Artificial Intelligence and MachineLearning

Project Documentation

HematoVision: Advanced Blood Cell Classification Using Transfer Learning
Project Documentation

1. Introduction

Project Title: HematoVision: Advanced Blood Cell Classification Using Transfer Learning

Team Members:

Team Leader: Koushik Javvaja

Team Member: Kokkiripati Hema Varsha

Team Member: Kollati Ribka

Team Member: Kuna Durga Sai Ruthvik

2. Project Overview

Purpose: HematoVision aims to develop an accurate and efficient model for classifying blood cells by employing transfer learning techniques. It provides a reliable and scalable tool for pathologists and healthcare professionals, ensuring precise and efficient blood cell classification.

Features:

Classification of blood cells (eosinophils, lymphocytes, monocytes, and neutrophils) using transfer learning.

Leverages pre-trained Convolutional Neural Networks (CNNs) for expedited training and improved accuracy.

Integration into automated diagnostic systems for real-time blood analysis and report generation.

Application in telemedicine platforms for remote consultations and diagnostics.

Incorporation into educational tools for medical training, offering interactive learning and instant feedback.

3. Architecture

Frontend: The frontend is built using HTML pages (home.html and result.html), which provide the User Interface (UI) for image selection and displaying prediction results.

Backend: The backend is developed using Flask, a Python web framework. It handles the logic for receiving image uploads, processing them through the trained model, and returning predictions to the frontend.

Database: No explicit database is mentioned or used in the provided project description. The project focuses on model integration and serving predictions via a web application.

4. Setup Instructions

Prerequisites:

Anaconda Navigator (refer to the provided link for download).

Prior knowledge of Deep Learning Concepts (Neural Networks, Deep Learning Frameworks, Transfer Learning, VGG16, Convolutional Neural Networks (CNNs), Overfitting and Regularization, Optimizers).

Prior knowledge of Flask Basics.

Installation:

Open Anaconda Prompt as administrator.

Type pip install numpy and press Enter.

Type pip install pandas and press Enter.

Type pip install scikit-learn and press Enter.

Type pip install matplotlib and press Enter.

Type pip install scipy and press Enter.

Type pip install seaborn and press Enter.

Type pip install tensorflow and press Enter.

Type pip install Flask and press Enter.

