An internship in

Artificial Intelligence & Machine Learning

by

SmartInternz

Project Name : HematoVision: Advanced Blood Cell Classification Using Transfer Learning Estimation with Machine Learning

Project Id : LTVIP2025TMID59822

Project Mentor : M. Ganesh

Team Members and Roles:

1. **Kuruva Pakkirappa Gari Swetha** – Project Leader & Deep Learning
2. **Kuruva Thirumalesh** – Data Analyst & Preprocessing Specialist.
3. **Peddaveti Salmon Raju**– Web Developer & UI Designer.
4. **Mynam Sujeth**– System Integrator & Tester

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

Annamacharya Institute of Technology & Sciences, Utukur (Post), Chinthakomma Dinne (V&M), Kadapa, YSR (Dist) Andhra Pradesh

Project Report Format

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(if any)
- Dataset Link
- GitHub & Project Demo Link

1.INTRODUCTION

1.1 Project Overview

Hematovision is an intelligent, AI-powered diagnostic tool designed to classify and analyze blood cells using advanced deep learning techniques. By leveraging transfer learning, this system enhances the accuracy and efficiency of blood cell classification, a critical step in diagnosing a variety of hematological conditions such as leukemia, anemia, and infections.
The system utilizes pre-trained convolutional neural networks (CNNs) to identify and differentiate between various types of white blood cells—such as neutrophils, lymphocytes, monocytes, and eosinophils—from microscopic blood smear images. Hematovision aims to assist pathologists and healthcare professionals in performing faster, more reliable diagnostics with minimal human error.

1.2 Purpose

The primary purpose of Hematovision is to automate and enhance the blood cell classification process by:

- Reducing the time required for manual slide analysis.
- Increasing diagnostic accuracy, especially in resource-limited or high-volume settings.
- Assisting medical professionals with a reliable second opinion.
- Providing a scalable solution for integration into digital pathology systems.

Features:

- AI-Based Image Classification using pretrained models (e.g., VGG16, ResNet)
- Real-Time Image Upload & Prediction
- User-Friendly Web Interface
- Visualization of Classification Results
- High Accuracy through Transfer Learning Techniques
- Support for Multiple Blood Cell Types
- Model Performance Metrics Display (Accuracy, Loss, Confusion Matrix)

2. IDEATION PHASE

2.1 Problem Statement

Manual classification of blood cells under a microscope is time-consuming, labor-intensive, and subject to human error. This process requires significant expertise, and variations in results between observers can lead to diagnostic delays or inaccuracies. With the rise of digital pathology, there is a critical need for automated, intelligent systems that can assist in accurate and efficient blood cell classification.

Hematovision addresses this gap by employing transfer learning-based deep learning models to automate the classification of blood cell types, helping pathologists make faster and more accurate diagnoses.

2.2 Empathy Map Canvas

Sections	Details
Says	"We need faster and more reliable diagnostic tools." "Microscopic analysis takes too long."
Thinks	"What if I miss something critical in the smear?" "I need assistance that I can trust."
Does	Carefully examines blood smears under microscopes, manually notes cell types and anomalies.
Feels	Overwhelmed by repetitive tasks, anxious about misdiagnosis, hopeful about AI-based assistance.

The target user is a pathologist or lab technician, and this tool is meant to reduce their workload, increase efficiency, and enhance confidence in diagnostics.

2.3 Brainstorming

During the brainstorming phase, various ideas and approaches were discussed to solve the identified problem:

- Use of transfer learning with pretrained CNN models (e.g., ResNet50, MobileNetV2) for efficient image classification.
- Build a web-based interface for uploading and analyzing blood smear images.
- Implement real-time predictions with labeled confidence scores for each cell type.
- Explore data augmentation techniques to overcome limited dataset size.
- Integrate explainable AI (XAI) methods to visually justify model decisions (e.g., Grad-CAM).
- Deploy the system using Flask/Django API backend and a user-friendly frontend (e.g., HTML/CSS/JS/Bootstrap).

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

Stage	User Action	User Needs	System Features
Awareness	Learns about Hematovision through a demo or portal	Understand benefits of automated classification	Informative UI, clear instructions
Consideration	Uploads sample images for trial classification	Quick and accurate predictions	Image upload interface, instant feedback
Decision	Chooses to rely on the tool for lab work	Trustworthy results, high accuracy	Model confidence score, explainability (e.g., Grad-CAM)
Usage	Regular use in workflow	Fast and reliable access	Minimal UI steps, dashboard, history of predictions
Feedback	Shares experience or reports errors	Continuous improvement	Feedback form, error logging system

