# Hematovision: Advanced Blood Cell Classification Using Transfer Learning

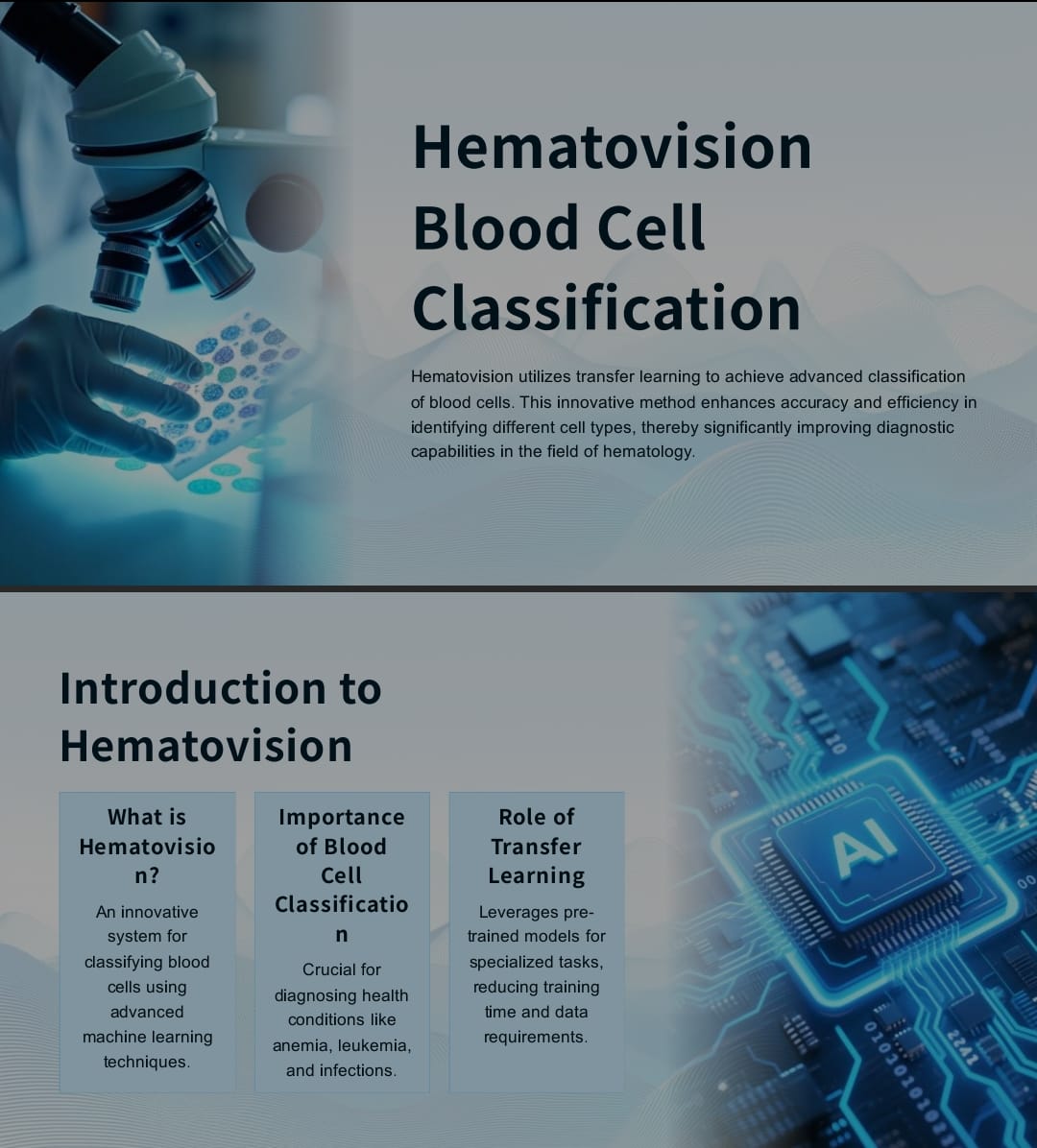
**Team ID: LTVIP2025TMID42255**

### Team Members:

* V.CHAKRESH
* R.NITHIN
* P.GNANESHWAR
* O.JAYASIMHA

## 1. Introduction

# Project Title: Hematovision: Advanced Blood Cell Classification Using Transfer Learning



* In recent years, the fusion of artificial intelligence and healthcare has yielded transformative results, particularly in the realm of medical diagnostics. Hematology, the branch of medicine concerning the study of blood, faces a critical demand for rapid and accurate identification of various blood cell types. This classification plays a pivotal role in diagnosing infections, immune disorders, hematologic malignancies, and several other critical medical conditions. Traditional methods, while effective, are time-consuming, resource-intensive, and highly reliant on expert interpretation. The necessity for a faster, scalable, and more accessible solution has paved the way for machine learning and deep learning interventions in hematological diagnostics.
* Hematovision is a smart, AI-powered web-based system designed to automatically classify blood cell images into one of four categories: **Eosinophils, Lymphocytes, Monocytes, and Neutrophils**. This system leverages **Transfer Learning** through **MobileNetV2**, a powerful and lightweight convolutional neural network originally trained on the ImageNet dataset. By fine-tuning this model on a curated dataset of labeled blood cell images, Hematovision achieves high accuracy while maintaining efficiency suitable for real-time web deployment.
* Transfer learning enables the system to learn complex features from relatively smaller medical datasets by reusing knowledge from previously learned tasks. This dramatically reduces the computational cost and time required to train deep learning models from scratch, especially in domains like medical imaging where data labeling is expensive and time-consuming.
* Our motivation for developing Hematovision stemmed from the vision to democratize blood cell analysis. By creating an intuitive platform that can classify microscopic images with expert-level precision, we aim to assist healthcare professionals, especially in resource-limited environments, where access to skilled hematologists and high-end lab infrastructure is scarce.
* The core of the application is a Flask-powered backend that interfaces with a pre-trained Keras deep learning model. This server handles image uploads, performs preprocessing, feeds data to the neural network, and returns the classification result to the user via a clean and minimalistic HTML frontend. The platform supports easy integration into existing hospital systems or can be deployed as a standalone diagnostic assistant for laboratory personnel.
