class_labels[predicted_class_idx]
22     return predicted_class_label, img_rgb
23
24 @app.route("/", methods=["GET", "POST"])
25 def upload_file():
26     if request.method == "POST":
27         if "file" not in request.files:
28             return redirect(request.url)
29         file = request.files["file"]
30         if file.filename == "":
31             return redirect(request.url)
32         if file:
```

```
25 def upload_file():
26     file_path = os.path.join("static", file.filename)
27     file.save(file_path)
28     predicted_class_label, img_rgb = predict_image_class(file_path, model)
29
30     _, img_encoded = cv2.imencode('.png', cv2.cvtColor(img_rgb, cv2.COLOR_RGB2BGR))
31     img_str = base64.b64encode(img_encoded).decode('utf-8')
32
33     return render_template("result.html", class_label=predicted_class_label, img_data=img_str)
34
35     return render_template("home.html")
36
37 if __name__ == "__main__":
38     port = int(os.environ.get("PORT", 5000))
39     app.run(debug=True, host="0.0.0.0", port=port)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

INFO:werkzeug: \* Running on http://127.0.0.1:5000

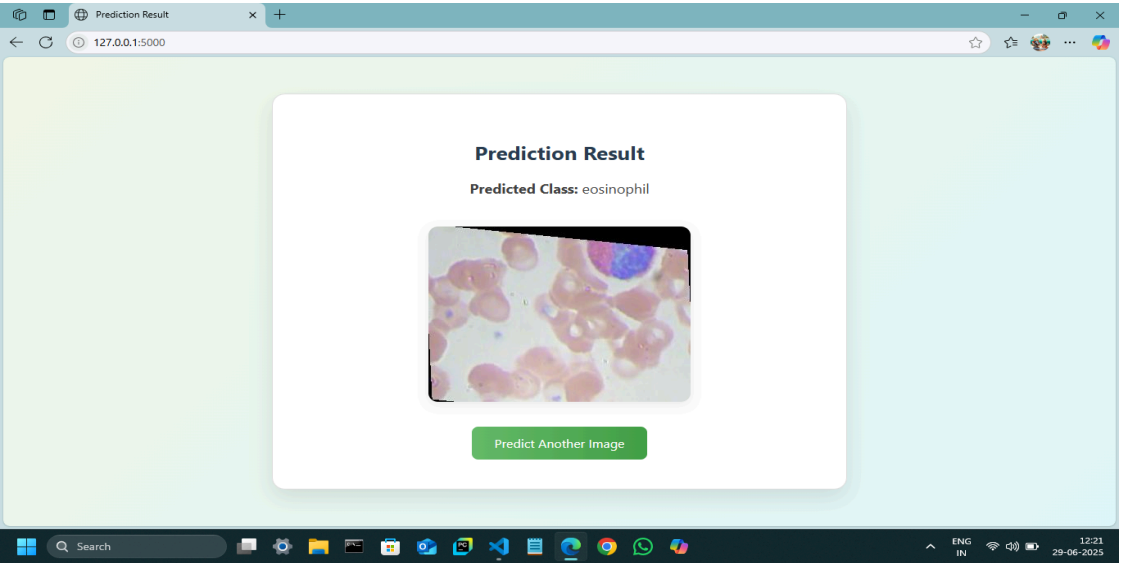
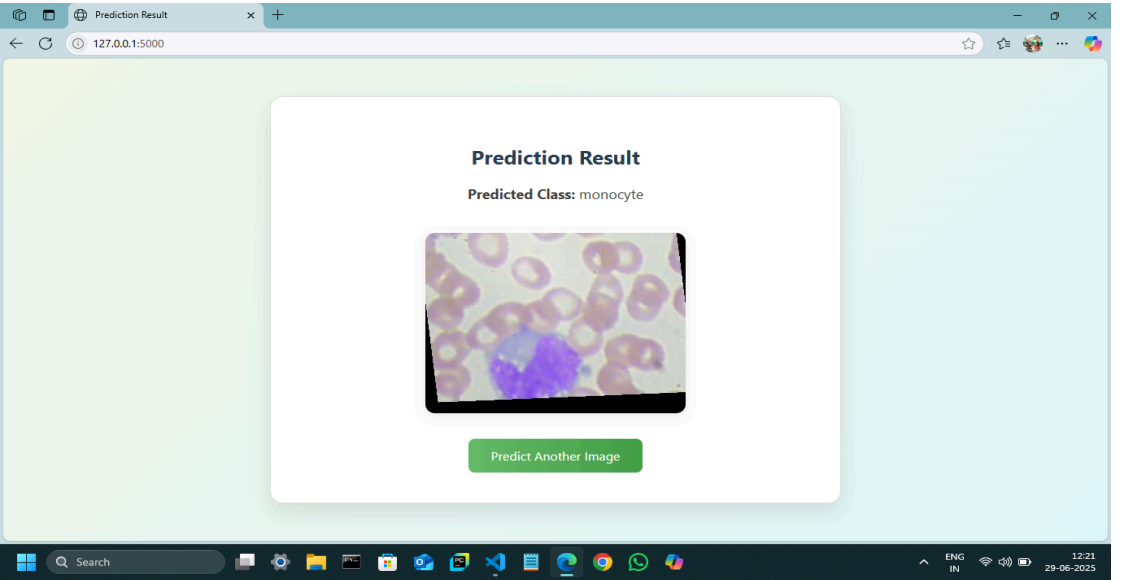
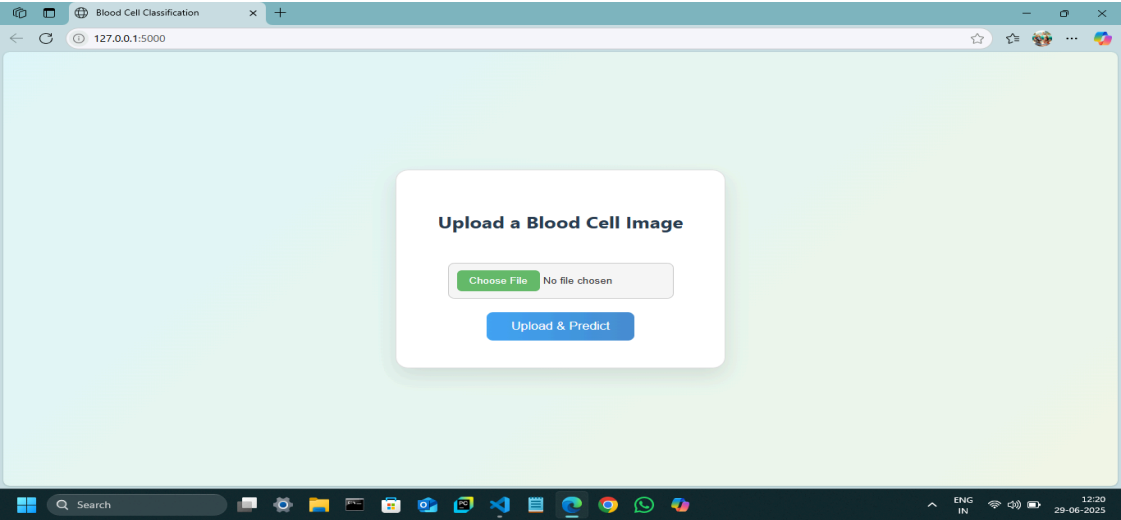
INFO:werkzeug: Press CTRL+C to quit

INFO:werkzeug: \* Restarting with stat

2025-06-29 12:18:59.243296: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF\_ENABLE\_ONEDNN\_OPTS=0'.

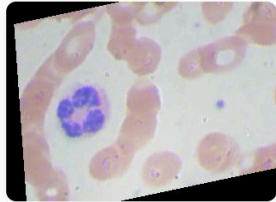
2025-06-29 12:18:55.746567: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF\_ENABLE\_ONEDNN\_OPTS=0'.

2025-06-29 12:19:17.843418: I tensorflow/core/platform/cpu\_feature\_guard.cc:218] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.



Prediction Result

Predicted Class: neutrophil



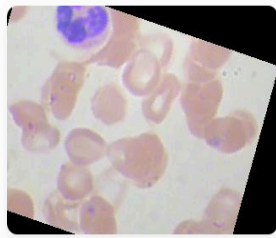
Predict Another Image

Search

ENG IN 12:21 29-06-2025

Prediction Result

Predicted Class: lymphocyte



Predict Another Image

Search

ENG IN 12:22 29-06-2025