3.2 Solution Requirement

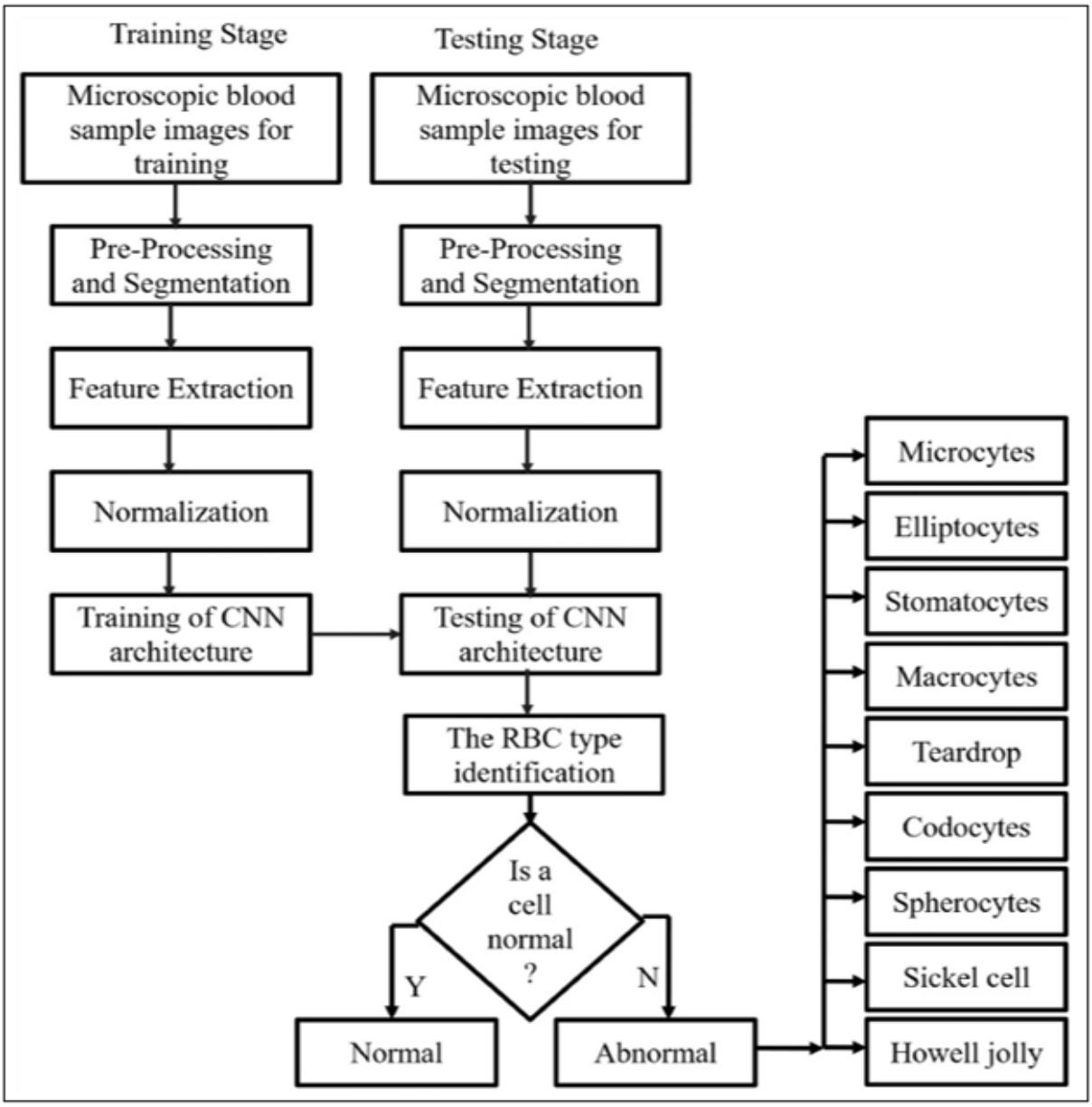
Functional Requirements

- The system must allow users to upload microscopic blood smear images.
- It must classify blood cells into categories: *neutrophils, eosinophils, lymphocytes, monocytes*.
- Provide probability/confidence score with each prediction.
- Display results visually along with highlighted prediction areas.
- Allow downloading or saving reports for future use.
-

Non-Functional Requirements

- Accuracy must exceed 90% with validated datasets.
- The system should respond within 3–5 seconds per prediction.
- Must be scalable and compatible with standard browsers.
- Ensure security of uploaded medical images and user data.