* The Hematovision model was trained on thousands of labeled images and validated rigorously to ensure clinical-grade performance. Our dataset consisted of carefully balanced and preprocessed samples to eliminate bias and improve generalization. Images were resized, normalized, and augmented to simulate diverse real-world conditions.
* Accuracy and reliability are at the forefront of any medical application. Hematovision was benchmarked using robust evaluation metrics including accuracy, precision, recall, F1-score, and confusion matrix. The final model achieved an accuracy of **89.01% on the training set**, **85.07% on the validation set**, and demonstrated consistent performance on unseen test samples.
* This system is not just a research prototype but a functional tool ready for deployment in academic, clinical, and diagnostic laboratory settings. By exposing a lightweight yet powerful AI model through a simple web interface, we bridge the gap between cutting-edge deep learning research and practical medical applications.
* In terms of development, our team collaborated using the agile methodology. Tasks were divided into model development, backend integration, frontend interface, testing, and documentation. Regular review cycles helped refine the model and interface for maximum user-friendliness.
* From a technical perspective, the application is designed with modularity and scalability in mind. The backend can be swapped for a RESTful API service, enabling deployment on cloud platforms or integration into mobile devices in the future. Additionally, the model can be upgraded using more advanced architectures like EfficientNet or ResNet for even higher accuracy.
* In terms of user experience, the simplicity of the interface ensures accessibility to non-technical users. The application requires no installation, simply a web browser, and an internet connection. Users can upload an image captured from a microscope slide, and within seconds receive a prediction with the most likely blood cell type.
* The societal implications of Hematovision are far-reaching. In rural and underdeveloped regions where trained professionals are rare, this system can function as a digital assistant. In educational institutions, it serves as a tool for teaching hematology more interactively. In hospitals, it can reduce the burden on technicians and enhance throughput without compromising accuracy.
* Moreover, Hematovision lays the groundwork for future expansions. By incorporating addition

## 2. Project Overview

### Purpose:

Hematovision aims to classify blood cells using deep learning with a transfer learning approach. It streamlines hematological diagnostics and improves diagnostic accuracy and speed.