5. Folder Structure

The project folder contains the following structure:

```
Project Folder/
├── app.py
import os
import numpy as np
import cv2
from flask import Flask, request, render_template, redirect, url_for
from tensorflow.keras.models import load_model
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
import matplotlib.pyplot as plt
import io
import base64

app = Flask(_name_)
model = load_model("Blood Cell.h5")
class_labels = ['eosinophil', 'lymphocyte', 'monocyte', 'neutrophil']

def predict_image_class(image_path, model):
    img = cv2.imread(image_path)
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img_resized = cv2.resize(img_rgb, (224, 224))
    img_preprocessed = preprocess_input(img_resized.reshape((1, 224, 224, 3)))
    predictions = model.predict(img_preprocessed)
    predicted_class_idx = np.argmax(predictions, axis=1)[0]
    predicted_class_label = class_labels[predicted_class_idx]
    return predicted_class_label, img_rgb

@app.route("/", methods=["GET", "POST"])
def upload_file():
    if request.method == "POST":
        if "file" not in request.files:
            return redirect(request.url)
        file = request.files["file"]
        if file.filename == "":
            return redirect(request.url)
        if file:
            file_path =os.path.join("static", file.filename)
            file.save(file_path)
            predicted_class_label, img_rgb = predict_image_class(file_path, model)

            _, img_encoded = cv2.imencode('.png', cv2.cvtColor(img_rgb,
cv2.COLOR_RGB2BGR))
            img_str = base64.b64encode(img_encoded).decode('utf-8')

            return render_template("result.html", class_label=predicted_class_label,
img_data=img_str)
    return render_template("home.html")

if _name_ == "_main_":
    port = int(os.environ.get("PORT", 5000))
    app.run(debug=True, host="0.0.0.0", port=port)
```

```
├── Blood Cell.h5
└── templates/
    ├── home.html
```

```
<!-- templates/home.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Blood Cell Classification</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background: linear-gradient(135deg, #e0f7fa, #f1f8e9);
            color: #333;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }

        .upload-box {
            background: #ffffff;
            border-radius: 16px;
            box-shadow: 0 12px 28px rgba(0, 0, 0, 0.1);
            padding: 40px 50px;
            text-align: center;
            transition: transform 0.3s ease;
            border: 1px solid #e0e0e0;
        }

        .upload-box:hover {
            transform: scale(1.015);
        }

        .upload-box h2 {
            margin-bottom: 25px;
            font-size: 24px;
            color: #2c3e50;
        }

        input[type="file"] {
            background: #f9f9f9;
            color: #333;
            border: 1px solid #ccc;
            padding: 10px;
            border-radius: 8px;
            margin: 20px 0;
            cursor: pointer;
            transition: border-color 0.2s ease;
        }

        input[type="file"]:hover {
            border-color: #66bb6a;
        }

        input[type="file"]::-webkit-file-upload-button {
            background-color: #66bb6a;
            color: white;
            border: none;
            padding: 8px 15px;
            border-radius: 6px;
            cursor: pointer;
            transition: background-color 0.3s;
        }

        input[type="file"]::-webkit-file-upload-button:hover {
            background-color: #43a047;
        }
```

```
        button {
            background: linear-gradient(to right, #42a5f5, #478ed1);
            color: #fff;
            padding: 12px 30px;
            font-size: 16px;
            border: none;
            border-radius: 8px;
            cursor: pointer;
            transition: background 0.3s, transform 0.2s;
        }

        button:hover {
            background: linear-gradient(to right, #1e88e5, #1565c0);
            transform: scale(1.05);
        }
    </style>
</head>
<body>
    <div class="upload-box">
        <h2>Upload a Blood Cell Image</h2>
        <form method="POST" enctype="multipart/form-data">
            <input type="file" name="file" accept="image/*" required><br>
            <button type="submit">Upload & Predict</button>
        </form>
    </div>
</body>
</html>
```

└── result.html

```
<!-- templates/result.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Prediction Result</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background: linear-gradient(135deg, #f1f8e9, #e0f7fa);
            color: #333;
            display: flex;
            justify-content: center;
            align-items: center;
        height: 100vh;
        margin: 0;
    }

    .result-box {
        background-color: #ffffff;
        padding: 40px 50px;
        border-radius: 16px;
        box-shadow: 0 12px 28px rgba(0, 0, 0, 0.1);
        text-align: center;
        max-width: 600px;
        width: 90%;
        transition: transform 0.3s ease;
        border: 1px solid #e0e0e0;
    }

    .result-box:hover {
        transform: scale(1.01);
    }

    .result-box h2 {
        font-size: 26px;
        color: #2c3e50;
        margin-bottom: 20px;
    }

    .result-box p {
        font-size: 18px;
        margin: 10px 0 20px;
        color: #444;
    }

    .result-box img {
        max-width: 100%;
        border-radius: 12px;
        margin-top: 20px;
        box-shadow: 0 4px 14px rgba(0, 0, 0, 0.08);
    }

    .result-box a {
        display: inline-block;
        margin-top: 30px;
        padding: 12px 28px;
        background: linear-gradient(to right, #66bb6a, #43a047);
```

```
            color: white;
            font-size: 16px;
            text-decoration: none;
            border-radius: 8px;
            transition: background 0.3s, transform 0.2s;
        }

        .result-box a:hover {
            background: linear-gradient(to right, #43a047, #2e7d32);
            transform: scale(1.05);
        }
    </style>
</head>
<body>
    <div class="result-box">
        <h2>Prediction Result</h2>
        <p><strong>Predicted Class:</strong> {{ class_label }}</p>
        <img src="data:image/png;base64,{{ img_data }}" alt="Uploaded Image">
        <br>
        <a href="{{ url_for('upload_file') }}">Predict Another Image</a>
    </div>
</body>
</html>
```

app.py: The main Python script for the Flask application.