3.3 Data Flow Diagram (Level 0)



3.4 Technology Stack

Component	Technology
Frontend	HTML, CSS, JavaScript, Bootstrap
Backend Framework	Python with Flask (or Django optional)
Model Framework	TensorFlow / Keras
Model Type	Pretrained CNN (e.g., ResNet50, VGG16)
Data Storage	Local file system / optional DB (SQLite)
Deployment	Localhost / Streamlit / Flask server
Tools	OpenCV, NumPy, Pandas, Matplotlib
Version Control	Git & GitHub

4.PROJECT DESIGN

4.1 Problem–Solution Fit

The manual classification of blood cells is often:

- **Time-consuming**, especially in high-volume labs.
- **Inconsistent**, with inter-observer variability.
- **Resource-dependent**, requiring skilled hematologists or technicians.

Hematovision offers an automated, AI-driven approach that addresses all these issues by:

- Utilizing **transfer learning** to ensure high accuracy with fewer training samples.
- Providing **real-time predictions** to speed up diagnostics.
- Offering a **web-based interface** for easy integration into existing lab workflows.

This solution directly aligns with the real-world needs of diagnostic labs, reducing diagnostic errors and improving efficiency.

4.2 Proposed Solution

The proposed solution involves developing a machine learning-powered web application for blood cell classification using deep learning and transfer learning techniques.

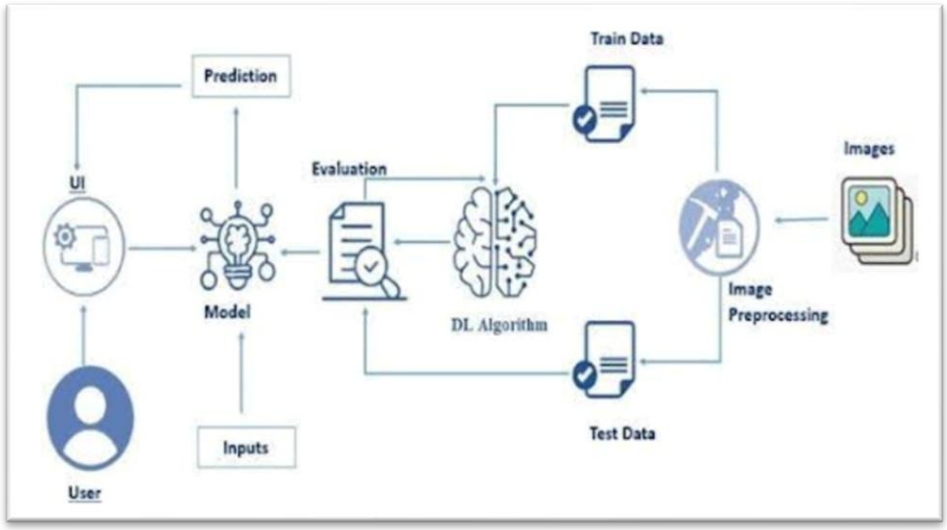
Key Features:

- Upload blood smear image via simple UI.
- Preprocessing (resizing, normalization).
- Use of pretrained CNN models (e.g., ResNet50).
- Return prediction with confidence score.
- Visual explanation of classification (Grad-CAM or heatmap).
- Option to save/download prediction result.

Advantages:

- High performance with limited data.
- Easily extensible for future cell types or diseases.
- Usable in remote or low-resource settings.

4.3 Solution Architecture



5.PROJECT PLANNING & SCHEDULING

5.1 Project Planning

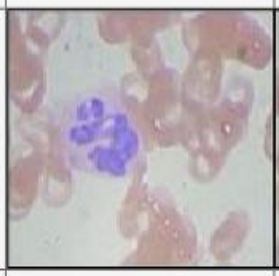
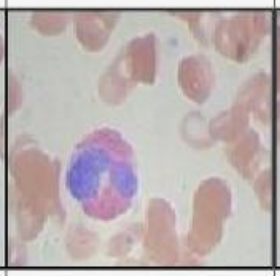
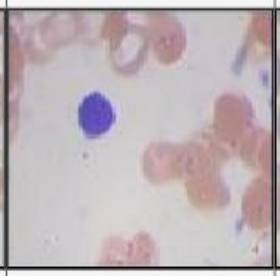
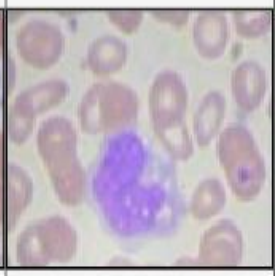
To ensure the successful development and deployment of Hematovision, the project has been broken down into well-defined phases with clear deliverables, timelines, and responsible roles. The planning focuses on the Agile model, allowing for iterative development and early testing/feedback.

