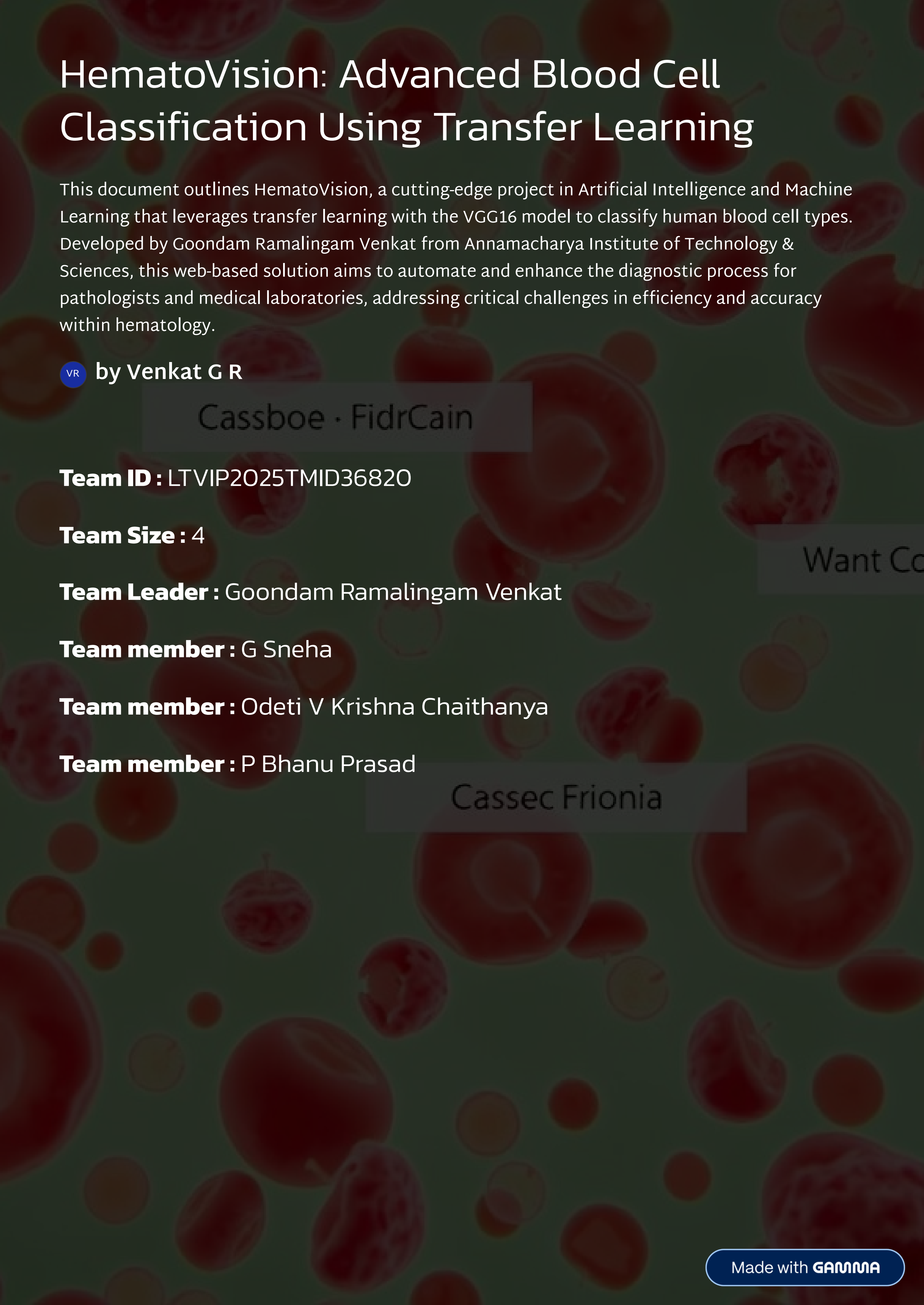


HematoVision: Advanced Blood Cell Classification Using Transfer Learning

This document outlines HematoVision, a cutting-edge project in Artificial Intelligence and Machine Learning that leverages transfer learning with the VGG16 model to classify human blood cell types. Developed by Goondam Ramalingam Venkat from Annamacharya Institute of Technology & Sciences, this web-based solution aims to automate and enhance the diagnostic process for pathologists and medical laboratories, addressing critical challenges in efficiency and accuracy within hematology.