Blood Cell.h5: The saved pre-trained blood cell classification model.

Templates: A directory containing the HTML template files for the user interface.

6. Running the Application

To run the web application locally:

Open Anaconda Prompt from the start menu.

Navigate to the project folder where your Python script (app.py) is located.

Type the command python app.py and press Enter.

Open your web browser and navigate to the localhost URL: http://127.0.0.1:5000.

7. API Documentation

The Flask application exposes the following endpoints:

Endpoint: /

Method: GET

Description: Renders the home.html page, which is the main UI for uploading images.

Parameters: None

Example Response: Renders the HTML content of home.html.

Endpoint: /output

Method: POST

Description: Receives the uploaded image file, processes it using the loaded Blood Cell.h5 model, and returns the prediction result.

Parameters:

image_file: The blood cell image uploaded by the user (via HTML form input).

Example Response: Renders the result.html page, displaying the predicted blood cell class (e.g., "Neutrophil", "Monocyte", "Lymphocyte", "Eosinophil").

8. Authentication

The provided project description does not include details about authentication or authorization mechanisms. The application appears to be designed for direct access without explicit user login or role-based access control.

9. User Interface

The application provides a simple web-based user interface:

Home Page (home.html):

Allows users to choose an image file to upload.

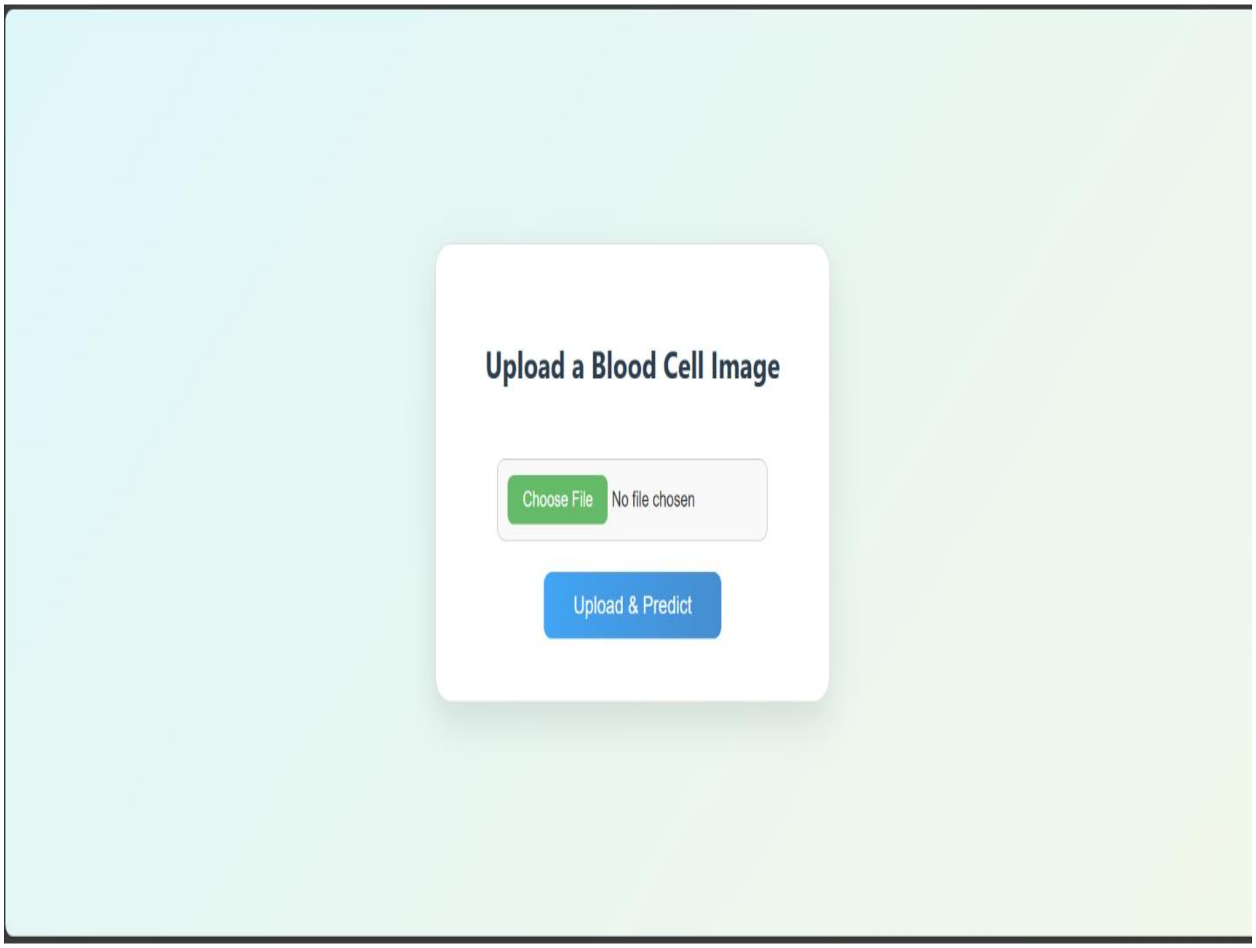
Features a "Predict" button to initiate the classification process.