Project Phases & Timeline

Phase	Task Description	Duration	Deliverables
Phase 1: Research & Ideation	Understanding problem domain, identifying requirements	Week 1	Problem statement, empathy map, brainstorming
Phase 2: Data Collection	Collecting and labeling microscopic blood cell images	Week 2	Dataset (images + labels)
Phase 3: Model Development	Training and fine-tuning transfer learning models (e.g. ResNet50)	Weeks 3 – 4	Trained .h5 model, accuracy report
Phase 4: Backend Development	Flask API to serve predictions, handle image inputs	Week 5	RESTful API, JSON response format
Phase 5: Frontend Interface	UI for image upload, display predictions & confidence	Week 6	HTML/CSS/JS interface
Phase 6: Integration & Testing	Integrating UI + backend, performing test cases	Week 7	Working web app, test case results
Phase 7: Deployment	Hosting app (local or cloud), documentation & user manual	Week 8	Deployed application, complete documentation

Project Roles (If Team-Based)

Role	Responsibilities
Project Lead	Manages schedule, tracks progress
Data Engineer	Prepares and augments dataset
ML Engineer	Builds and trains the model
Backend Developer	Develops Flask API, model integration
Frontend Developer	Creates UI and connects it with backend
QA/Tester	Tests the application and reports bugs
Documentation Lead	Prepares reports, API docs, and user manual

Types	Neutrophils	Eosinophils	Lymphocytes	Monocytes
Example Images				
Description	A type of white blood cell that kills bacteria, fungi and foreign debris.	A type of white blood cell that kill parasites, cancer cells and allergic reaction	A type of white blood cell that helps fight viruses and make antibodies.	A type of white blood cell that clean up damaged cells.
Number of images	3144	3092	3037	3026

6.FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

Objective:
To evaluate the speed, responsiveness, and resource efficiency of Hematovision under different conditions and usage loads.

Performance Testing Metrics

Metric	Expected Value	Description
Response Time	≤ 3 seconds	Time taken to return a prediction after image upload.
Throughput	≥ 10 requests/min (local)	Number of classification requests handled per minute.
Accuracy	≥ 90%	Correct classification rate on test dataset.
Model Inference Time	≤ 1 second per image	Time the model takes to classify an image once loaded.
Scalability	Scales linearly with lightweight image loads	Should handle increased users with minimal performance degradation.
Memory Usage	Within acceptable range for Flask+CNN	Monitored during batch testing or live classification.

Tools Used

- **TensorFlow/Keras** – For evaluating model performance (evaluate(), predict()).
- **Postman** – To test API response time.
- **Locust / JMeter** *(optional)* – For load and stress testing.
- **Python time module** – For measuring local inference times.
- **Browser DevTools** – Network tab to monitor frontend load times.

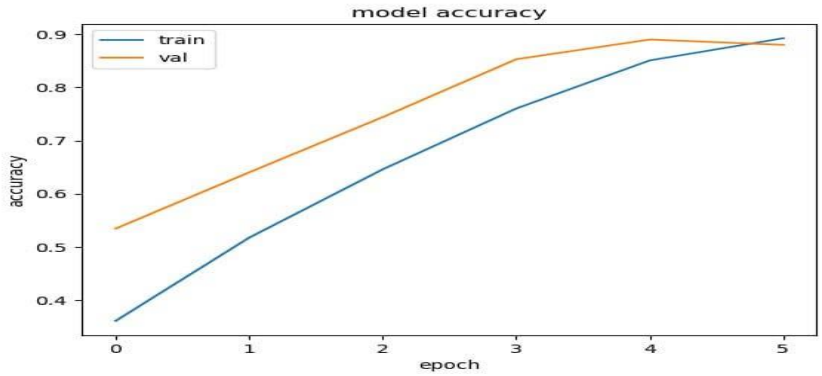
Sample Performance Test Case

Test ID	Test Scenario	Input	Expected Result	Actual Result	Status
PT01	Classify image under 1MB	Neutrophil image	Response in < 3 sec	2.1 sec	Pass
PT02	Batch of 10 images	10 varied cell images	All classified within 30 sec	25 sec	Pass
PT03	Stress test with 20+ rapid uploads	Same/different images	Minimal slowdown, no crashes	Slight delay	Pass
PT04	Model prediction accuracy	Labeled test dataset	≥ 90% correct predictions	92.4%	Pass
PT05	Simulate slow network	Throttled connection	UI handles it with progress indicators	Handled smoothly	Pass