VR **by Venkat G R**

Team ID : LTVIP2025TMID36820

Team Size : 4

Team Leader : Goondam Ramalingam Venkat

Team member : G Sneha

Team member : Odeti V Krishna Chaithanya

Team member : P Bhanu Prasad

Introduction and Problem Statement

HematoVision aims to develop an accurate and efficient model for classifying blood cells by employing transfer learning techniques. Utilizing a dataset of 12,000 annotated blood cell images, categorized into distinct classes such as eosinophils, lymphocytes, monocytes, and neutrophils, the project leverages pre-trained convolutional neural networks (CNNs) to expedite training and improve classification accuracy. Transfer learning allows the model to benefit from pre-existing knowledge of image features, significantly enhancing its performance and reducing computational costs. This approach provides a reliable and scalable tool for pathologists and healthcare professionals, ensuring precise and efficient blood cell classification.

Scenario 1: Automated Diagnostic Systems for Healthcare

Integrating HematoVision into automated diagnostic systems in clinical settings can revolutionize blood analysis. By using transfer learning, the system quickly adapts to the specifics of blood cell classification, capturing images of blood samples, classifying the cells in real-time, and generating detailed reports. This automation reduces the manual workload on pathologists, speeds up diagnostic processes, and ensures high accuracy in results, ultimately improving patient care and treatment efficiency.

Scenario 2: Remote Medical Consultations

HematoVision can be employed in telemedicine platforms to enhance remote consultations and diagnostics. With transfer learning, the model's ability to accurately classify blood cells from diverse sources is improved, allowing healthcare providers to upload blood cell images for automated analysis. This enables timely and accurate assessments without the need for in-person visits, facilitating better access to specialized medical expertise and improving healthcare delivery in remote or underserved areas.

Scenario 3: Educational Tools for Medical Training

HematoVision's transfer learning-based classification model can be integrated into educational tools for medical training. By incorporating this advanced technology into interactive learning platforms, students and laboratory technicians can upload and analyze blood cell images to receive instant feedback. This hands-on learning experience enhances their understanding of blood cell morphology and classification, providing practical skills and knowledge that are crucial for accurate diagnostic practice and medical training.

Delay in Classification

Manual review of blood slides leads to significant diagnostic delays, impacting patient treatment timelines.

Human Error

Microscopic analysis is prone to subjective interpretation and fatigue, resulting in potential misclassifications.

Need for Automation

Existing labs and remote diagnostics require scalable, automated tools to enhance efficiency and accessibility.

Project Objectives and System Architecture

HematoVision's primary objectives are to automate blood cell classification, provide real-time predictions via a user-friendly web interface, and ultimately assist healthcare professionals in early and accurate diagnosis. The system is designed to streamline the diagnostic workflow, ensuring faster and more reliable results.

The system architecture is compartmentalized into distinct layers, each serving a crucial function to ensure seamless operation and efficient processing of blood cell images.

UI Layer

The user-facing web interface facilitates easy image uploads and interaction.

