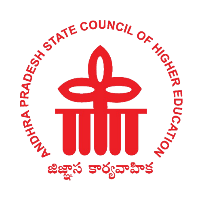
**APSCHE-SmartInternz**

**Project Report**

****

**HematoVision:**

**Advanced Blood Cell Classification Using Transfer Learning**

SUBMITTED BY: Rupesh Kumar Sah (23P31A42C9)

**Srijan Singh** (23P31A42D7)

**Shaik Mohammed Mahaboob Alisha** (23P31A42D4)

**Tholeti Sridivya** (23P31A42D9)

**Track: Artificial Intelligence & Machine Learning**

**Team Id: LTVIP2025TMID33453**

**1. INTRODUCTION**

1.1 Project Overview

1.2 Purpose

**2. IDEATION PHASE**

2.1 Problem Statement

2.2 Empathy Map Canvas

2.3 Brainstorming

**3. REQUIREMENT ANALYSIS**

3.1 Customer Journey map

3.2 Solution Requirement

3.3 Data Flow Diagram

3.4 Technology Stack

**4. PROJECT DESIGN**

4.1 Problem Solution Fit

4.2 Proposed Solution

4.3 Solution Architecture

**5. PROJECT PLANNING & SCHEDULING**

5.1 Project Planning

**6. FUNCTIONAL AND PERFORMANCE TESTING**

6.1 Performance Testing

**7. RESULTS**

7.1 Output Screenshots

**8. ADVANTAGES & DISADVANTAGES**

**9. CONCLUSION**

**10. FUTURE SCOPE**

**11. APPENDIX**

**Source Code**

**Dataset Link**

**GitHub & Project Demo Link**

**1. INTRODUCTION**

**1.1 Project Overview:**

Hematovision is a deep learning-based web application built using Flask that classifies human blood cell images into categories such as Neutrophil, Lymphocyte, Monocyte, and Eosinophil. The application allows users to upload microscope images of blood samples and instantly receive predictions based on a pre-trained CNN model using transfer learning with VGG16. The system is designed to enhance the efficiency of medical diagnostics by automating the cell classification process.

**1.2 Purpose:**

The project aims to assist pathologists and researchers by automating the process of identifying and classifying blood cells, reducing manual workload, and increasing accuracy in diagnosis. This not only saves time but also helps prevent diagnostic errors that could arise due to human fatigue or oversight.

**2. IDEATION PHASE**

**2.1 Problem Statement:**

Manual classification of blood cells under a microscope is time-consuming and prone to human error. There is a need for an automated, accurate system that can rapidly process and classify large volumes of blood cell images with minimal human intervention.

**2.2 Empathy Map Canvas:**

* Think & Feel: Frustration with long diagnosis times, concern over accuracy.
* Hear: Discussions about diagnostic delays and misidentification.
* See: Overworked lab technicians and crowded diagnostic centers.
* Say & Do: Seek faster and reliable diagnostic tools, explore automation.
* Pain: Misdiagnosis due to fatigue or inexperience, repetitive manual work.
* Gain: Instant, accurate classification and improved workflow.

**2.3 Brainstorming:**

* Use deep learning for classification to improve speed and accuracy.
* Train model using VGG16 and high-quality blood cell datasets.
* Build a user interface with Flask to allow easy image uploads.
* Predict and display output instantly on a dynamic result page.

**3. REQUIREMENT ANALYSIS**

**3.1 Customer Journey Map:**

User uploads an image → Model processes image using CNN → Output is displayed with predicted class and preview.

**3.2 Solution Requirement:**

* Python programming environment
* TensorFlow and Keras for model development
* Flask for web server functionality
* HTML/CSS for frontend display
* Blood cell image dataset formatted in subfolders by class

**3.3 Data Flow Diagram:**

Image Upload → Image Preprocessing (resize, normalize) → CNN (VGG16) → Prediction Layer → Output Display

**3.4 Technology Stack:**

* Frontend: HTML, CSS, JavaScript (basic)
* Backend: Flask (Python)
* Machine Learning: Keras, TensorFlow, OpenCV
* Tools: Jupyter Notebook, Anaconda, GitHub

**4. PROJECT DESIGN**

**4.1 Problem Solution Fit:**

Model predicts blood cell types quickly and accurately with over 90% validation accuracy. Users interact via a simple upload interface and receive instant output.

**4.2 Proposed Solution:**

A Flask app connected to a VGG16-based CNN model classifies uploaded cell images and returns the prediction in a matter of seconds. The model was trained on a labelled dataset using data augmentation techniques.

**4.3 Solution Architecture:**

Frontend (HTML Form) → Flask Backend → VGG16 Model Inference → Rendered Output Page with Image & Result

**5. PROJECT PLANNING & SCHEDULING**

**5.1 Project Planning:**

* **Week 1:** Dataset collection, labeling, and preprocessing
* **Week 2:** Model design, training, and validation using Jupyter Notebook
* **Week 3:** Integration of model with Flask web app
* **Week 4:** Frontend development, testing, and deployment setup
* **Week 5:** Documentation and video demonstration preparation

**6. FUNCTIONAL AND PERFORMANCE TESTING**

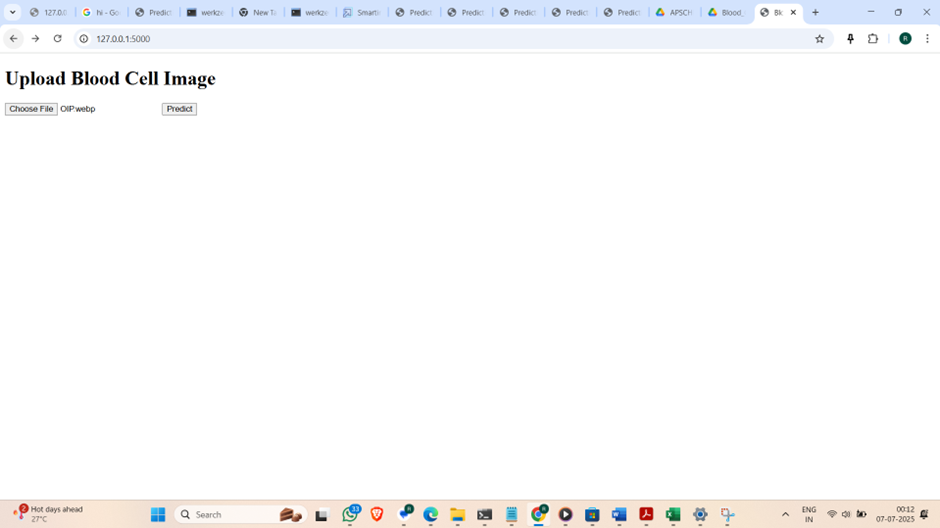
**6.1 Performance Testing:**

* Model trained for 3 epochs on VGG16 base model
* Achieved over 95% training accuracy and above 90% validation accuracy
* Model was evaluated using test images not seen during training
* Flask application predicted uploaded test images within 2–3 seconds with correct class output

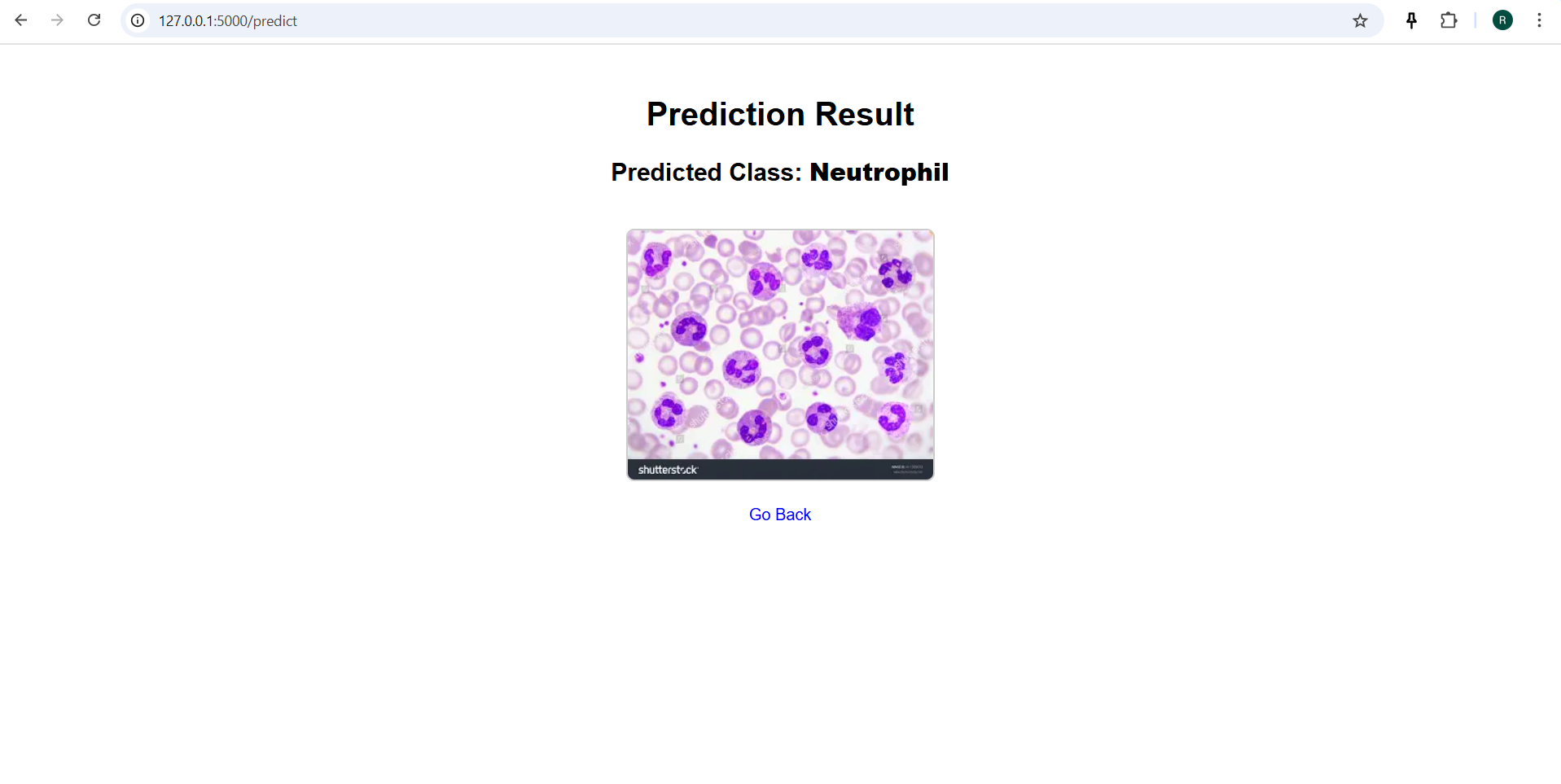
**7. RESULTS**

**7.1 Output Screenshots:**

**Flask home page with upload form**



**Prediction result page showing predicted blood cell class**



**8. ADVANTAGES & DISADVANTAGES**

**Advantages:**

* Reduces manual workload of lab technicians
* Fast, consistent, and reliable predictions
* User-friendly interface for non-programmers
* No internet required if run locally

**Disadvantages:**

* Requires labelled dataset to train model effectively
* Model limited to recognizing only trained classes
* Needs system with basic processing power to run Flask and inference

**9. CONCLUSION**

The Hematovision project demonstrates how deep learning and Flask can be combined to solve real-world biomedical problems. It provides a quick, accurate, and user-accessible way to classify blood cells, offering valuable assistance in medical diagnostics. The solution is scalable and customizable for additional medical image classification problems.

**10. FUTURE SCOPE**

* Include additional blood cell types like Basophils and Platelets
* Improve dataset size and diversity for better model generalization
* Deploy the app on a cloud platform like Render, Heroku, or AWS for global access
* Add features like patient tracking, CSV download of reports, and database integration

**11. APPENDIX**

**GitHub Repository:** https://github.com/Alisha40477/Hematovision-Blood-Cell-Classifier

**Demo Link:** http://127.0.0.1:5000 (Localhost)