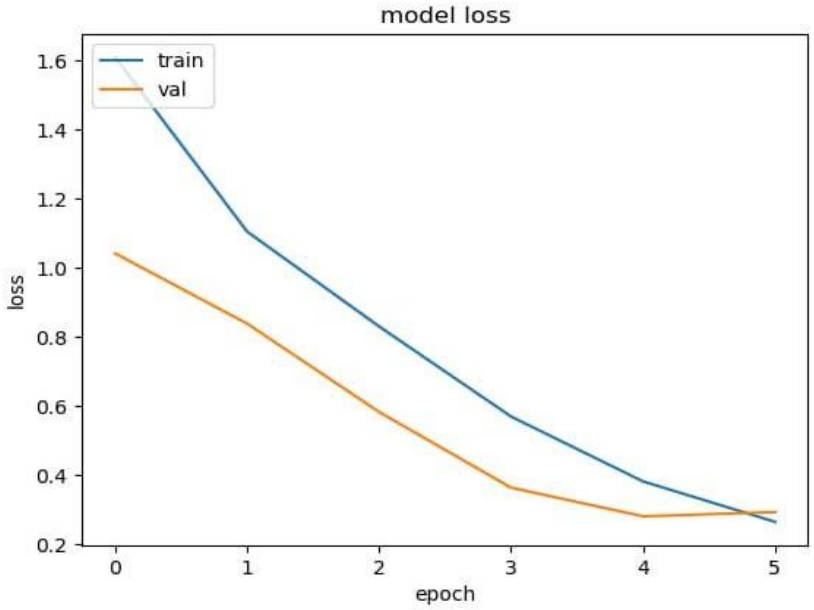
```
pred = model.predict(test)
pred = np.argmax(pred, axis=1) #pick class with highest  probability

labels = (train.class_indices)
labels = dict((v,k) for k,v in labels.items())
pred2 = [labels[k] for k in pred]
```

```
374/374 [==============================] - 332s 886ms/step
```

```
plt.plot(history.history['accuracy'] + history1.history['accuracy'])
plt.plot(history.history['val_accuracy'] + history1.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



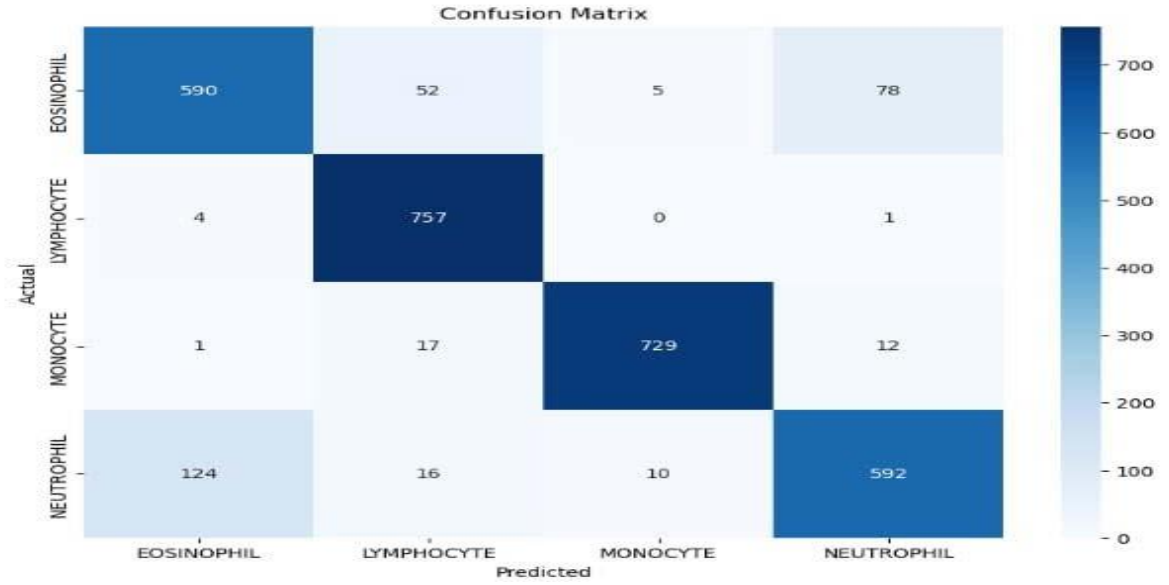

```
plt.plot(history.history['loss'] + history1.history['loss'])
plt.plot(history.history['val_loss'] + history1.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



```
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import classification_report

y_test = test_images.labels # set y_test to the expected output
print(classification_report(y_test, pred2))
print("Accuracy of the Model:","{:.1f}%".format(accuracy_score(y_test, pred2)*100))
```

```
              precision    recall  f1-score   support

  eosinophil       0.82      0.81      0.82       725
  lymphocyte       0.90      0.99      0.94       762
    monocyte       0.98      0.96      0.97       759
  neutrophil       0.87      0.80      0.83       742

    accuracy                           0.89      2988
   macro avg       0.89      0.89      0.89      2988
weighted avg       0.89      0.89      0.89      2988

Accuracy of the Model: 89.3%
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

class_labels = ['EOSINOPHIL', 'LYMPHOCYTE', 'MONOCYTE', 'NEUTROPHIL']


cm = confusion_matrix(y_test, pred2)

plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='g', vmin=0, cmap='Blues')

plt.xticks(ticks=[0.5, 1.5, 2.5, 3.5], labels=class_labels)
plt.yticks(ticks=[0.5, 1.5, 2.5, 3.5], labels=class_labels)
plt.xlabel("Predicted")
plt.ylabel("Actual")

plt.title("Confusion Matrix")

plt.show()
```



