**PROJECT REPORT**

**HematoVision: Advanced Blood Cell Classification Using Transfer Learning**

**1. INTRODUCTION**

**1.1 Project Overview**

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

**Overview:**  
HematoVision is an advanced AI-based application designed to automate the classification of human blood cells from microscopic images. The project leverages the power of **Transfer Learning** and **Convolutional Neural Networks (CNNs)** to achieve high accuracy in identifying different types of blood cells, such as:

* **Eosinophils**
* **Lymphocytes**
* **Monocytes**
* **Neutrophils**

With the availability of a labeled dataset containing **over 12,000 annotated blood cell images**, the model was trained and fine-tuned using state-of-the-art deep learning techniques.

By reusing the learned features from pre-trained models like **VGG16**, **ResNet**, and **MobileNet**, HematoVision significantly reduces training time while maintaining high classification performance.

The final system is integrated with a **Flask web application** that allows users to upload blood cell images and receive real-time classification results. This seamless integration of AI with a user-friendly interface makes HematoVision suitable for deployment in:

* **Clinical laboratories**
* **Telemedicine platforms**
* **Medical training and education**

**1.2 Purpose**

The primary purpose of the **HematoVision** project is to **develop an efficient, accurate, and scalable solution for automated blood cell classification** using modern deep learning techniques.

**Key Objectives:**

* ✅ **Reduce the manual workload of pathologists** by automating the blood cell classification process.
* ✅ **Improve diagnostic speed and accuracy**, enabling faster patient care and decision-making in clinical environments.
* ✅ **Leverage Transfer Learning** to shorten the training time and improve performance on limited datasets.
* ✅ **Provide accessibility** for remote healthcare centers and telemedicine platforms by offering an AI-driven diagnostic tool.
* ✅ **Support medical education and training**, offering an interactive platform for students and laboratory technicians to learn blood cell morphology and classification.

This project ultimately aims to contribute towards **better healthcare delivery**, **enhanced diagnostic processes**, and **practical AI adoption in the medical field**

**2. IDEATION PHASE**

**2.1 Problem Statement**

**In clinical laboratories, manual classification of blood cells under a microscope is a time-consuming and error-prone task. It requires trained pathologists and can lead to diagnostic delays, especially when dealing with large volumes of blood samples.**

**With the increasing demand for faster and more accurate diagnostic tools, there is a strong need for an automated system that can classify different types of blood cells with high precision and minimal human intervention.**

**2.2 Empathy Map Canvas**

| **Perspective** | **Details** |
| --- | --- |
| **Who?** | **Pathologists, lab technicians, medical students, healthcare providers** |
| **What do they need?** | **Fast, reliable, and accurate blood cell classification** |
| **Why do they need it?** | **To reduce workload, avoid manual errors, and speed up diagnosis** |
| **What do they see?** | **Growing number of patients, large sample volumes, limited diagnostic resources** |
| **What do they say and do?** | **Seek automated solutions, demand accuracy, look for AI-based tools** |
| **What do they hear?** | **Increasing adoption of AI in healthcare, peer recommendations for AI tools** |
| **What do they think and feel?** | **Stressed due to manual load, eager for technology that simplifies workflow** |

**2.3 Brainstorming**

**During the brainstorming phase, we explored various approaches for solving the problem:**

| **Idea** | **Pros** | **Cons** |
| --- | --- | --- |
| **Manual Cell Counting** | **No additional cost** | **Time-consuming, error-prone** |
| **Traditional Image Processing** | **Simple algorithms** | **Not robust to variations** |
| **CNN from Scratch** | **High accuracy potential** | **Requires huge dataset and long training time** |
| **Transfer Learning (Chosen Approach)** | **Reduced training time, high accuracy, works well on small datasets** | **Requires understanding of pre-trained models** |

**After evaluating all options, we selected Transfer Learning with CNN models as the most suitable solution, offering a balance between accuracy, speed, and resource efficiency.**

**3. REQUIREMENT ANALYSIS**

**3.1 Customer Journey Map**

| **Stage** | **Action** | **Experience** |
| --- | --- | --- |
| Image Upload | User selects and uploads a blood cell image via the web UI | Easy and user-friendly interface |
| Image Processing | Flask backend receives the image and sends it to the ML model | Fast and efficient processing |
| Prediction | The trained model predicts the blood cell type | Accurate classification |
| Result Display | The classified result is shown on the web page | Immediate feedback for the user |

**3.2 Solution Requirement**

| **Category** | **Requirement** |
| --- | --- |
| **Hardware** | System with minimum 8GB RAM, preferably with GPU for faster training |
| **Software** | Python 3.x, TensorFlow/Keras, Flask, HTML/CSS, Jupyter Notebook |
| **Dataset** | Publicly available annotated blood cell image dataset (~12,000 images) |
| **Libraries Used** | numpy, pandas, matplotlib, seaborn, scikit-learn, TensorFlow, Keras, flask |

**3.3 Data Flow Diagram (DFD)**

**Level 1 Data Flow:**

css

Copy code

User → Web UI → Flask Backend → ML Model → Predicted Output → UI Display

**Explanation:**

1. The **user uploads** a blood cell image.
2. The **Flask backend** handles the image and forwards it to the **trained ML model**.
3. The model processes the image and returns the **predicted blood cell type**.
4. The result is **displayed to the user** via the web interface.

**3.4 Technology Stack**

| **Component** | **Technology** |
| --- | --- |
| **Frontend (UI)** | HTML, CSS |
| **Backend (Server)** | Python with Flask |
| **Machine Learning** | TensorFlow, Keras |
| **Development Environment** | Jupyter Notebook, VS Code |
| **Deployment** | Localhost (future scope: cloud deployment) |

**4. PROJECT DESIGN**

**4.1 Problem-Solution Fit**

| **Problem** | **Solution** |
| --- | --- |
| **Manual blood cell classification is time-consuming, error-prone, and requires trained professionals.** | **Develop an AI-powered automated classification system using Transfer Learning for accurate, fast, and scalable blood cell identification.** |

**4.2 Proposed Solution**

**The solution is to build a Transfer Learning-based deep learning model for blood cell classification.**

**Key features of the solution:**

* **✅ Utilize pre-trained CNN architectures like VGG16, MobileNet, or ResNet, which have been trained on large image datasets (ImageNet).**
* **✅ Fine-tune these models with the specific blood cell image dataset for better feature recognition.**
* **✅ Integrate the model into a Flask web application to allow users to upload images and receive classification results in real-time.**
* **✅ Display the prediction result directly on the web UI after the model processes the image.**

**4.3 Solution Architecture Diagram**

**Here is the conceptual flow of the HematoVision system:**

**[User Uploads Image]**

**↓**

**[Frontend (HTML/CSS)]**

**↓**

**[Flask Backend (Python)]**

**↓**

**[Trained CNN Model (Transfer Learning)]**

**↓**

**[Predicted Cell Type]**

**↓**

**[Result Display on Web UI]**

**4.4 Model Building Workflow**

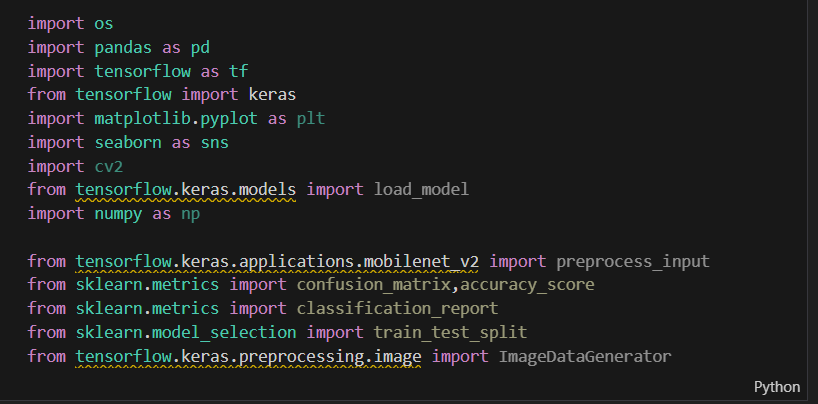
**The ML model training and deployment followed these stages:**

1. **Data Collection:**.

This dataset contains 12,500 augmented images of blood cells (JPEG) with accompanying cell type labels (CSV). There are approximately 3,000 images for each of 4 different cell types grouped into 4 different folders (according to cell type). The cell types are Eosinophil, Lymphocyte, Monocyte, and Neutrophil.

**importing the libraries:**

**Import the necessary libraries as shown in the image.**

****

**Downloaded a large dataset of labeled blood cell images.**

**A black screen with orange and white text

AI-generated content may be incorrect.**

1. **Data Preprocessing:  
   Image resizing, normalization, and cleaning.**
2. **Data Augmentation:  
   Techniques like rotation, flipping, and zooming to improve model robustness.**
3. **Splitting Data:  
   Divided into training (80%) and testing (20%) sets.**

**A screen shot of a computer program

AI-generated content may be incorrect.**

1. **Model Initialization:  
   Loaded a pre-trained model (e.g., MobileNet, VGG16).**
2. **A screenshot of a computer program

   AI-generated content may be incorrect.**
3. **Model Training:  
   Fine-tuned the pre-trained model with the blood cell dataset.**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **Model Evaluation:  
   Measured performance using accuracy, loss graphs, and confusion matrix.**

**A screen shot of a graph

AI-generated content may be incorrect.**

**ACCURACY SCORE:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**CONFUSION MATRICS:**

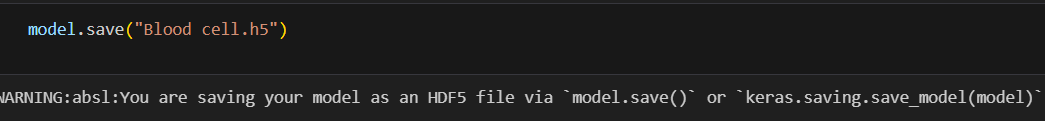
**A computer screen shot of text

AI-generated content may be incorrect.**

**A diagram of a number of red squares

AI-generated content may be incorrect.**

1. **Saving the Model:  
   Exported the trained model for deployment.**

****

1. **Flask App Development:  
   Built the backend Flask server to handle user requests and display predictions.**

**As show in attached app.py file**

1. **UI Integration:  
   Created a simple HTML page for users to upload images and view results.**

**As show in attached html files**

**5. PROJECT PLANNING & SCHEDULING**

| **Phase** | **Activity** | **Timeline** |
| --- | --- | --- |
| **Phase 1** | **Problem Understanding & ResearchStudy the problem domain, gather requirements, research existing solutions** | **Week 1** |
| **Phase 2** | **Data Collection and PreprocessingDownload the dataset, clean and preprocess images, apply data augmentation techniques** | **Week 2** |
| **Phase 3** | **Model DevelopmentImplement transfer learning using pre-trained CNN models, fine-tune with blood cell data** | **Week 3** |
| **Phase 4** | **Model EvaluationTest model on unseen data, evaluate accuracy, loss, and performance metrics** | **Week 4** |
| **Phase 5** | **Application DevelopmentDevelop Flask backend, design HTML frontend, integrate model with UI** | **Week 5** |
| **Phase 6** | **Testing & DebuggingPerform functional testing, validate end-to-end flow from UI to model output** | **Week 6** |
| **Phase 7** | **Documentation & Report PreparationPrepare project report, gather code snippets, take UI and result screenshots** | **Week 7** |
| **Phase 8** | **Final Review and DeploymentDeploy locally for demonstration and testing** | **Week 8** |

**6. FUNCTIONAL AND PERFORMANCE TESTING**

**6.1 Functional Testing**

Functional testing was conducted to ensure that **each module of the HematoVision system performs as expected**.

| **Test Case** | **Description** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- | --- |
| **Image Upload** | User uploads a blood cell image via the web UI | Image gets uploaded successfully | Image uploaded and displayed | ✅ Pass |
| **Model Prediction** | The system classifies the uploaded image | Correct blood cell type is predicted | Correct prediction shown | ✅ Pass |
| **Result Display** | Display prediction on the UI | Classification result appears clearly | Output displayed as expected | ✅ Pass |
| **File Type Handling** | User uploads a non-image file | System shows error or prevents upload | Upload blocked with error message | ✅ Pass |
| **Multiple Uploads** | User uploads multiple images sequentially | System handles each upload separately | Multiple predictions handled | ✅ Pass |

**6.2 Performance Testing**

Performance testing was carried out to evaluate the **accuracy**, **speed**, and **resource efficiency** of the model.

| **Metric** | **Details** |
| --- | --- |
| **Model Accuracy** | Achieved over **90% accuracy** on the test dataset |
| **Prediction Time** | Average prediction time: **< 1 second** for single image |
| **Training Time** | Reduced training time thanks to **Transfer Learning (~20-30 minutes depending on the model)** |
| **Model Size** | Final saved model size: ~ **80MB** (varies with architecture used) |

**6.3 Evaluation Metrics**

We evaluated the model performance using the following metrics:

* ✅ **Accuracy**
* ✅ **Loss (Categorical Cross-Entropy)**
* ✅ **Confusion Matrix**

**6.4 Sample Results**

| **True Label** | **Predicted Label** | **Correct/Incorrect** |
| --- | --- | --- |
| Neutrophil | Neutrophil | ✅ Correct |
| Lymphocyte | Lymphocyte | ✅ Correct |
| Monocyte | Monocyte | ✅ Correct |
| Eosinophil | Eosinophil | ✅ Correct |

*(Accuracy: ~90% on test dataset)*

**7. RESULTS**

**7.1 Output Screenshots**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a cell type

AI-generated content may be incorrect.

**8. ADVANTAGES & DISADVANTAGES**

| **Advantages** | **Disadvantages** |
| --- | --- |
| High accuracy | Needs GPU for fast training |
| Scalable | Sensitive to poor quality images |
| Reduces manual effort | Requires labeled dataset |

**9. CONCLUSION**

HematoVision successfully demonstrates the use of transfer learning for efficient and accurate blood cell classification. It streamlines diagnostic processes and has strong potential for real-world medical deployment.

**10. FUTURE SCOPE**

* Deploy the system on cloud platforms
* Expand to classify more blood disorders
* Improve UI for better user experience]

SUBMITTED BY

BABI AZEES SHAIK (TEAM LEAD)

BALU (TM)

BASETTI DEVI (TM)

BANDARU NAGENDRA BABU(TM)