Flask Backend

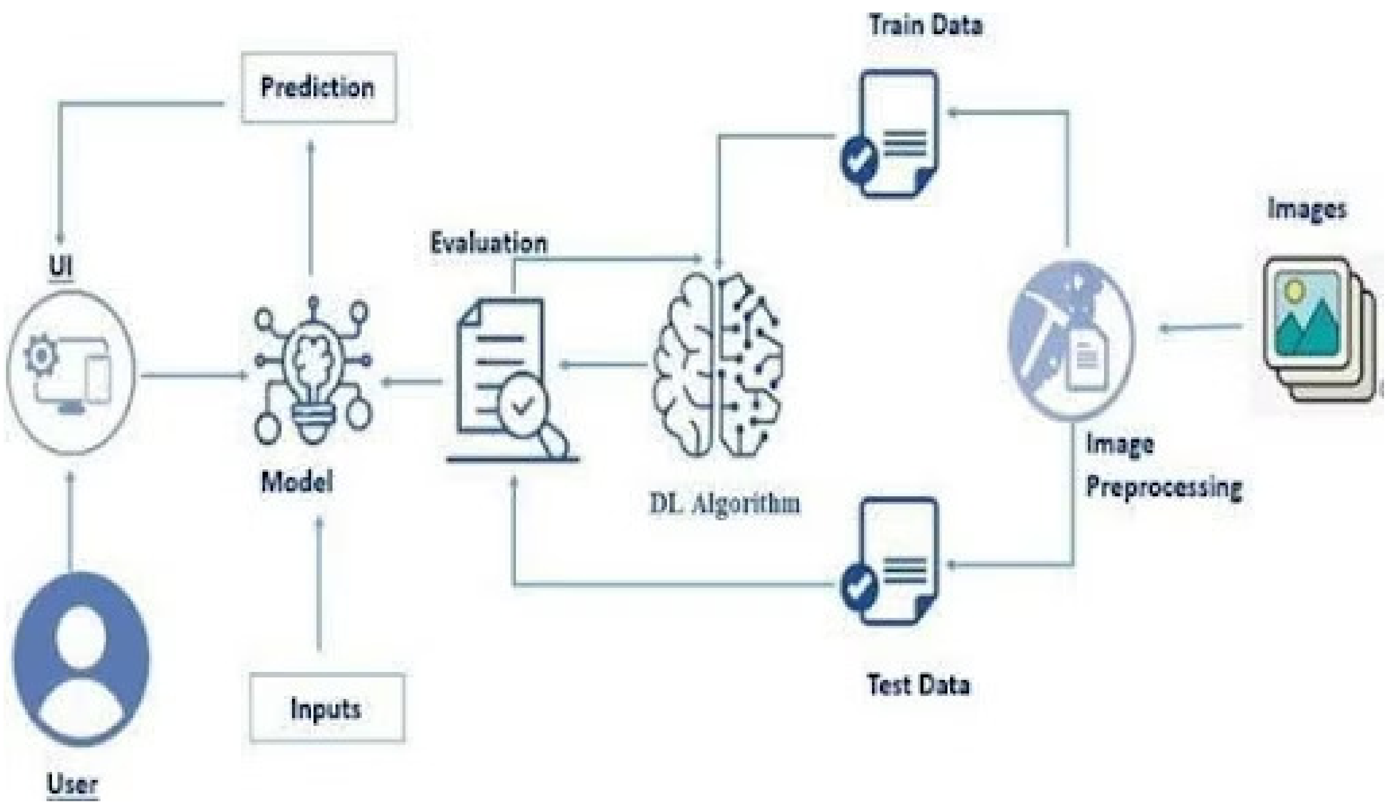
Handles all incoming user requests, manages model loading, and processes prediction tasks.

Model Layer

Utilizes the pre-trained VGG16 Convolutional Neural Network (CNN) with applied transfer learning for classification.

Output Layer

Displays the predicted class label of the blood cell and its corresponding confidence score.



Prerequisites and Project Flow

Prior Knowledge

You must have prior knowledge of the following topics to complete this project.

- DL Concepts
  - Neural Networks:: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/
- Deep Learning Frameworks:: https://www.knowledgehut.com/blog/data-science/pytorch-vs-tensorflow
- Transfer Learning: https://towardsdatascience.com/a-demonstration-of-transfer-learning-of-vgg-convolutional-neural-network-pre-trained-model-with-c9f5b8b1ab0a
- VGG16: https://www.geeksforgeeks.org/vgg-16-cnn-model/
- Convolutional Neural Networks (CNNs): https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/ s://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
- Overfitting and Regularization: https://www.analyticsvidhya.com/blog/2021/07/prevent-overfitting-using-regularization-techniques/
- Optimizers: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/
- Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Objectives

By the end of this project, you will:

- Know fundamental concepts and techniques used for Deep Learning.
- Gain a broad understanding of data.
- Have knowledge of pre-processing the data/transformation techniques on outliers and some visualization concepts.

Project Work flow

The project follows a systematic flow, ensuring each phase builds upon the previous one, from data handling to final deployment.

1 Dataset Preprocessing

Cleaning and preparing image data.

2 Model Training

Utilizing VGG16 with transfer learning.

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.

Download 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.

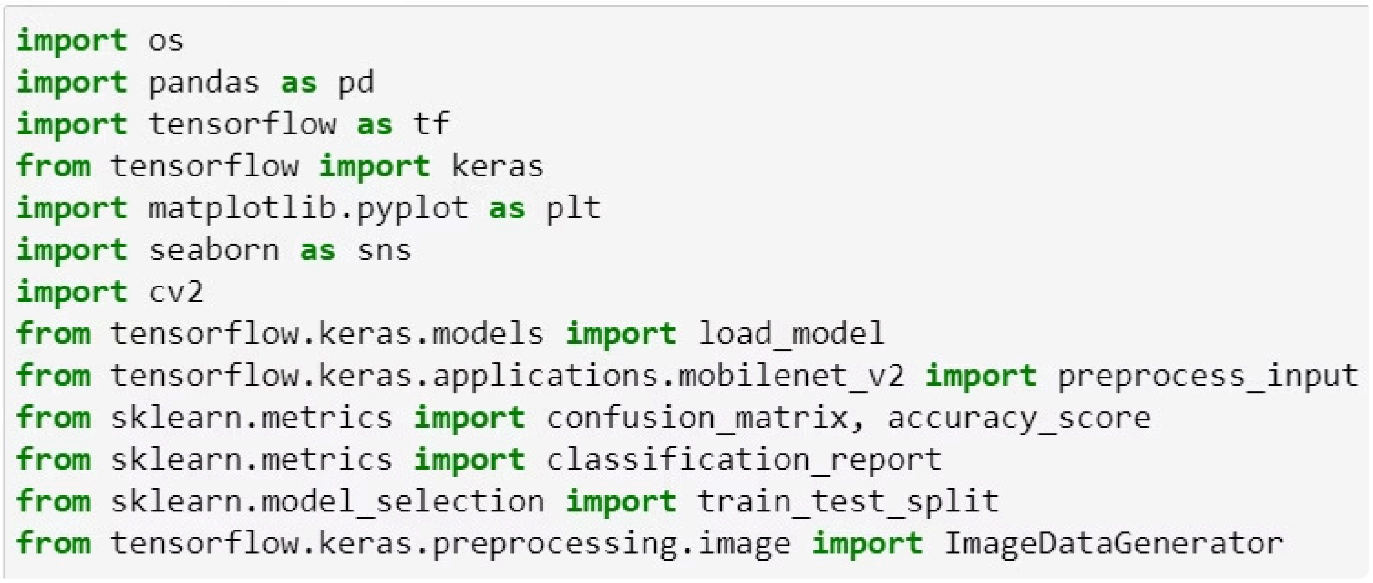
Link: https://www.kaggle.com/datasets/paultimothymooney/blood-cells/data

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.

Activity 1.1: Importing the libraries:

- Import the necessary libraries as shown in the image.

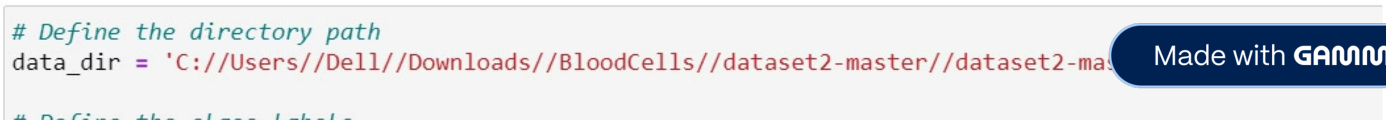


```python
import os
import pandas as pd
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
from tensorflow.keras.models import load_model
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Activity 1.2: Read the 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.

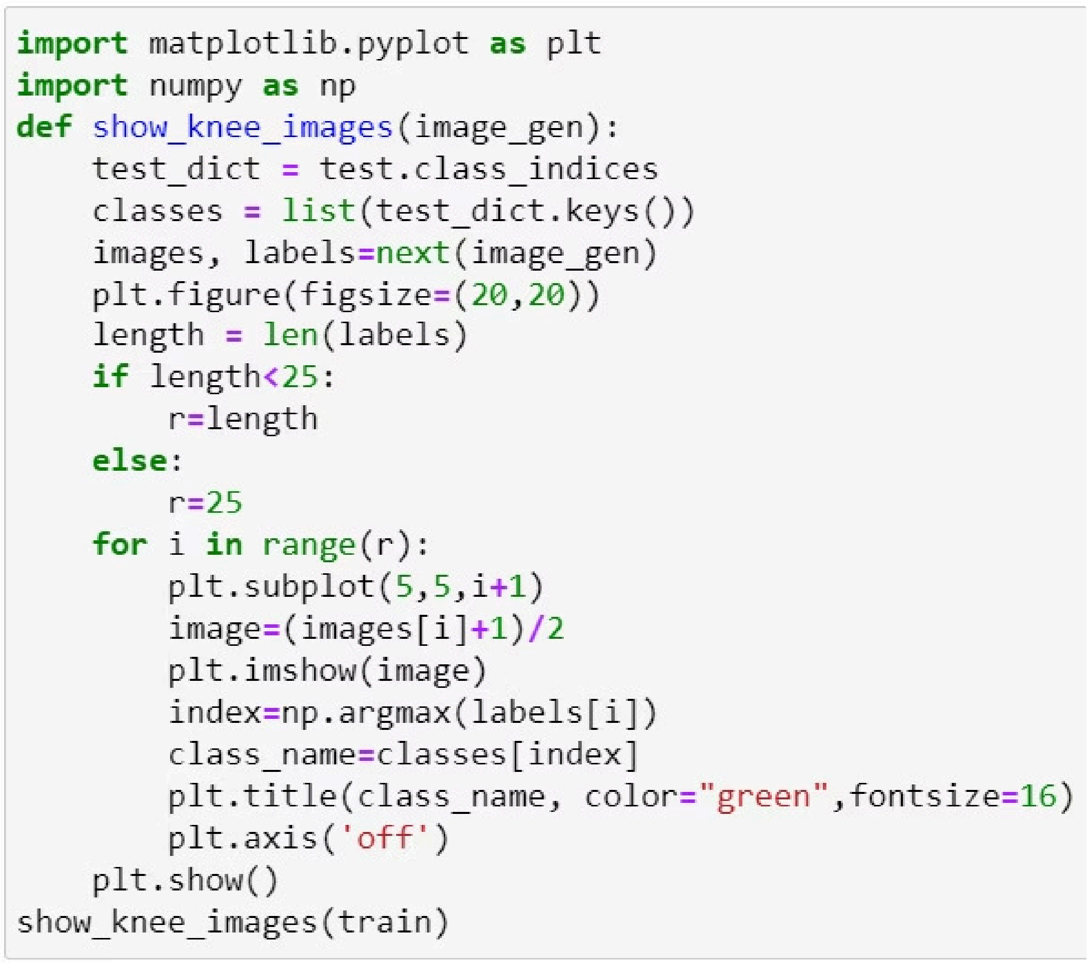
At first, unzip the data and convert it into a pandas data frame.



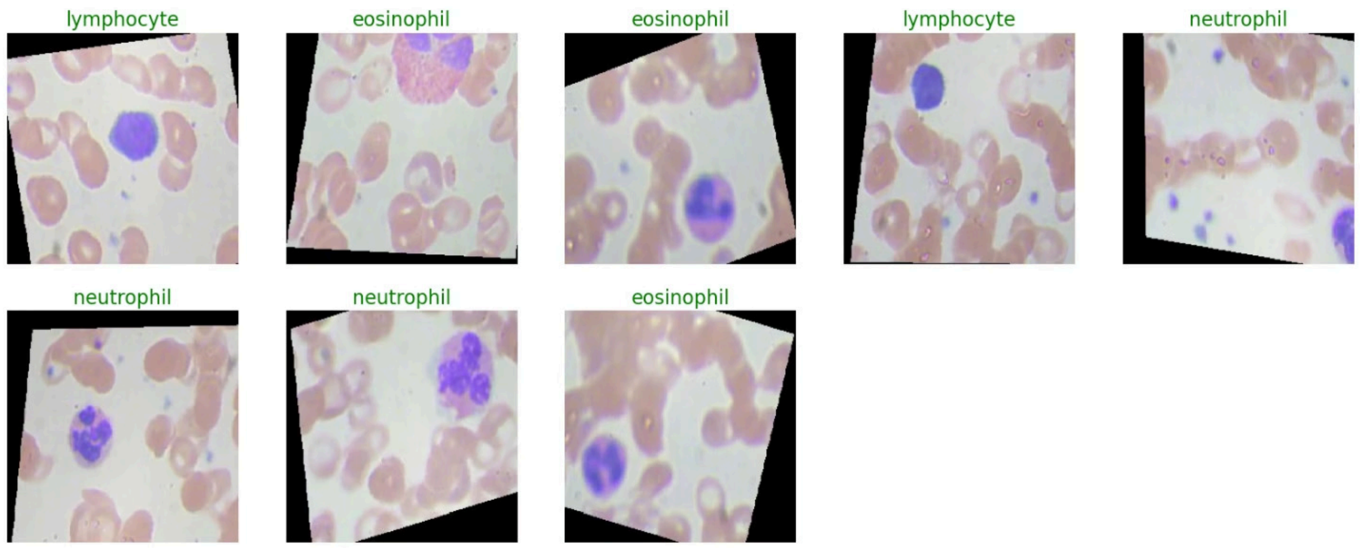
```python
# Define the directory path
data_dir = 'C://Users//Dell//Downloads//BloodCells//dataset2-master//dataset2-ma
```

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.



```python
import matplotlib.pyplot as plt
import numpy as np
def show_knee_images(image_gen):
    test_dict = test.class_indices
    classes = list(test_dict.keys())
    images, labels=next(image_gen)
    plt.figure(figsize=(20,20))
    length = len(labels)
    if length<25:
        r=length
    else:
        r=25
    for i in range(r):
        plt.subplot(5,5,i+1)
        image=(images[i]+1)/2
        plt.imshow(image)
        index=np.argmax(labels[i])
        class_name=classes[index]
        plt.title(class_name, color="green",fontsize=16)
        plt.axis('off')
    plt.show()
show_knee_images(train)
```

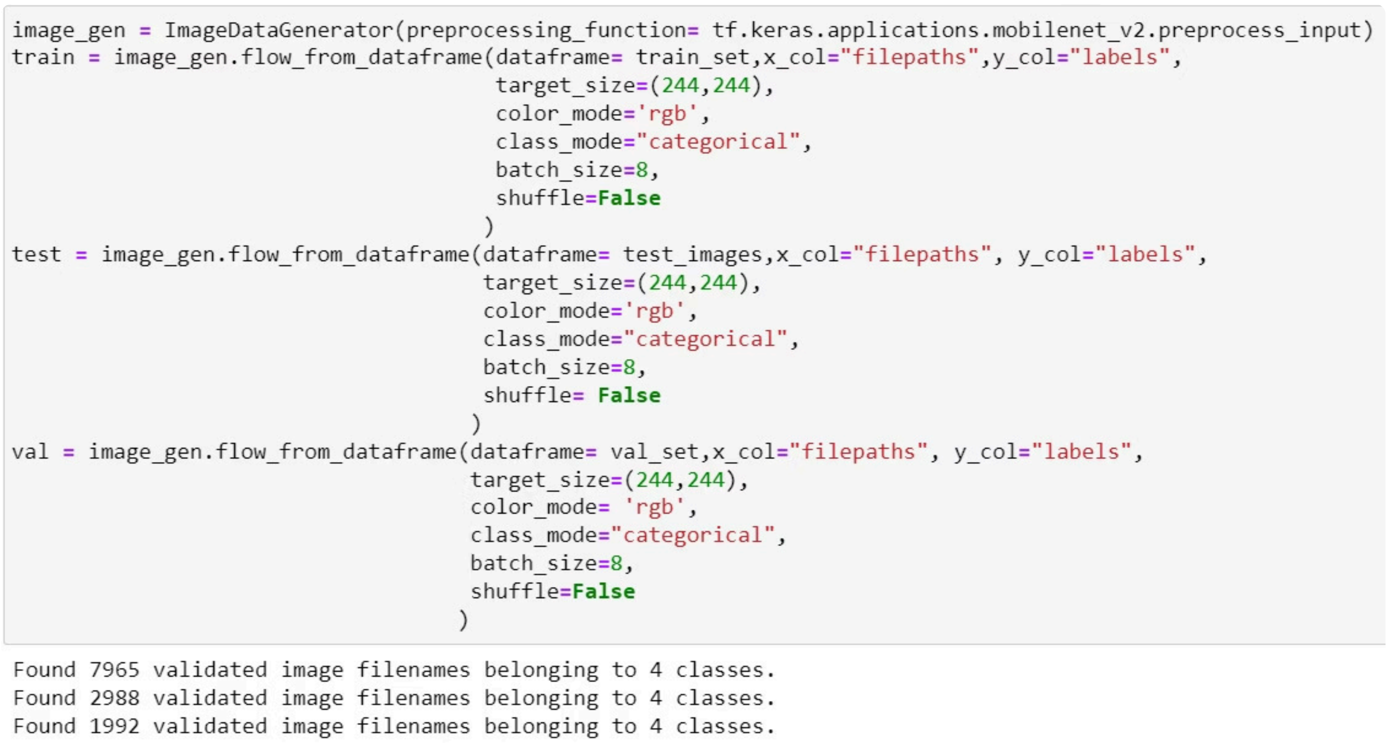


In the above code, I used class ace of diamond for prediction, This code randomly selects an image file from a specified folder (folder_path) containing JPEG, PNG, or JPEG files, and then displays the selected image using IPython's display function. It utilizes Python's OS and rando

manipulation and random selection, respectively. And It has predicted correctly a

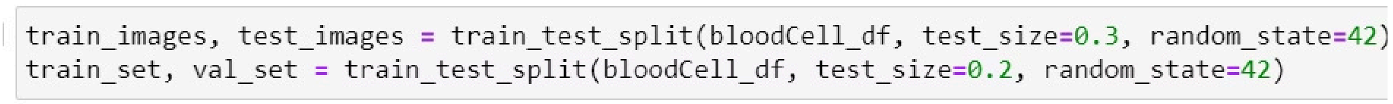
Model Building: Transfer Learning with VGG16

Train-Test-Split:In this project, we have already separated data for training and testing.

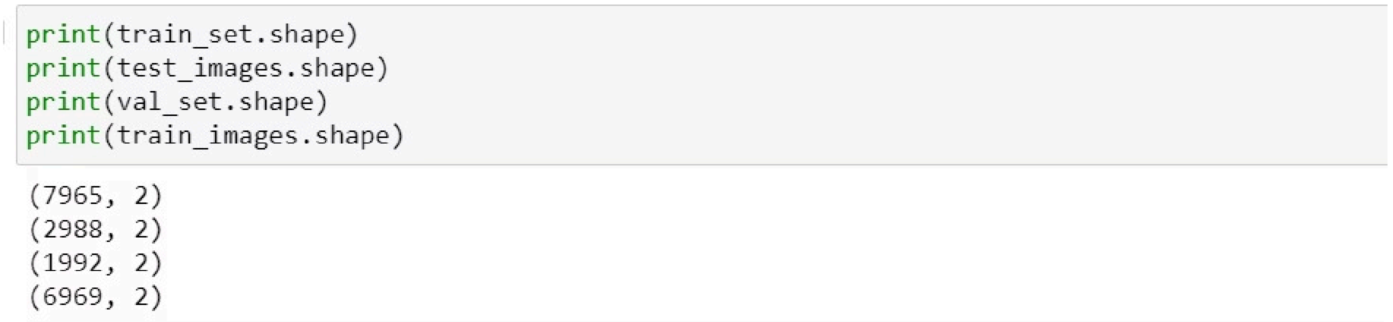