(As described in the prompt) "By clicking on choose file it will ask us to upload the image, then by clicking on the predict button, it will take us to the result.html

Result Page (result.html):

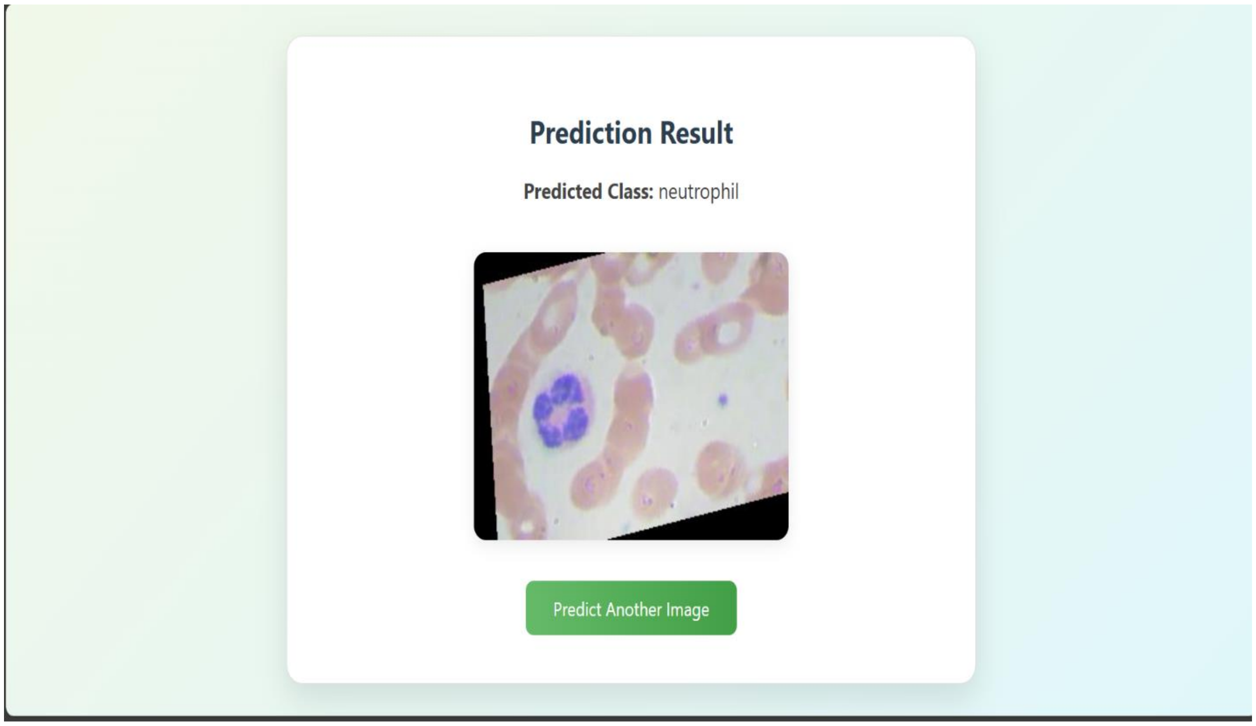
Displays the predicted classification of the uploaded blood cell image.

WEB PAGE



Examples:

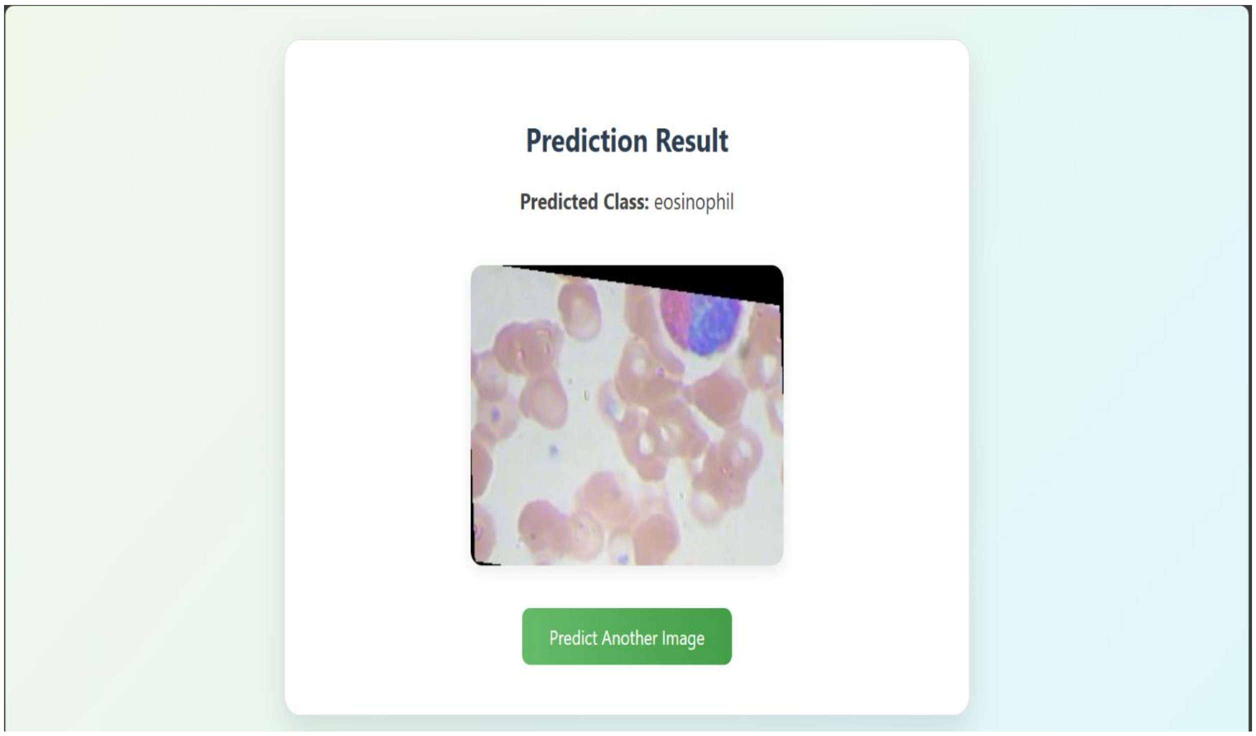
Test For Class-1: Neutrophil



Prediction Result

Predicted Class: neutrophil

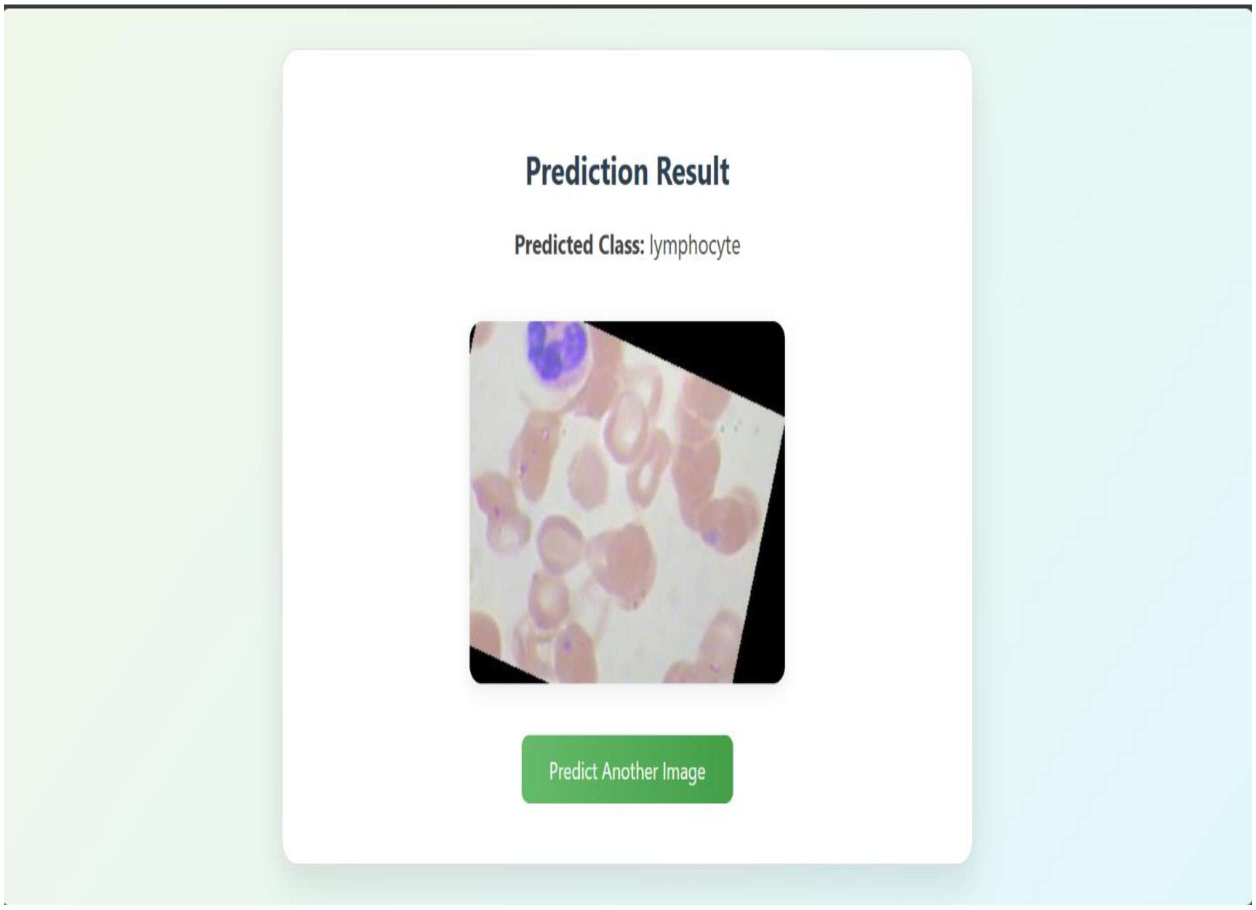
Test For Class- 2 : Eosinophil



Prediction Result

Predicted Class: eosinophil

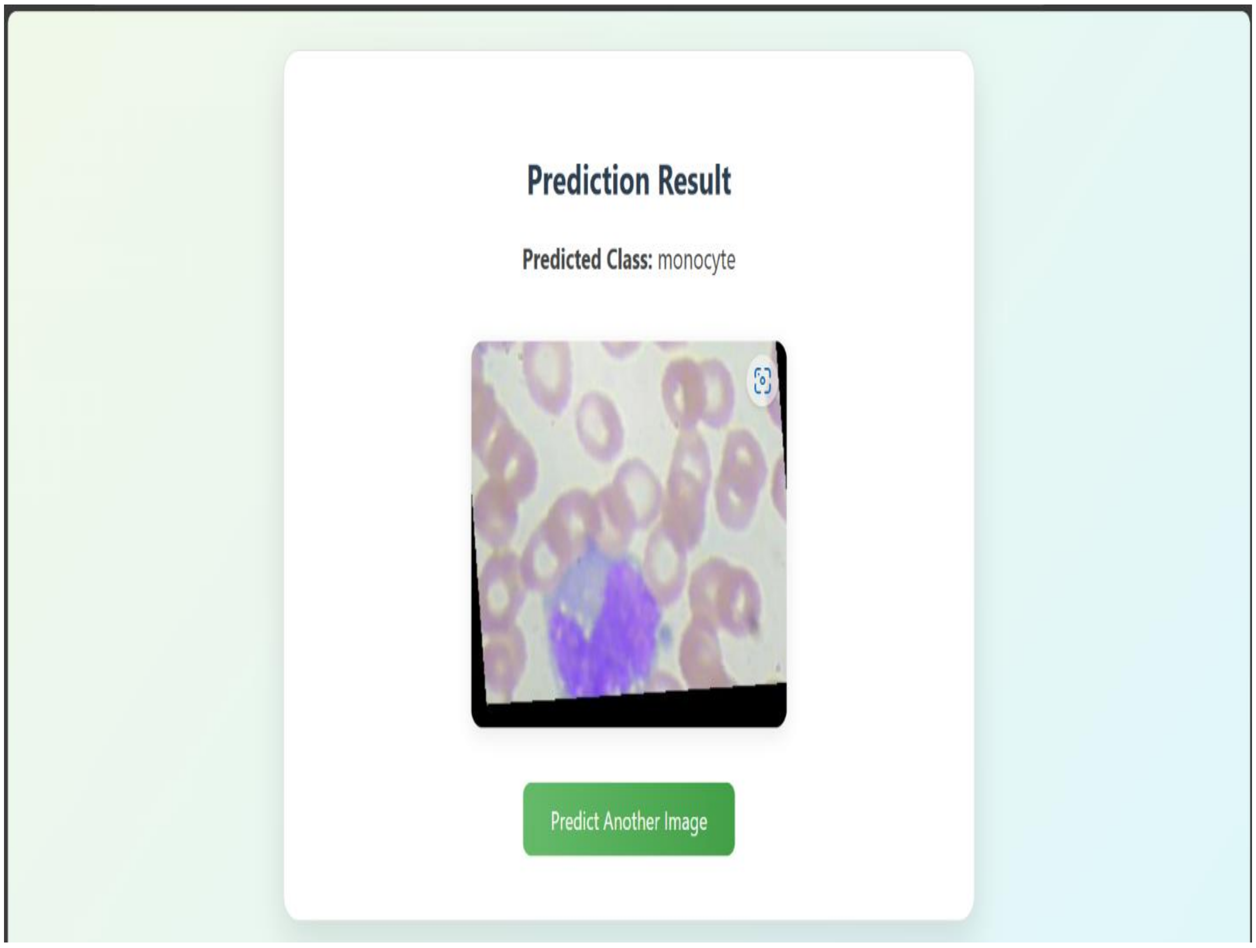
Test For Class-3: Lymphocyte



Prediction Result

Predicted Class: lymphocyte

Test For Class-4: Monocyte



Predicted Class: monocyte

10. Testing

The model was tested using the predict() function of the MobileNetV2 model. The evaluation involved predicting the class of various blood cell images (Neutrophil, Monocyte, Lymphocyte, Eosinophil) and showcasing the results on the UI. The model was trained for 5 epochs with callbacks like Model Checkpoint and Early Stopping to ensure optimal performance.

11. Screenshots or Demo

https://drive.google.com/file/d/1fm-U_yhKdk9TwT8hzplfy4dX-T7ysyFM/view?usp=drivesdk

12. Known Issues

There are no known bugs or issues explicitly documented in the provided project description.

13. Future Enhancements

The provided project description does not explicitly outline potential future features or improvements. However, based on the project's scope, potential enhancements could include:

Implementing a user authentication system.

Adding a history of predictions for users.

Integrating with a database to store prediction logs and user data.

Improving the UI/UX for a more intuitive experience.

Expanding the model to classify more types of blood cells or detect anomalies.

Deploying the application to a cloud platform for wider accessibility.