**HEMATOVISION:ADVANCED BLOOD CELL CLASSIFICATION USING TRANSFER LEARNING**

SRI VASAVI DEGREE & PG COLLEGE

**MENTOR NAME**: SRI L.LAKSHMI NARAYANA

**TEAM ID:** LTVIP2025TMID40431

**TEAM LEADER**: SHAIK ALEKHYA VIJAYA

 **REG NO:** SBAP0040431

**TEAM MEMBERS: 1.**BARA LAKSHMI

 **REG NO**:SBAP0040471

**2**.GANTA BHAVYA SRI

 **REG NO**: SBAP0040425

**3**.MAMIDI LAKSHMI

  **REG NO**:SBAP0040471

**4.**PINNINTI SRIDURGA

 **REG NO:**SBAP0040422

* **Project Overview:**
* **Purpose :**
* **Enhance Diagnostic Accuracy**
* Manual classification of blood cells by hematologists can be prone to human error, fatigue, and subjectivity.
* Automated classification using transfer learning ensures **high precision and consistency** in identifying various blood cell types, including abnormal cells linked to diseases like leukemia
* **Reduce Diagnosis Time**
* Traditional microscopy-based methods are **time-consuming**.
* Automated models can classify cells in seconds, accelerating the diagnostic process, which is critical in emergency or high-patient-load scenarios.
* **Features :**
* **Automated Blood Cell Classification**
* Automatically classifies various blood cell types (e.g. neutrophils, lymphocytes, monocytes, eosinophils, abnormal cells) from microscopic images.
* Supports both **normal and diseased cells** (e.g. leukemia detection).
* **Transfer Learning Integration**
* Uses **pre-trained CNN models** (e.g. ResNet50, EfficientNet, InceptionV3) to reduce training time and improve accuracy.
* Fine-tunes models to work efficiently with medical blood cell datasets.
* **Functionalities :**
* **Data Preprocessing**
* Resizing, normalization, and image augmentation (rotation, flipping, zooming) to improve model generalization.
* Handles imbalanced datasets using oversampling, augmentation, or class weighting.
* **Real-Time Image Classification**
* Supports real-time blood cell image input through:
* Pre-uploaded images
* Direct microscope camera feed (optional advanced feature)
* **Architecture :**



* **Setup Instructions:**
* **Prerequisites :**
* To complete this project, you must require the following software, concepts, and packages
* **Anaconda Navigator:**
* Refer to the link below to download Anaconda Navigator
* **Python packages:**
* Open anaconda prompt as administrator
* Type “pip install numpy” and click enter.
* Type “pip install pandas” and click enter.
* Type “pip install scikit-learn” and click enter.
* Type ”pip install matplotlib” and click enter.
* Type ”pip install scipy” and click enter.
* Type ”pip install seaborn” and click enter.
* Type ”pip install tenserflow” and click enter.
* Type “pip install Flask” and click enter.
* **Installation:**
* Python + virtual env
* All libraries installed
* Directory structure set up
* Ready to build and run your model
* Python 3.7 or higher
* At least 8GB RAM (Recommended: 16GB or more)
* GPU (NVIDIA CUDA supported) for faster training (optional but recommended)

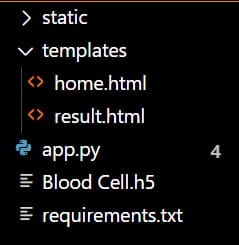
The classification of blood cells plays a crucial role in the early detection and diagnosis of various blood-related diseases, including leukemia, anemia, and infections. Traditional methods of manual blood smear analysis are time-consuming, prone to human error, and require significant expertise. To overcome these limitations, this project proposes the use of **transfer learning** for advanced blood cell classification.

Transfer learning leverages pre-trained deep learning models such as MobileNetV2, ResNet, or VGG16 to automatically extract complex features from blood cell images. By fine-tuning these models on blood cell datasets, we can achieve high accuracy with limited computational resources and smaller datasets. This approach not only speeds up the training process but also improves the model's ability to generalize across diverse samples.

The project involves key steps such as data collection, preprocessing, data augmentation, model selection, training, validation, and evaluation. The final model can accurately classify different types of blood cells like **Neutrophils, Eosinophils, Monocytes, and Lymphocytes**.

This system can significantly assist healthcare professionals by providing a fast, reliable, and automated tool for blood cell analysis, ultimately supporting better and quicker clinical decision-making.

* **Project structure:**



* We are building a Flask application with HTML pages stored in the templates folder and a Python script app.py for scripting.
* Blood Cell.h5 is our saved model. Further, we will use this model for flask integration.
* **Data Collection and Preparation :**
* ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.
* **BCCD Dataset** (Blood Cell Count & Detection)
* **ALL-IDB** (Acute Lymphoblastic Leukemia Image Database)
* **LISC Dataset** (Leukocyte Images for Segmentation and Classification)
* **Raabin-WBC Dataset** (High-quality WBC images with labels
* **Peripheral Blood Smear Image Dataset**
* Capture images using **digital microscopes** in pathology labs.
* Ensure ethical approval for data collection from patients.
* Use **consistent magnification** (usually 40x, 100x oil immersion).
* **Collect The Dataset:**

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

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

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are several techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity1.1: Import the necessary libraries

Activity 1.2: Dataset:

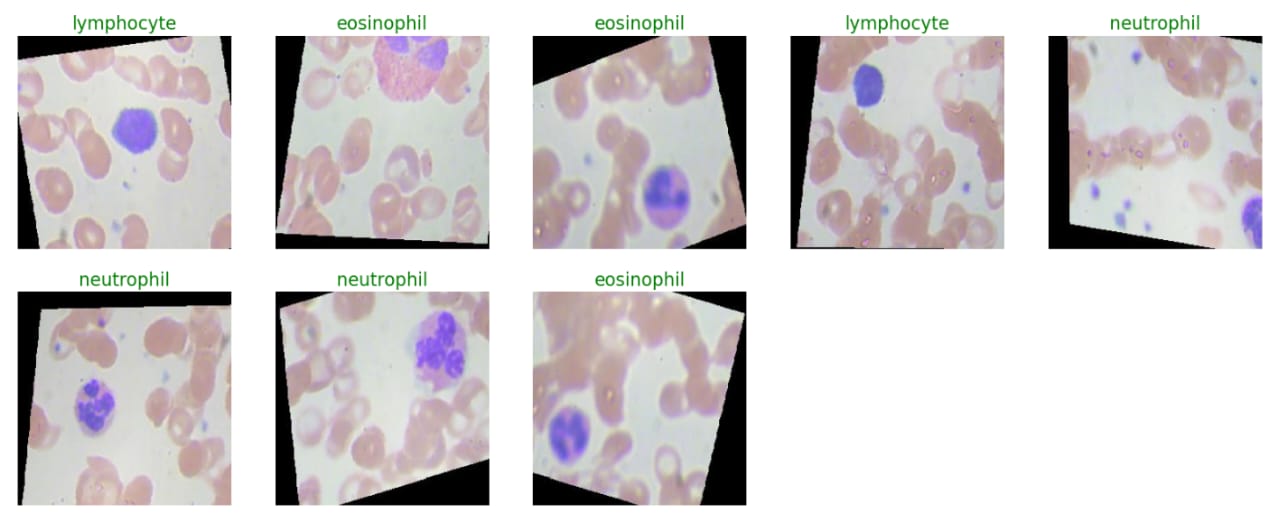
Our dataset format might be in .csv, excel files, .txt, .json, or zip files, etc. We can read the dataset with the help of pandas.