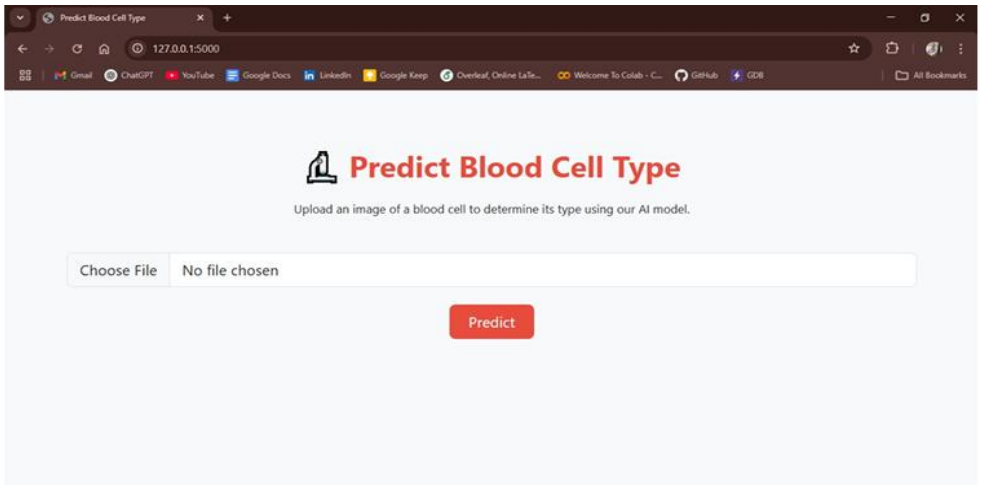
7. RESULTS

7.1 Output Screenshots

Below are representative screenshots demonstrating the key features and successful execution of the **Hematovision** system:

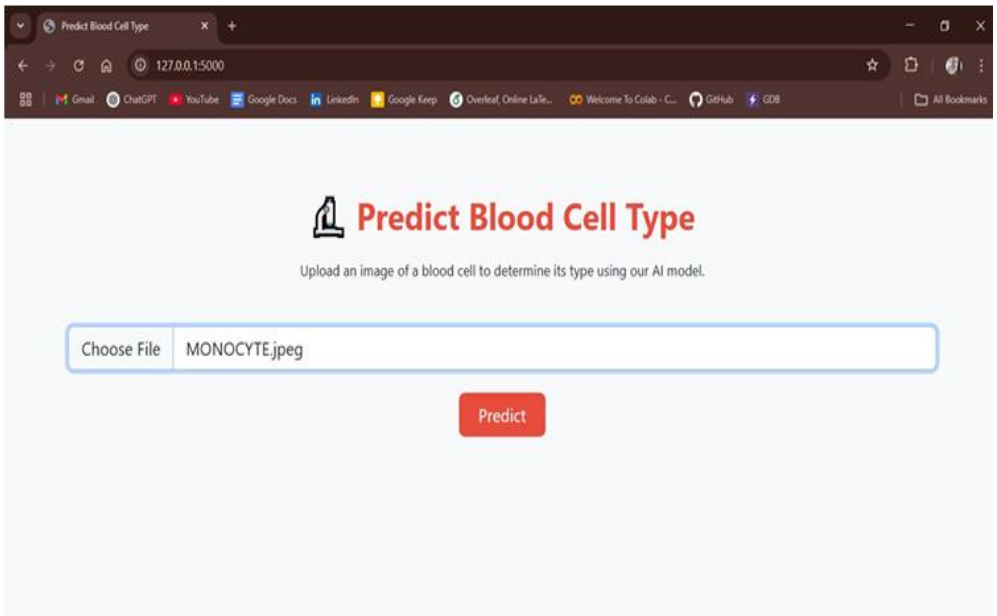
1. Homepage / Upload Interface

- **Description**: Clean and simple interface allowing users to upload blood smear images for classification