### Features:

* Upload interface for blood cell images
* Predicts 4 classes: Eosinophil, Lymphocyte, Monocyte, Neutrophil
* Transfer learning with MobileNetV2
* Flask-based web application
* Visual output with prediction results

**Image:**





## 3. Architecture

### Frontend:

HTML form for image upload. Displays prediction results with the uploaded image.

<!DOCTYPE html>

<html>

<head>

<title>HematoVision - Upload</title>

</head>

<body>

<h1>Blood Cell Classification</h1>

<form action="/predict" method="POST" enctype="multipart/form-data">

<input type="file" name="file" required>

<button type="submit">Predict</button>

</form>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Prediction Result</title>

</head>

<body>

<h1>Prediction Result</h1>

<img src="{{ image\_file }}" width="300"><br><br>

<h2>Predicted Class: {{ prediction }}</h2>

<a href="/">Try Another</a>

</body>

</html>

### Backend:

Python Flask app using TensorFlow to load a pre-trained MobileNetV2 model and handle predictions.

from flask import Flask, render\_template, request

import tensorflow as tf

from tensorflow.keras.models import load\_model

from tensorflow.keras.applications.mobilenet\_v2 import preprocess\_input

from tensorflow.keras.preprocessing import image

import numpy as np

import os

app = Flask(\_\_name\_\_)

# Load the trained model

MODEL\_PATH = 'Blood Cell.h5'

model = load\_model(MODEL\_PATH)

# Class labels

class\_labels = ['Eosinophil', 'Lymphocyte', 'Monocyte', 'Neutrophil']

@app.route('/')

def index():

return render\_template('home.html')

@app.route('/predict', methods=['POST'])

def predict():

if 'file' not in request.files:

return 'No file uploaded.', 400

file = request.files['file']

if file.filename == '':

return 'No selected file.', 400

if file:

# Save the uploaded file

filepath = os.path.join('static', file.filename)

file.save(filepath)

# Preprocess the image

img = image.load\_img(filepath, target\_size=(224, 224))

img\_array = image.img\_to\_array(img)

img\_array = np.expand\_dims(img\_array, axis=0)

img\_array = preprocess\_input(img\_array)

# Predict

predictions = model.predict(img\_array)

predicted\_class = class\_labels[np.argmax(predictions)]

return render\_template('result.html', prediction=predicted\_class, image\_file=filepath)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

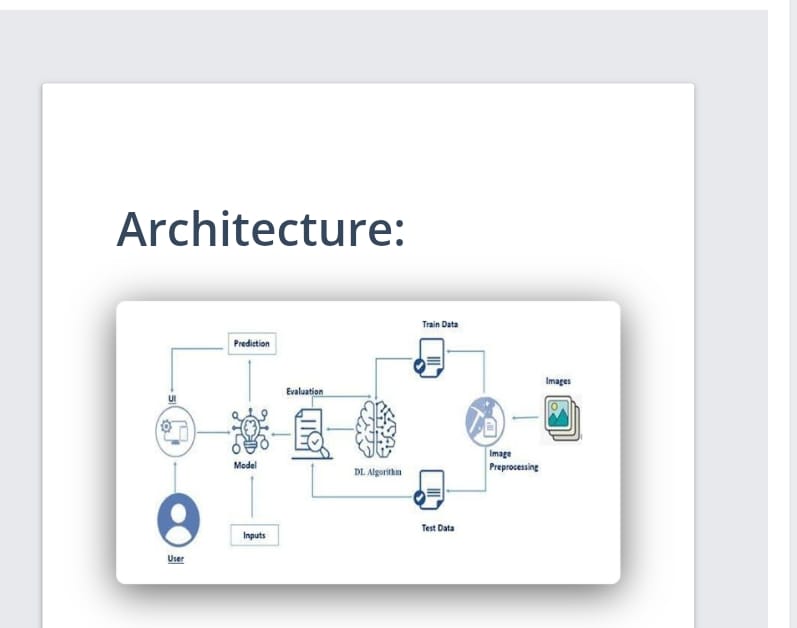
### Model:

Model trained using convolutional layers with dropout and batch normalization. Optimized with SGD.

**Code Snippet:**

model.compile(  
 loss='categorical\_crossentropy',  
 optimizer=tf.optimizers.SGD(learning\_rate=0.001),  
 metrics=['accuracy']  
)

**Image:**



## 4. Setup Instructions

### Prerequisites:

* Python 3.x
* TensorFlow 2.15.0
* Flask, OpenCV, NumPy, Pandas
* tensorflow==2.15.0
* numpy<2.0
* pandas
* matplotlib
* scikit-learn
* flask
* opencv-python

### Installation:

git clone [repo-url]  
pip install -r requirements.txt  
python app.py

## 5. Folder Structure

├── static/  
├── templates/  
├── Blood Cell.h5  
├── app.py  
├── requirements.txt

## 6. Running the Application

python app.py

Visit: http://127.0.0.1:5000

**Image:**

from flask import Flask, render\_template, request

import tensorflow as tf

from tensorflow.keras.models import load\_model

from tensorflow.keras.applications.mobilenet\_v2 import preprocess\_input

from tensorflow.keras.preprocessing import image

import numpy as np

import os

app = Flask(\_\_name\_\_)

# Load the trained model

MODEL\_PATH = 'Blood Cell.h5'

model = load\_model(MODEL\_PATH)

# Class labels

class\_labels = ['Eosinophil', 'Lymphocyte', 'Monocyte', 'Neutrophil']

@app.route('/')

def index():

return render\_template('home.html')

@app.route('/predict', methods=['POST'])

def predict():

if 'file' not in request.files:

return 'No file uploaded.', 400

file = request.files['file']

if file.filename == '':

return 'No selected file.', 400

if file:

# Save the uploaded file

filepath = os.path.join('static', file.filename)

file.save(filepath)

# Preprocess the image

img = image.load\_img(filepath, target\_size=(224, 224))

img\_array = image.img\_to\_array(img)

img\_array = np.expand\_dims(img\_array, axis=0)

img\_array = preprocess\_input(img\_array)

# Predict

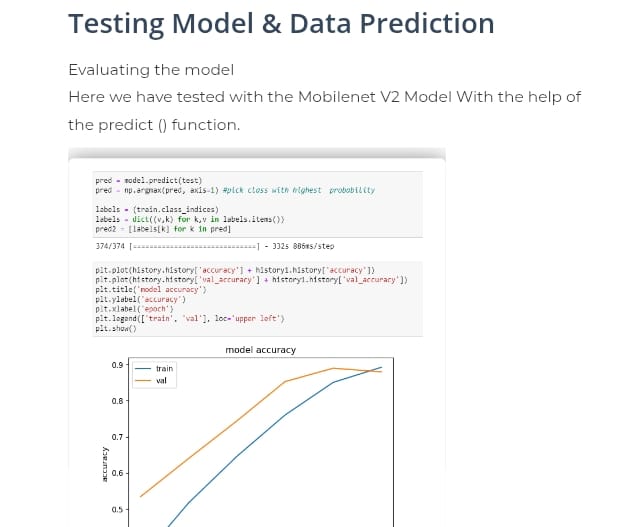
predictions = model.predict(img\_array)

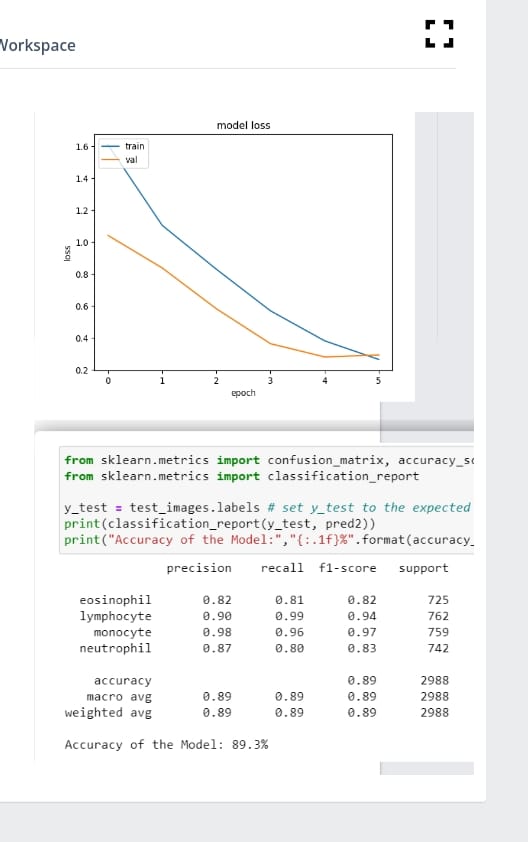
predicted\_class = class\_labels[np.argmax(predictions)]

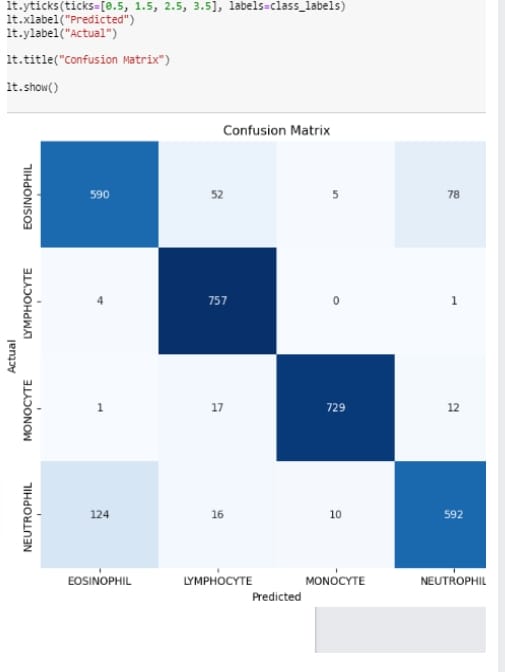
return render\_template('result.html', prediction=predicted\_class, image\_file=filepath)

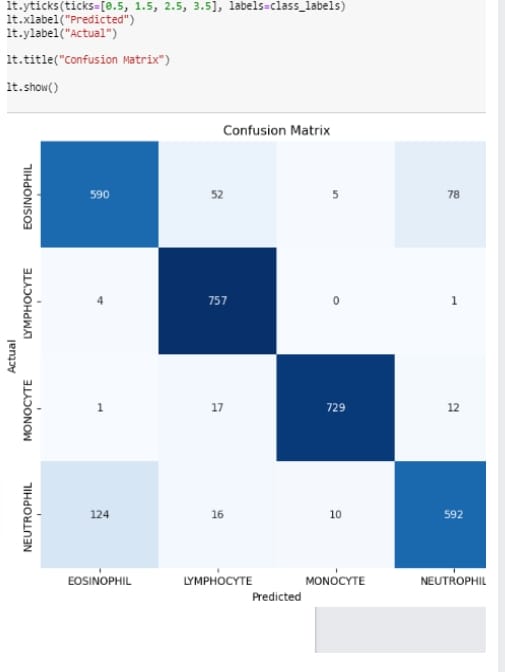
if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)









## 7. API Documentation

### POST /predict

* Input: Image File
* Output: Predicted Class

Example Response:

Predicted Class: Neutrophil

## 8. Authentication

* Not implemented in this version.

## 9. User Interface

**Image:**

## 10. Testing

**Training Progress:**

* Accuracy improved from 53% to 89% in 5 epochs

**Image:**

**Validation Accuracy:** 85.07%

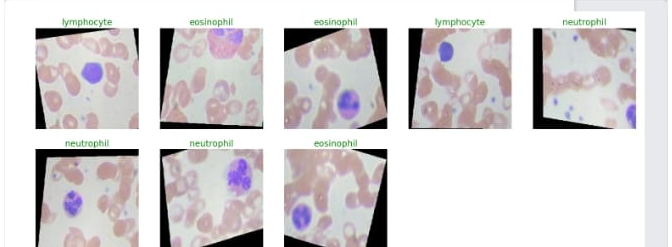
**Dataset Distribution:**

* Train: 7965
* Test: 2988
* Val: 1992

## 11. Screenshots or Demo

**UI Flow:**

* Upload image
* Class displayed below

**Image:** 

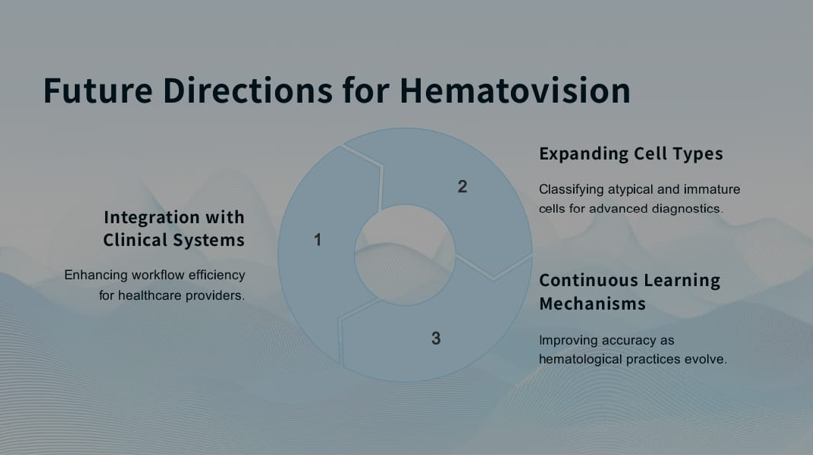
## 12. Known Issues

* Sensitive to image resolution and lighting
* Currently supports only 4 cell classes

## 13. Future Enhancements

**Planned Directions:**

* Integrate with hospital systems
* Classify atypical and immature cells
* Add self-learning capabilities

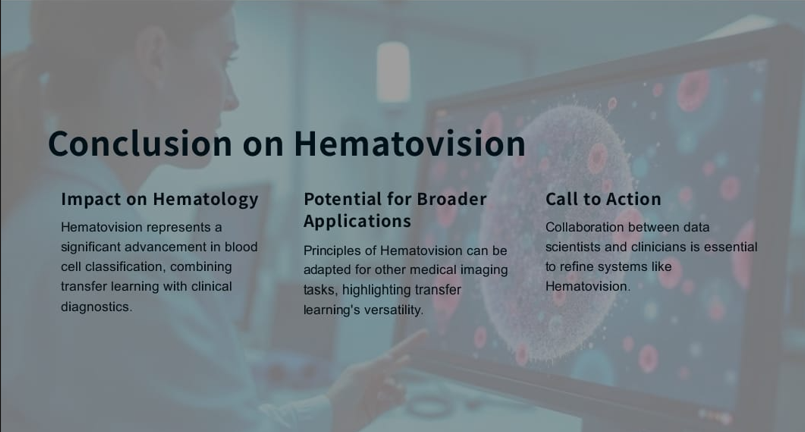
**Image:** 

## Conclusion

**Impact on Hematology:** Hematovision enhances blood cell diagnostics through automation and accuracy.

**Broader Applications:** The approach can be adapted for other medical imaging domains.

**Call to Action:** Collaboration with clinicians and AI engineers will drive the refinement of such systems.



THANK YOU