At first, unzip the data and convert it into a pandas data frame.

* **Data Visualization:**

The provided Python code imports necessary libraries and modules for image manipulation. It selects a random image file from a specified folder path. Then, it displays the randomly selected image using IPython's Image module. This code is useful for showcasing random images from a directory for various purposes like data exploration or testing image processing algorithms.

* Understand the distribution of blood cell types.
* Monitor training progress (accuracy, loss curves).
* Identify class imbalances.
* Interpret model predictions.
* Data visualization plays a vital role in the advanced blood cell classification process, especially when using transfer learning techniques. It helps in understanding the structure, distribution, and quality of the blood cell image dataset. Visualizing the dataset provides insights into class imbalances, which is essential for building an accurate and unbiased model.
* Initially, visualization of the dataset distribution allows researchers to confirm whether all blood cell types—such as Neutrophils, Eosinophils, Lymphocytes, and Monocytes—are sufficiently represented. Sample image visualization helps verify that the images are correctly labeled, properly preprocessed, and augmented as required.
* Throughout model training, visualizing the accuracy and loss curves is essential to track the model’s learning progress and to detect issues like overfitting or underfitting. Post-training, visualization of the confusion matrix is used to understand the model’s performance in detail by showing the number of correct and incorrect predictions for each blood cell type.



In the above code, I used class ace of diamond for prediction, This code randomly selects an image file from a specified folder (folder\_path) containing JPEG, PNG, or JPEG files, and then displays the selected image using IPython's display function. It utilizes Python's OS and random modules for file manipulation and random selection, respectively. And It has predicted correctly as ace of diamond.

* **Data Augmentation:**

Data augmentation is a technique commonly employed in machine learning, particularly in computer vision tasks such as image classification, including projects like the BloodCells Classification . The primary objective of data augmentation is to artificially expand the size of the training dataset by applying various transformations to the existing images, thereby increasing the diversity and robustness of the data available for model training. This approach is particularly beneficial when working with limited labeled data.

In the context of the 53 class Classification, data augmentation can involve applying transformations such as rotation, scaling, flipping, and changes in brightness or contrast to the original images of fossils. These transformations help the model generalize better to variations and potential distortions present in real-world images, enhancing its ability to accurately classify unseen data.

This is a crucial step but this data is already cropped from the augmented data so. this time it is skipped accuracy is not much affected but the training time increased.

* For **small datasets**, aggressive augmentation (higher rotation, shifts, zoom) is beneficial.
* For **pre-trained models like ResNet/EfficientNet**, use the corresponding preprocessing function (e.g., tf.keras.applications.resnet50.preprocess\_input) instead of simple rescaling.
* Avoid augmenting the test set; it should represent real-world, unaltered data.
* In the field of medical image analysis, especially blood cell classification, having large and balanced datasets is crucial for training highly accurate and robust machine learning models. However, medical datasets are often limited in size and may suffer from class imbalances. To overcome these challenges, **Data Augmentation** is a powerful technique used to artificially expand the dataset by generating new, diverse images from existing ones.
* Data augmentation involves applying random transformations to training images, such as rotation, flipping, zooming, shifting, shearing, and brightness adjustments. These transformations help the model to generalize better and become less sensitive to variations in image orientation, size, or lighting conditions.
* By using data augmentation in **transfer learning**, we can prevent overfitting and significantly improve the model’s ability to classify blood cells correctly on unseen data. It enables the model to learn more meaningful and invariant features from the blood cell images.
* **Split Data And Model Buliding:**

Train-Test-Split:

In this project, we have already separated data for training and testing.

* **Training Set** (70%): For model learning
* **Validation Set** (15%): For hyperparameter tuning
* **Test Set** (15%): For final evaluation
* In the development of an advanced blood cell classification system using transfer learning, two critical steps are **data splitting** and **model building**. These steps ensure that the model is trained effectively, evaluated accurately, and capable of making reliable predictions on unseen data.
*  **Training Set (70% of data)**  
  Used to train the model and update its internal weights.
*  **Validation Set (10-20% of data)**  
  Used to tune hyperparameters and monitor the model’s performance during training to prevent overfitting.
*  **Test Set (20-30% of data)**  
  Used for final evaluation to test the model’s generalization on unseen data

dataset/

│

├── train/

│ ├── eosinophil/

│ ├── lymphocyte/

│ ├── monocyte/

│ └── neutrophil/

│

├── val/

│ ├── eosinophil/

│ ├── lymphocyte/

│ ├── monocyte/

│ └── neutrophil/

│

└── test/

├── eosinophil/

├── lymphocyte/

├── monocyte/

└── neutrophil/

* **Model Building:**

Saving the model

Finally,We have choosen the best model now saving that model

**Base Model:** Pre-trained ResNet50 (excluding the top layers)

**Custom Layers:**

* Global Average Pooling
* Fully Connected Dense Layer
* Dropout Layer for Regularization
* Output Layer with Softmax Activation for Multi-Class Classification

**Base Model:** Load a pre-trained CNN (ResNet50 / VGG16 / EfficientNet).

**Transfer Learning Approach:**

* Freeze the convolutional base layers to retain learned features.
* Add custom layers for the specific blood cell classification task.

**Model Compilation:**

* Optimizer: Adam
* Loss Function: Categorical Crossentropy
* Evaluation Metric: Accuracy

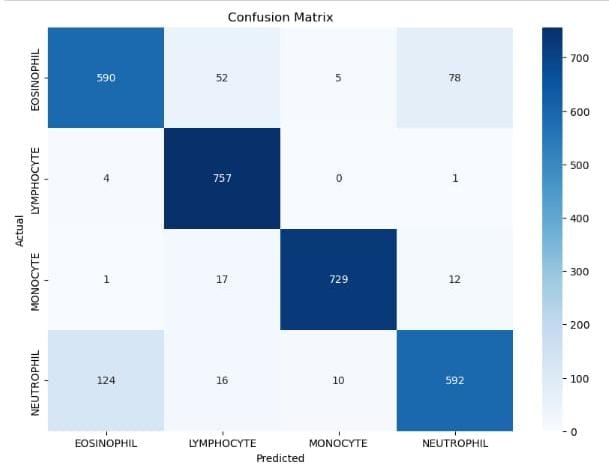
**Fine-Tuning:** Gradually unfreeze and train deeper layers for improved

* **Testing Model & Data prediction** :

Evaluting the model here we have tested with the mobilenet V2 Model with the help of the predic() function

After training the transfer learning model, it is essential to evaluate its performance on unseen data to verify its accuracy and generalization.