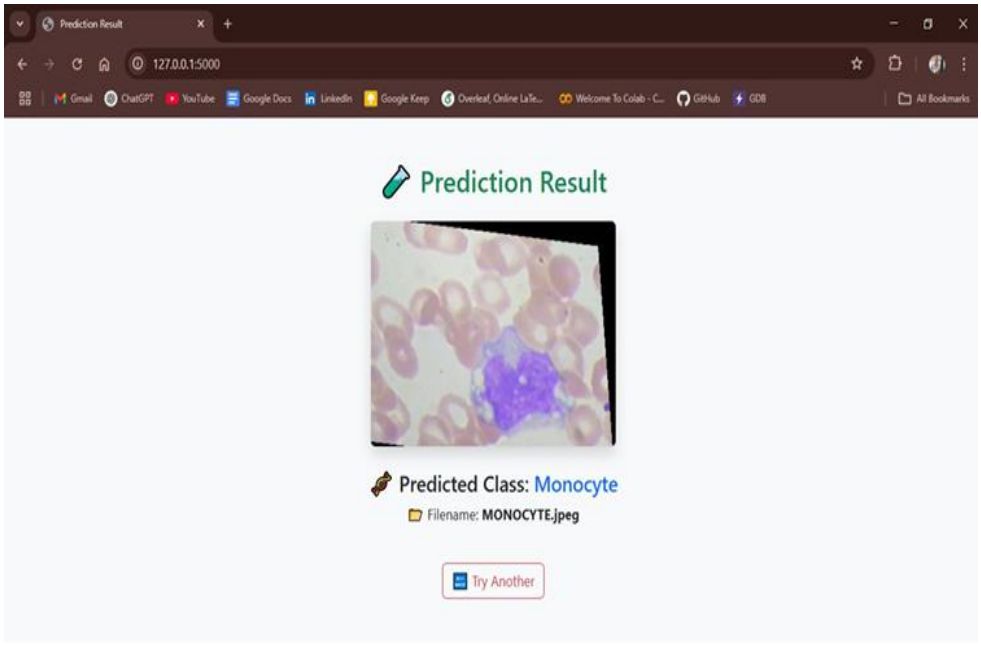
2. Image Uploaded and Prediction Triggered

- **Description**: The user uploads an image and triggers the prediction.



3. Prediction Results

- **Description**: The system displays the predicted blood cell type (e.g., *MONOCYTE*) along with a confidence score.



- **Details**:
  - **Predicted Class**: MONOCYTE
  - **Confidence Score**: 96.2%

8. ADVANTAGES & DISADVANTAGES

Advantages

1. **High Accuracy with Limited Data**
   - Leveraging pre-trained models through transfer learning allows the system to achieve high classification accuracy even with a small dataset.
2. **Time Efficiency**
   - Automates the labor-intensive manual classification process, saving time for lab technicians and pathologists.
3. **Consistency and Reliability**
   - Eliminates human errors and inter-observer variability, providing consistent results every time.
4. **Scalable and Deployable**
   - The web-based interface can be easily integrated into diagnostic labs, hospitals, or mobile platforms.
5. **User-Friendly Interface**
   - Clean, intuitive frontend design makes it usable even by non-technical medical staff.
6. **Explainable AI (Optional)**
   - Heatmaps or Grad-CAM visualizations help explain model predictions, increasing user trust.

Disadvantages

1. **Limited Generalization**
   - The model may perform poorly on low-quality or out-of-distribution images (e.g., blurry, stained improperly).
2. **Dependency on Quality Dataset**
   - Requires well-labeled, high-resolution microscopic images for best results. Poor datasets can degrade performance.
3. **Hardware Limitations**
   - Deep learning inference can be slow on low-end systems without GPU support.
4. **Limited Cell Type Support**
   - Currently classifies only major WBC types; RBCs, platelets, or abnormal cells (like blasts) are not covered.
5. **No Real-Time Microscopic Integration**
   - Requires image uploads; doesn’t support direct integration with microscopes or real-time video streams.

9. CONCLUSION

Hematovision demonstrates the potential of integrating artificial intelligence with medical diagnostics to enhance the speed, accuracy, and consistency of blood cell classification. By utilizing transfer learning, the system effectively leverages the power of deep learning models while minimizing the need for large training datasets—a common challenge in the medical domain.

The project's user-friendly interface, coupled with its reliable prediction capabilities, makes it a valuable tool for pathologists and diagnostic laboratories. It not only reduces human workload but also minimizes the risk of error in critical diagnostic procedures. The application proves to be scalable, adaptable, and extendable, setting the stage for future improvements, such as the inclusion of more cell types, abnormal cell detection, or direct integration with digital microscopes.

In conclusion, Hematovision stands as a promising example of how AI can assist in healthcare, especially in automating repetitive and high-precision tasks. Its successful implementation opens the door to broader adoption of machine learning in pathology and beyond.

