# Project Report: HematoVision - Blood Cell Classification

## 1. Introduction

HematoVision is a full-stack application that leverages deep learning to classify human white blood cells from microscopic images. The project’s primary goal is to provide a simple, web-based tool for users to upload an image and receive an instant classification of the cell type. This technology has potential applications in medical diagnostics, education, and research by automating a traditionally manual process.

The system is built using a Python-based backend with the Flask framework and a simple HTML/CSS frontend. The core of the application is a convolutional neural network (CNN) built with TensorFlow and Keras, utilizing transfer learning from the MobileNetV2 architecture for high accuracy and efficiency.

## 2. Project Objectives

* To develop a robust deep learning model capable of accurately classifying four types of white blood cells: Eosinophil, Lymphocyte, Monocyte, and Neutrophil.
* To build a user-friendly web interface for easy image submission.
* To create a scalable backend service that can load the model, process user requests, and serve predictions.
* To integrate the machine learning model with the web application to create a seamless end-to-end user experience.
* To document the project structure, setup, and usage for future development and maintenance.

## 3. System Architecture & Design

The application follows a standard client-server architecture:

* **Client (Frontend)**: A simple web interface built with HTML and CSS. It provides a form for the user to select and upload an image file. It is responsible for rendering the prediction result returned by the server.
* **Server (Backend)**: A Flask application written in Python. Its responsibilities include:
  + Handling HTTP requests from the client.
  + Serving the static HTML pages.
  + Managing file uploads and saving them to the server.
  + Loading the pre-trained Keras model (Blood Cell.h5).
  + Preprocessing the uploaded image to match the model’s input requirements.
  + Executing the prediction using the model.
  + Returning the final classification to the client.
* **Machine Learning Model**: A pre-trained deep learning model (Blood Cell.h5). It was trained on a labeled dataset of blood cell images and is capable of identifying features specific to each cell type.

## 4. Methodology

### 4.1. Dataset

The model was trained on the “Blood Cell Images” dataset, which contains approximately 12,500 images of blood cells, categorized into four classes. The data was split into training and testing sets to ensure proper evaluation.

### 4.2. Data Preprocessing & Augmentation

The ImageDataGenerator from Keras was used for preprocessing. The key steps included: - Rescaling pixel values from the [0, 255] range to the [0, 1] range. - Resizing all images to 224x224 pixels, the required input size for MobileNetV2. - Applying data augmentation techniques (rotation, zoom, flips) to the training set to increase its diversity and prevent overfitting.

### 4.3. Model Training

* **Transfer Learning**: The MobileNetV2 architecture, pre-trained on the ImageNet dataset, was used as the base model.
* **Model Customization**: The top classification layer of MobileNetV2 was removed and replaced with a new sequence of layers: a Flatten layer, a Dropout layer (for regularization), and a final Dense layer with a softmax activation function for multi-class classification.
* **Compilation**: The model was compiled with the Adam optimizer, categorical\_crossentropy loss function, and accuracy as the evaluation metric.
* **Training**: The model was trained for 5 epochs. ModelCheckpoint was used to save the best-performing model based on validation accuracy.

## 5. Implementation Details

* **Flask (app.py)**: Manages routing, file handling, and orchestrates the prediction process.
* **TensorFlow/Keras (train\_model.py)**: Defines the model architecture, data preprocessing pipeline, and the training loop.
* **HTML/CSS (templates/ & static/)**: Provides the structure and styling for the user interface.
* **Requirements**: Key Python libraries include flask, tensorflow, and numpy.

## 6. Conclusion and Future Scope

The HematoVision project successfully demonstrates the integration of a powerful deep learning model into a practical web application. The system correctly performs the end-to-end task of accepting a user-submitted image and returning a classification.

Future work could focus on: - **Improving Model Accuracy**: Training for more epochs, fine-tuning more layers of the base model, or experimenting with different CNN architectures. - **Enhancing the User Interface**: Adding features like drag-and-drop file uploads, displaying prediction confidence scores, and providing more detailed information about the predicted cell type. - **Deployment**: Deploying the application to a cloud service (e.g., Heroku, AWS, Google Cloud) to make it publicly accessible. - **Expanding Functionality**: Training the model to detect more cell types or even identify abnormalities.