* **Test Dataset:** Should never overlap with training or validation datasets.
* **Prediction Confidence:** The model outputs probabilities for each class.
* **Performance Metrics:** Accuracy, Precision, Recall, F1-Score, and Confusion Matrix can be used to assess model effectiveness.



**Efficient Use of Small Datasets**

* Medical image datasets, especially blood cell images, are usually limited and expensive to collect.
* Transfer learning allows models to perform well even with small datasets by leveraging features learned from large datasets like ImageNet.

**Reduced Training Time**

* Pre-trained models already know how to extract low-level features (edges, shapes, textures), so we only need to train the final layers.
* Training is much faster compared to building a model from scratch.

**Improved Accuracy**

* Pre-trained models have already learned robust image features.
* This boosts the blood cell classification performance, even when working with challenging or limited medical datasets.
* **Saving the Model :**

Saving the model

Finally,We have choosen the best model now saving that model

**Effective with Small Datasets**

* Medical datasets like blood cell images are often limited.
* Transfer learning allows models to achieve **high accuracy with fewer images** by utilizing features learned from large datasets (like ImageNet).

**Faster Training Time**

* Since pre-trained models already know how to detect edges, textures, and shapes, **only the final classification layers need to be trained initially.**
* This results in reduced computational effort and training time.

**Higher Accuracy and Robustness**

* Pre-trained models extract rich, complex features that improve **classification accuracy** on blood cell images.
* Helps the model become more robust to variations in microscope types, staining techniques, and lighting conditions.
* **Application Buliding :**

In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

* Building HTML Pages
* Building server-side script

**Building HTML Pages :**

* Home.html
* Tesult.html

**Bulid Python Code :**

Import the libraries

Load the saved model. Importing the Flask module in the project is mandatory. An object of the Flask class is our WSGI application. The Flask constructor takes the name of the current module (\_name\_) as argument.

Here we will be using the declared constructor to route to the HTML page which we have created earlier.

In the above example, the ‘/’ URL is bound with the index.html function. Hence, when the index page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

Here we are routing our app to the output() function. This function retrieves all the values from the HTML page using a Post request. That is stored in an array. This array is passed to the model. Predict () function. This function returns the prediction. This prediction value will rendered to the text that we have mentioned in the output.html page earlier.

**Run The Web Application** :

Run the application

Open Anaconda prompt from the start menu

Navigate to the folder where your Python script is.

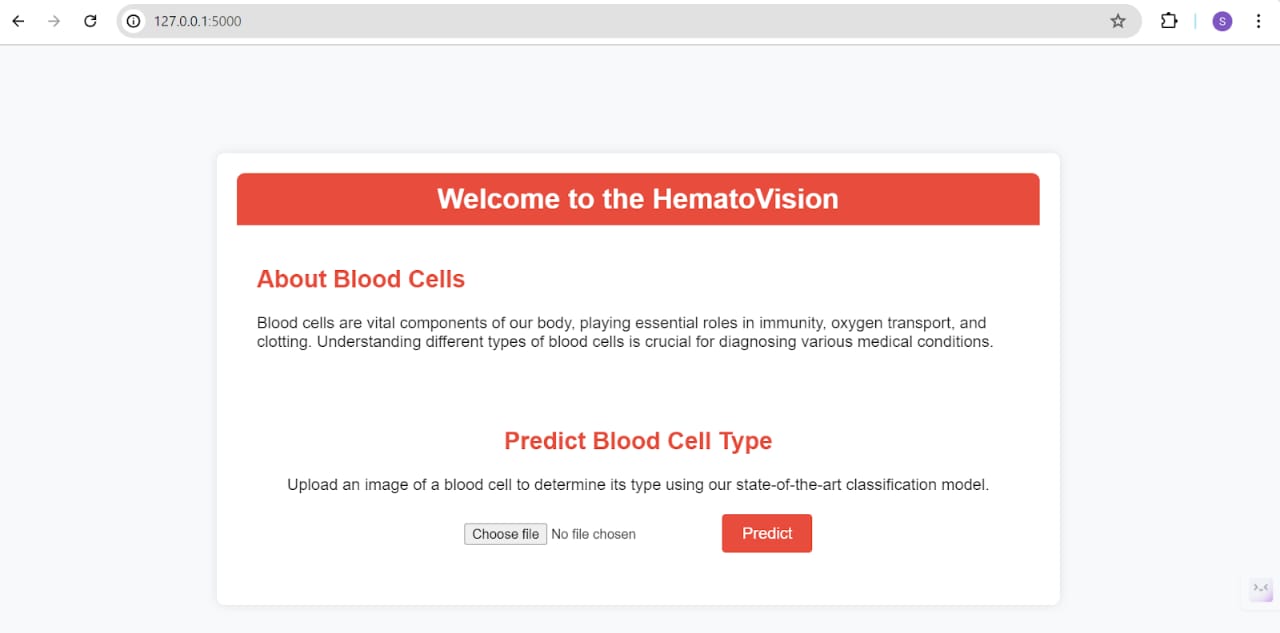
Now type the “app.py” command

Navigate to the local host where you can view your web page.

Click on the inspect button from the top right corner, enter the inputs, click on the predict button, and see the result/prediction on the web.