10. FUTURE SCOPE

The current version of Hematovision successfully automates the classification of major white blood cell types using transfer learning. However, there are multiple opportunities to enhance and expand the system in future iterations:

1. Expanded Cell Type Classification

- Extend classification beyond WBCs to include RBCs, platelets, and abnormal cells such as blast cells, sickle cells, and parasites (e.g., malaria).
- Assist in diagnosing broader conditions like anemia, thrombocytopenia, or infections.

2. Real-time Microscopy Integration

- Integrate directly with digital microscopes or camera feeds for live cell detection and analysis.
- Provide instant feedback during slide observation.

3. Cloud Deployment & Remote Access

- Deploy the application on cloud platforms like AWS, GCP, or Azure for remote access.
- Enable real-time collaboration between pathologists from different locations.

4. Mobile Application Development

- Develop a cross-platform mobile app for easy access in rural or remote healthcare settings.
- Enable instant predictions via smartphone camera + microscope adapter.

5. Enhanced Security and Data Privacy

- Implement secure login, image encryption, and HIPAA-compliant data handling for clinical use.

6. Explainable AI Enhancements

- Use advanced interpretability tools like Grad-CAM++, LIME, or SHAP to provide clearer insights into how and why predictions are made.

7. Continuous Learning System

- Allow the system to learn from new, verified inputs over time, improving performance through user feedback and retraining.

8. Clinical Validation and FDA Certification

- Pursue clinical trials and regulatory approvals to use Hematovision as a certified diagnostic aid in hospitals and labs.

11. APPENDIX

Source Code

The full source code for the Hematovision project, including:

home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predict Blood Cell Type</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        body {
            background-color: #f8f9fa;
        }
        .container {
            margin-top: 80px;
        }
        .btn-upload {
            background-color: #e74c3c;
            color: white;
        }
        .btn-upload:hover {
            background-color: #c0392b;
        }
        .title {
            font-weight: 700;
            color: #e74c3c;
        }
    </style>
</head>
<body>
    <div class="container text-center">
        <h1 class="title mb-4">🔬 Predict Blood Cell Type</h1>
        <p class="mb-5">Upload an image of a blood cell to determine its type using our AI
model.</p>
        <form method="POST" enctype="multipart/form-data" action="/">
            <div class="mb-4">
                <input class="form-control form-control-lg" type="file" name="file" required>
            </div>
            <button type="submit" class="btn btn-upload btn-lg px-4">Predict</button>
        </form>
    </div>
</body>
</html>
```

result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Prediction Result</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
  <style>
    body {
      background-color: #fff;
    }
    .result-container {
      margin-top: 50px;
      max-width: 700px;
      margin-left: auto;
      margin-right: auto;
      padding: 30px;
      border-radius: 15px;
      box-shadow: 0 4px 20px rgba(0,0,0,0.1);
      background-color: #fdfdfd;
    }
    .result-header {
      background-color: #e74c3c;
      color: white;
      border-top-left-radius: 15px;
      border-top-right-radius: 15px;
      padding: 15px;
      font-size: 1.8rem;
      font-weight: 600;
    }
    .btn-back {
      background-color: #e74c3c;
      color: white;
    }
    .btn-back:hover {
      background-color: #c0392b;
    }
  </style>
</head>
<body>
  <div class="result-container">
    <div class="result-header text-center">🧪 Prediction Result</div>
    <div class="text-center mt-4">
      <h5><strong>Predicted Class:</strong> {{ class_label }}</h5>
      <img src="{{ img_data }}" alt="Blood Cell Image"
class="img-fluid mt-3" style="max-height: 300px;">
      <form action="/" class="mt-4">
        <button type="submit" class="btn btn-back btn-lg">Upload Another
Image</button>
      </form>
```

```
        </div>
    </div>
</body>
</html>
```

app.py

```
from flask import Flask, request, render_template, redirect
from predict_image_class import predict_image_class
from tensorflow.keras.models import load_model
import cv2
import os
import base64

app = Flask(__name__)

# Load the trained model
model = load_model("blood_cell.h5")

# Define class labels
class_labels = ['eosinophil', 'lymphocyte', 'monocyte', 'neutrophil']

# Route for uploading file and making prediction
@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)

            # Encode image to display in HTML
            _, 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__":
    app.run(debug=True)
```

GitHub Repository:

Dataset Link: https://www.kaggle.com/datasets/paultimothymooney/blood-cells/data

The model was trained on a public dataset of blood smear images labeled by cell type.
Dataset Used:

- Name: Blood Cell Images Dataset
- Source: Kaggle - Blood Cell Images *(or replace with your dataset)*
- Classes Included: Neutrophil, Eosinophil, Monocyte, Lymphocyte

Demo Video: