

HematoVision: Advanced Blood Cell Classification Using Transfer Learning

## **Submitted By**

**Group Members:**

**Team ID :** LTVIP2025TMID46914

**Team Size :** 4

**Team Leader :** Suri Mahira Khatun

**Team member :** Chandhamala Akhila

**Team member :** N.Kethana

**Team member :** M.Mounika

**1. Project Overview**

Blood cell classification is a critical task in hematology, enabling the diagnosis of various blood disorders including leukemia, anemia, and infections. This project, **HematoVision**, leverages **transfer learning** to accurately classify blood cells from microscopic images. The solution integrates a deep learning model with a **Flask-based web application**, allowing users to upload cell images and instantly receive classification results via an interactive and visually appealing interface.

**2. Objectives**

* To collect and preprocess image data of blood cells.
* To apply transfer learning using pre-trained CNN models for image classification.
* To build and train an optimized model to classify white blood cell types (e.g., lymphocytes, monocytes, neutrophils, eosinophils).
* To deploy the trained model using Flask.
* To create an intuitive and user-friendly web interface for image upload and prediction display.

**3. Tools & Technologies Used**

* **Programming Language:** Python
* **Libraries & Frameworks:**
  + TensorFlow, Keras – for deep learning
  + OpenCV, PIL, NumPy, Matplotlib – for image processing and visualization
  + Flask – for model deployment as a web application
* **Frontend:** HTML, CSS
* **Model Deployment:** Flask Web Framework
* **Platform:** Google Colab / Jupyter Notebook / VS Code

**4. Dataset Description**

The dataset contains **labeled microscopic images of white blood cells**, primarily:

* **Neutrophils**
* **Eosinophils**
* **Lymphocytes**
* **Monocyte**

Each image is labeled with its corresponding class. The data was sourced from:

* Kaggle: Blood Cell Images Dataset

**5. Workflow**

**1. Data Preprocessing**

* Resized and normalized images
* Applied data augmentation to improve generalization
* Converted images to NumPy arrays and labeled appropriately

**2. Model Training**

* Used transfer learning with **MobileNetV2**
* Fine-tuned top layers of the model on the dataset
* Evaluated with metrics: accuracy, confusion matrix

**3. Scaling**

* Used Rescaling(1./255) to normalize pixel values
* Image data prepared using image\_dataset\_from\_directory()

**4. Model Serialization**

* Saved the trained model using model.save('hematovision\_model.h5')

**5. Flask App**

* Flask routes to upload image and get predictions
* Backend loads model and performs prediction
* Output label displayed along with the uploaded image

**6. Frontend UI**

* Built using HTML & CSS
* Input form to upload images
* Set a medical-themed background image to enhance UX

**6. Screenshots**

Include:

* Homepage with image upload option
* Example of uploaded image
* Output showing predicted cell type

**7. Results**

The model achieved high accuracy (>90%) on validation data.  
**Sample Prediction:**

* Uploaded Image: White blood cell
* Output: **Predicted class – Neutrophil**

**8. Challenges Faced**

* Handling varied image sizes and lighting conditions
* Ensuring correct image-label mapping after augmentation
* Managing large model size for deployment
* Adapting pre-trained models to medical image domain

**9. Conclusion**

**HematoVision** demonstrates how deep learning and transfer learning can significantly enhance the efficiency and accuracy of blood cell classification. The web-based interface ensures easy accessibility and can be extended to aid medical professionals in diagnostic tasks.

**10. Future Work**

* Extend to classify abnormal cells (e.g., leukemic blasts)
* Integrate Explainable AI (XAI) techniques like **Grad-CAM**
* Host on cloud platforms like **Render, Heroku, or AWS**
* Enable batch predictions and PDF report generation
* Add integration with medical lab software systems

**11. References**

* TensorFlow and Keras documentation
* Kaggle: Blood Cells Dataset
* Flask official documentation
* Tutorials from SmartBridge, Coursera, and Medium articles

**Thank You!**  
*Project guided and supported by SmartBridge.*