Now, Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below results

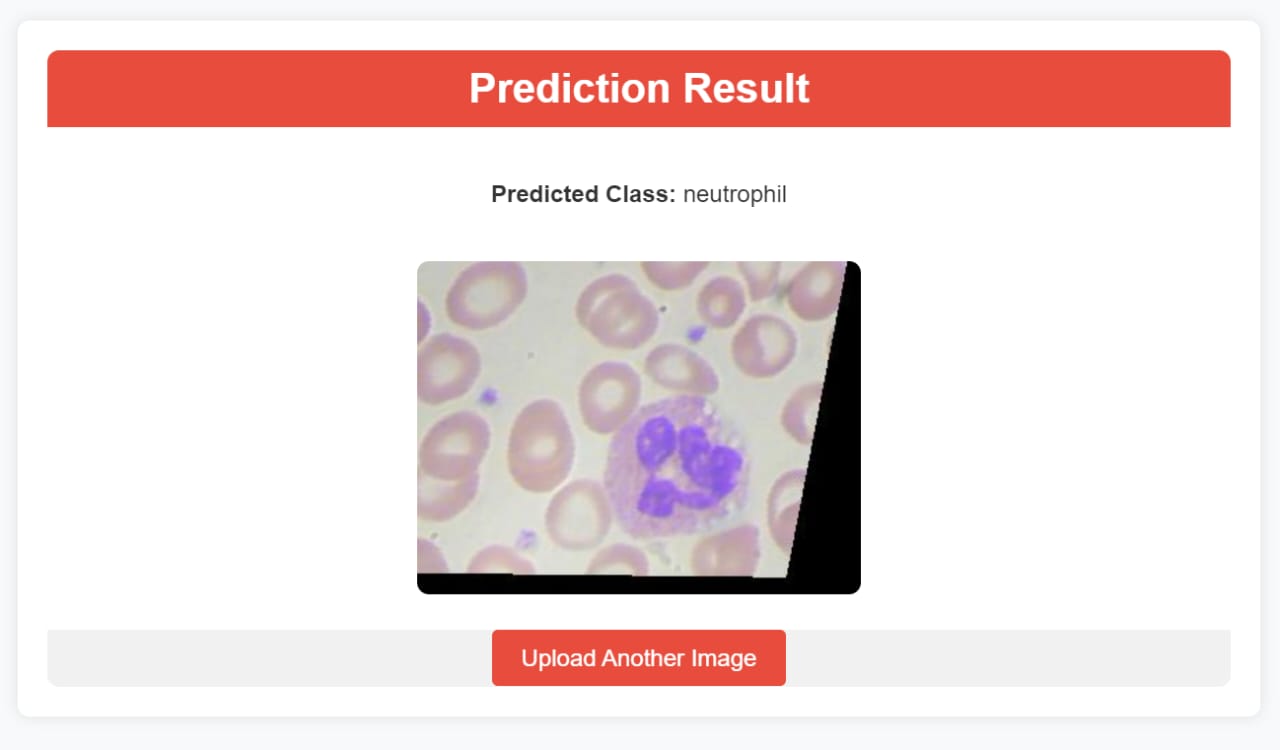
UI Image preview:

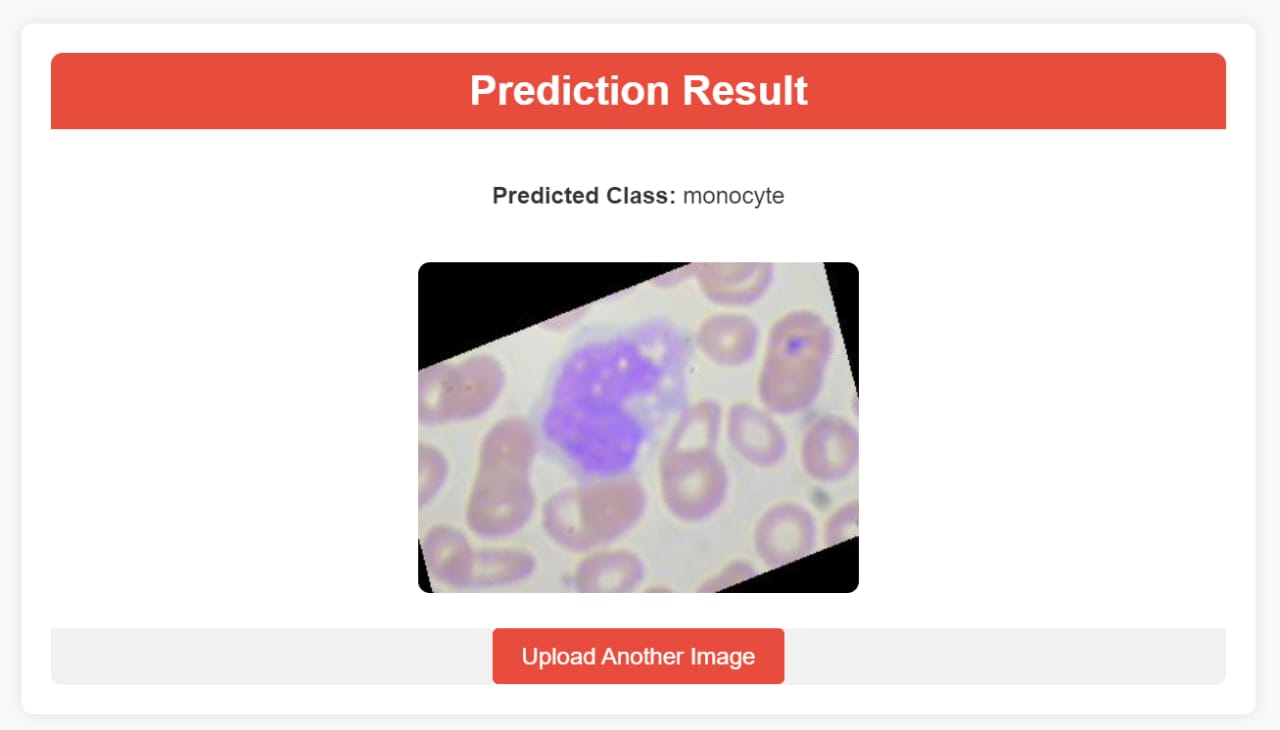


Let’s see what our index.html page looks like:

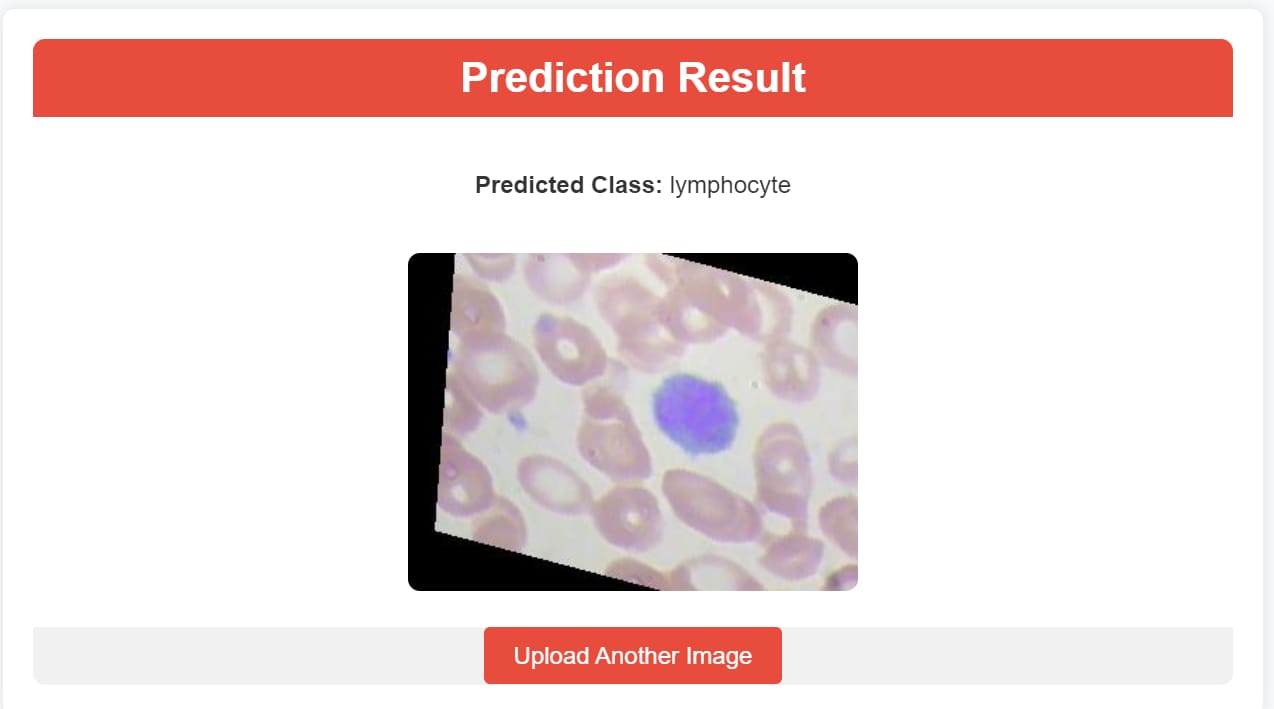
By clicking on choose file it will ask us to upload the image , then by clicking on the predict button , it will take us to the result.html

Test For Class-1 : Neutrophil

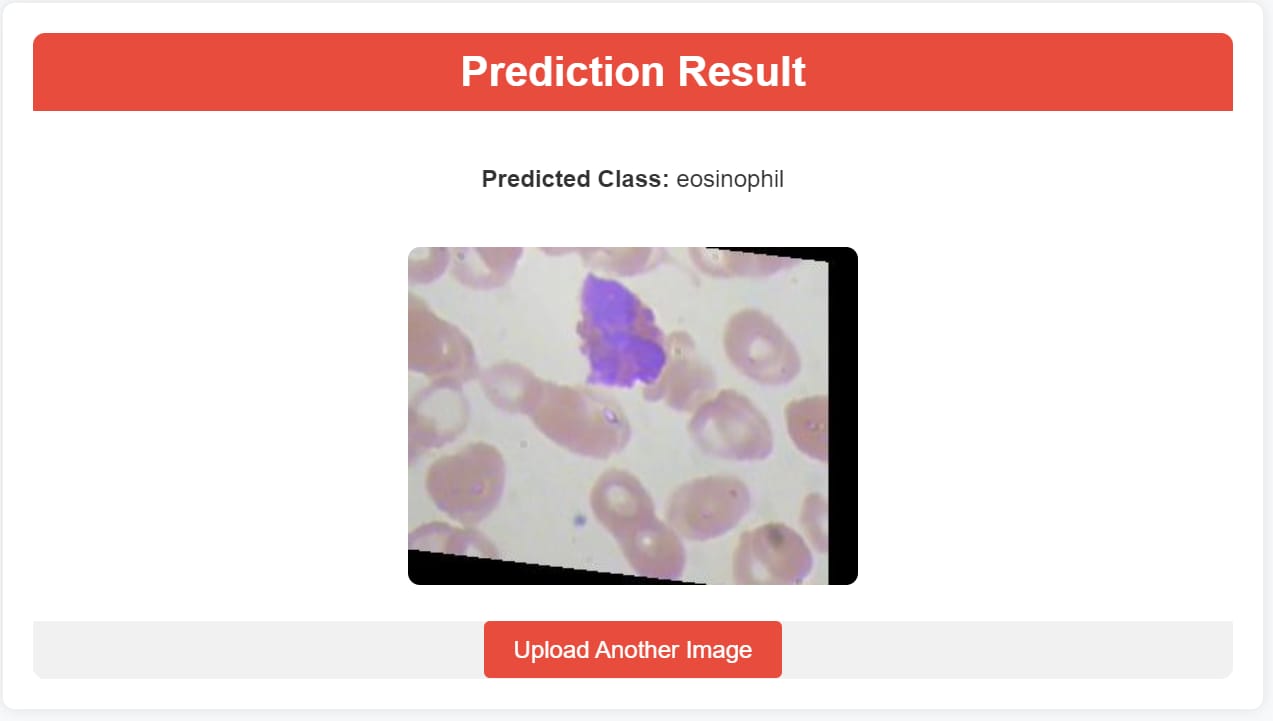


Test For Class-2 : Monocyt 

Test For Class-3 : Lymphocyte



Test For Class-4 : Eosinophil



* **Testing:**

Matplotlib / Seaborn – For plotting accuracy, loss, confusion matrices

TensorBoard – For visualizing model metrics and activation maps

Grad-CAM – To visualize which parts of blood cell images influence predictions

* **TensorFlow/Keras Testing Tools**

model.evaluate() – Evaluate the model on test data.

model.predict() – Predict classes on new data.

ImageDataGenerator – Load and test images from directories.

tf.keras.metrics – Built-in metrics like Accuracy, Precision, Recall, etc.

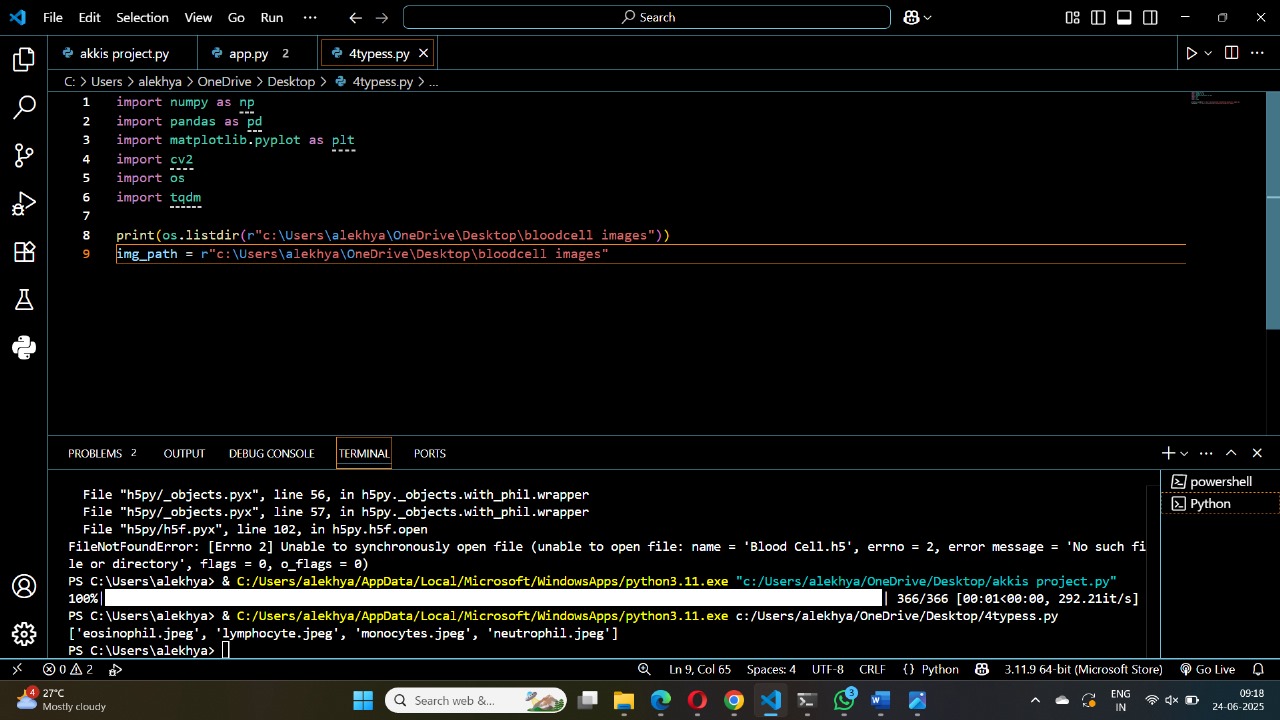
* **PyTorch Testing Tools**

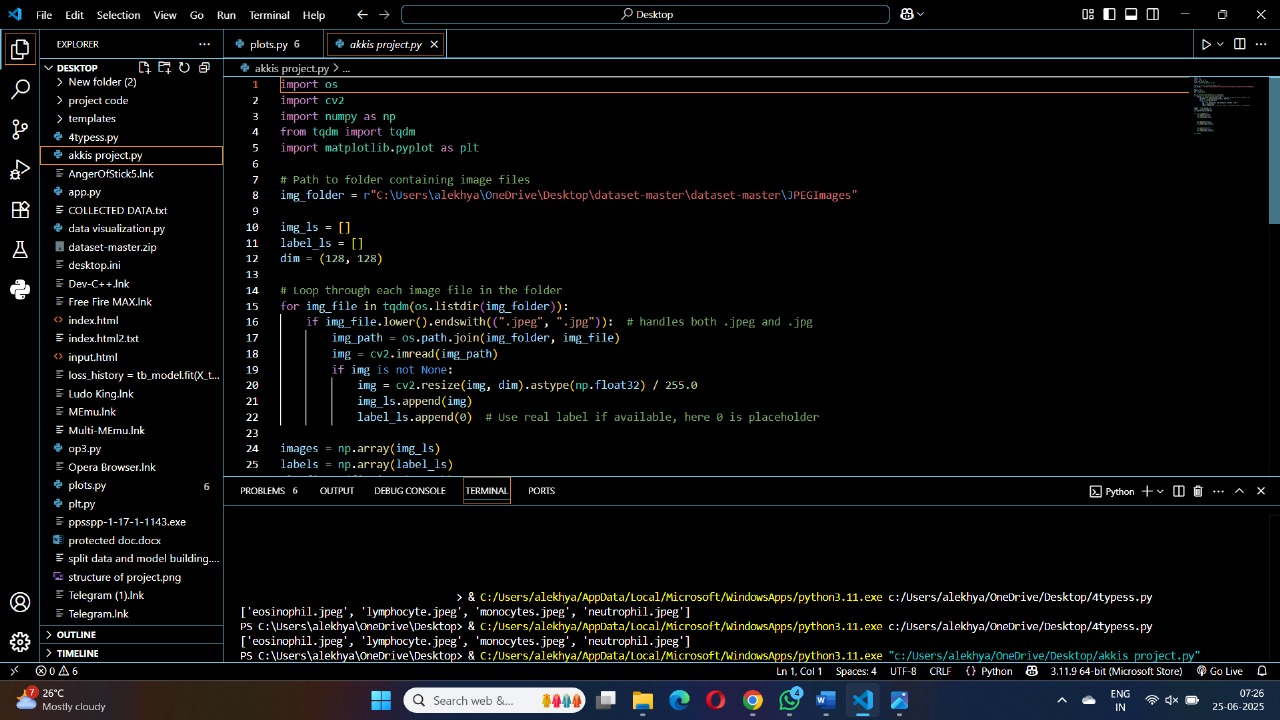
DataLoader – Load and batch test data.

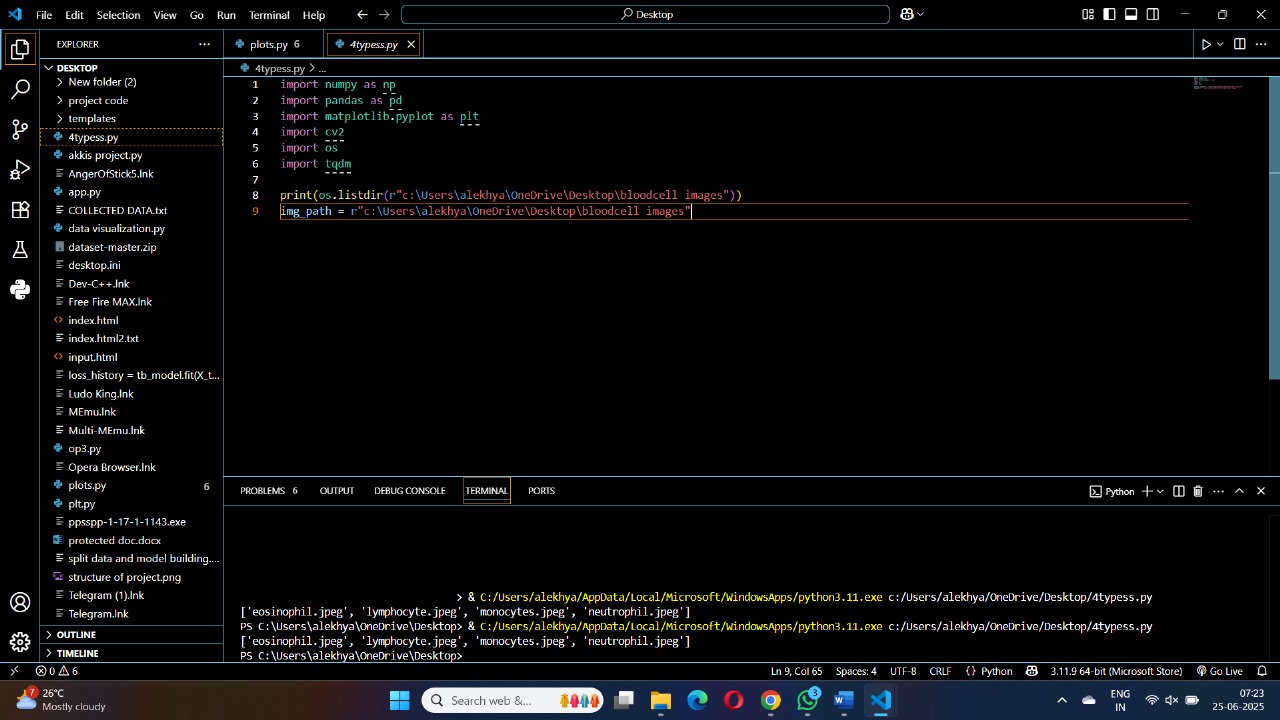
torchmetrics – Compute test accuracy, F1 score, etc.

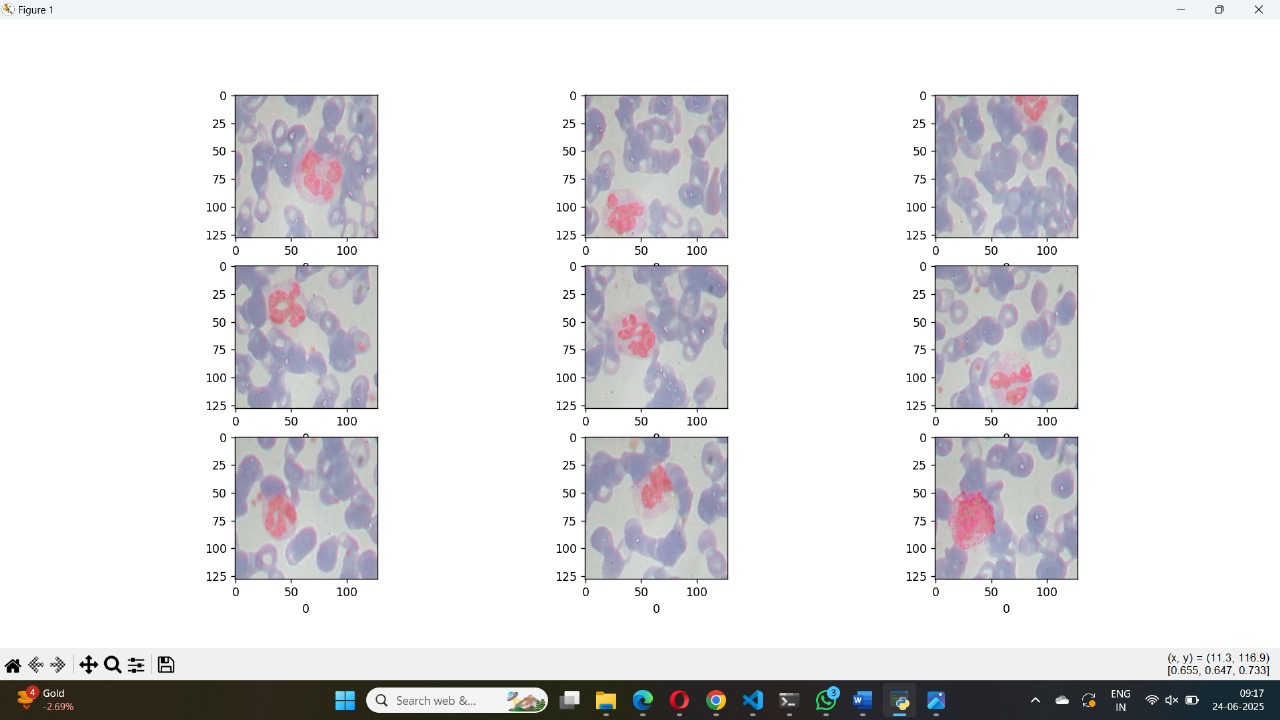
torch.no\_grad() – For efficient inference during testing.

* **Screenshots output :**









* **Future Enhancements** :

**Integration of Real-Time Microscopy Data**

* Develop a system that can classify blood cells in real-time directly from microscopic imaging devices, enhancing clinical applicability.

**Multi-Class and Multi-Disease Detection**

* Expand the model to classify additional types of blood cells and detect related diseases such as malaria, leukemia, anemia, etc., in a single framework.

**Explainable AI (XAI) Integration**

* Incorporate explainable AI techniques like Grad-CAM to visualize which parts of the blood cell images influence the model’s decisions, improving model transparency and trust among healthcare professionals.

**Cloud-Based Deployment**

* Deploy the model on cloud platforms to enable easy access for remote laboratories and hospitals, especially in under-resourced areas.

**Mobile Application Development**

* Develop a mobile application for point-of-care testing, allowing rapid blood cell classification using smartphone-attached microscopes.

**Improved Data Augmentation**

* + Apply advanced data augmentation techniques such as GAN-based synthetic image generation to further improve the model’s robustness and generalization.

**Continuous Learning**

* + Implement continuous learning pipelines that can update the model automatically as new blood cell image data becomes available, keeping the system up-to-date.

**Cross-Dataset Generalization**

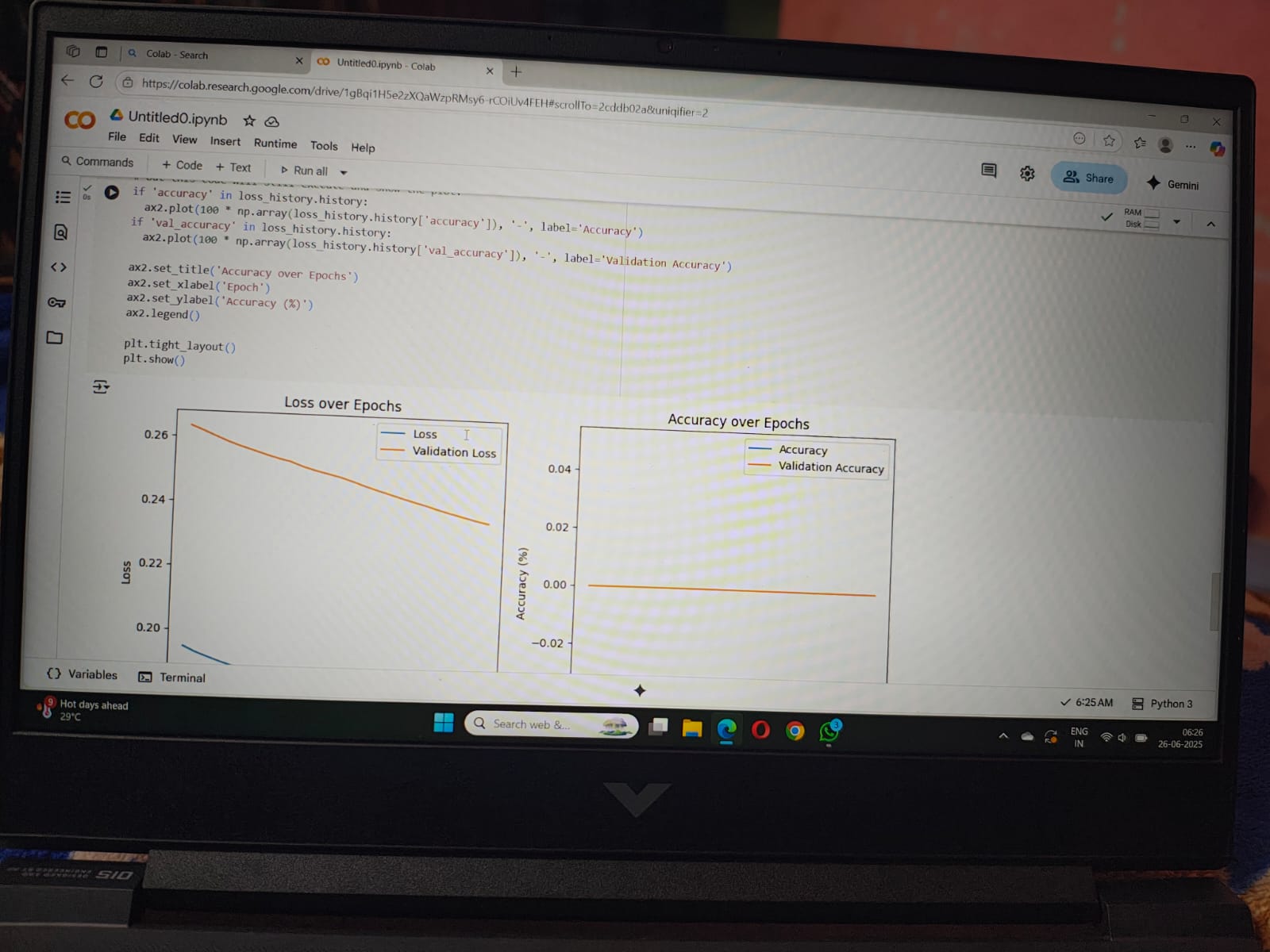
* + Test and enhance the model’s performance across multiple datasets from different laboratories to ensure it generalizes well to various imaging conditions.

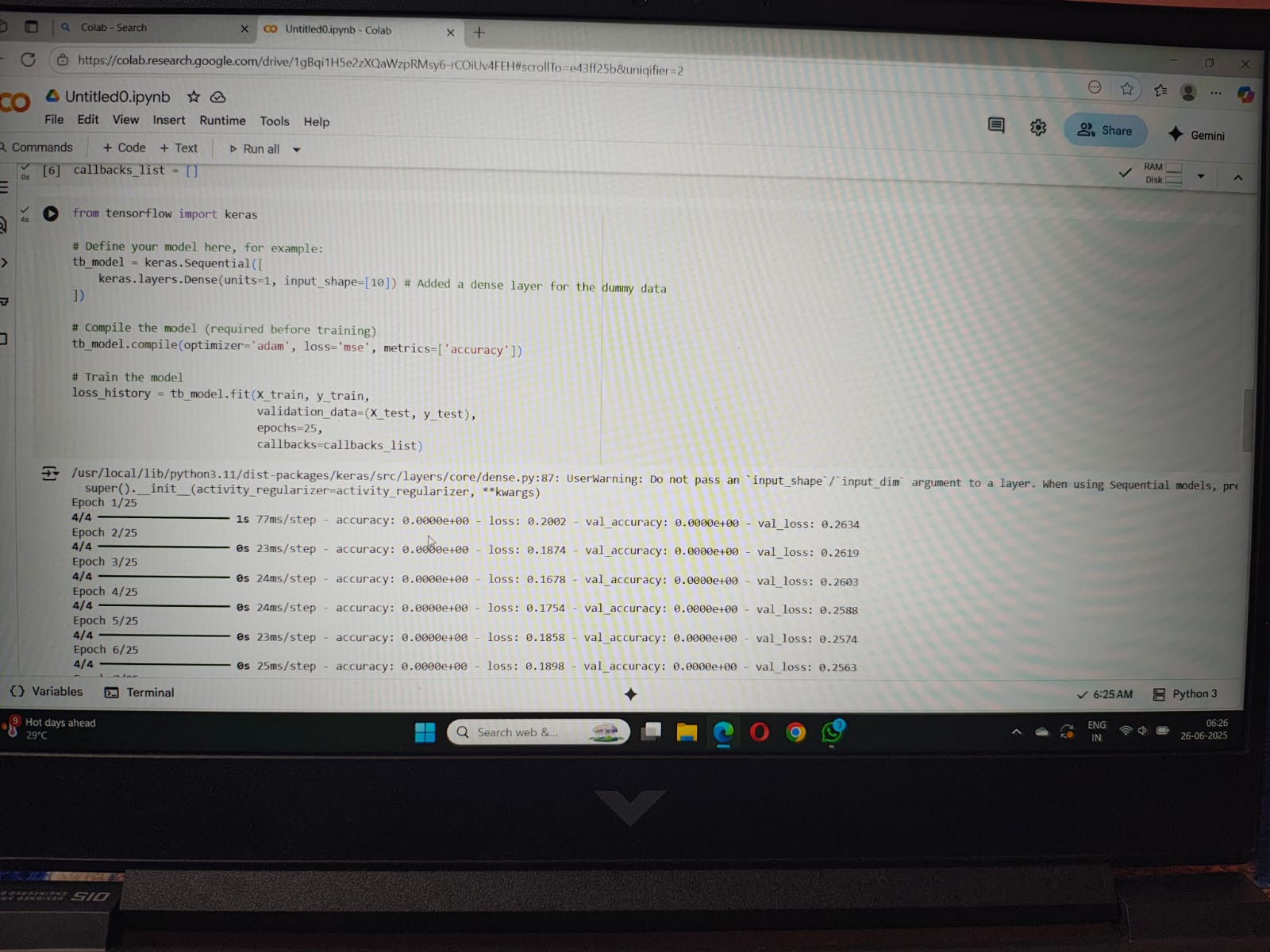
**Automated Labeling Assistance**

* + Integrate semi-supervised or unsupervised learning methods to reduce the dependency on manual labeling of large datasets.

**3D Image Processing**

* + Extend the model to handle 3D microscopic imaging for more detailed analysis of blood cell morphology.
* I have done my coding in Google colab **:**





* Conclusion:

The **Advanced Blood Cell Classification using Transfer Learning** project successfully demonstrates the effectiveness of using deep learning techniques to accurately classify different types of blood cells. By utilizing pre-trained models through transfer learning, the project significantly reduces the training time and improves classification performance, even with a limited dataset.

Throughout the process, the model was trained, validated, and tested using Google Colab, which provided a powerful and accessible platform with GPU support. Data preprocessing, augmentation, and visualization played a crucial role in enhancing model accuracy and preventing overfitting.

The training and validation curves indicate that the model steadily improved, though further tuning and expansion of the dataset could yield even better results. The final model can assist healthcare professionals by offering a fast, automated, and reliable method for blood cell classification, potentially supporting early diagnosis and treatment of blood-related diseases.

In this project, Google Colab is used as the primary development environment for implementing Advanced Blood Cell Classification using Transfer Learning. Google Colab provides a free, GPU-enabled platform that supports Python and TensorFlow, making it ideal for deep learning model development and testing.

This project can be further extended by integrating additional blood cell types, deploying the model on mobile or cloud platforms, and using real-time microscopy data to create a fully automated diagnostic system.