```
image_gen = ImageDataGenerator(preprocessing_function= tf.keras.applications.mobilenet_v2.preprocess_input)
train = image_gen.flow_from_dataframe(dataframe= train_set,x_col="filepaths",y_col="labels",
                                      target_size=(244,244),
                                      color_mode='rgb',
                                      class_mode="categorical",
                                      batch_size=8,
                                      shuffle=False
                                     )
test = image_gen.flow_from_dataframe(dataframe= test_images,x_col="filepaths", y_col="labels",
                                     target_size=(244,244),
                                     color_mode='rgb',
                                     class_mode="categorical",
                                     batch_size=8,
                                     shuffle= False
                                    )
val = image_gen.flow_from_dataframe(dataframe= val_set,x_col="filepaths", y_col="labels",
                                    target_size=(244,244),
                                    color_mode= 'rgb',
                                    class_mode="categorical",
                                    batch_size=8,
                                    shuffle=False
                                   )
```

```
Found 7965 validated image filenames belonging to 4 classes.
Found 2988 validated image filenames belonging to 4 classes.
Found 1992 validated image filenames belonging to 4 classes.
```



```
train_images, test_images = train_test_split(bloodCell_df, test_size=0.3, random_state=42)
train_set, val_set = train_test_split(bloodCell_df, test_size=0.2, random_state=42)
```



```
print(train_set.shape)
print(test_images.shape)
print(val_set.shape)
print(train_images.shape)
```

```
(7965, 2)
(2988, 2)
(1992, 2)
(6969, 2)
```

1 Pre-trained VGG16

The VGG16 model, pre-trained on the extensive ImageNet dataset, serves as the foundation for feature extraction.

2 Top Layer Removal

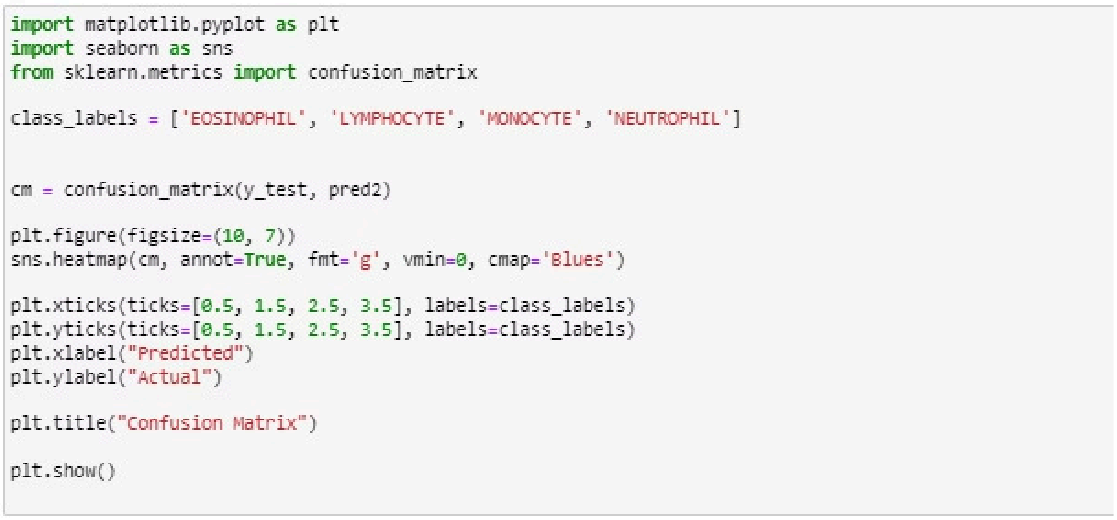
The original classification layers of VGG16 are removed (include_top=False) to adapt it for blood cell classification.

3 New Classifier Added

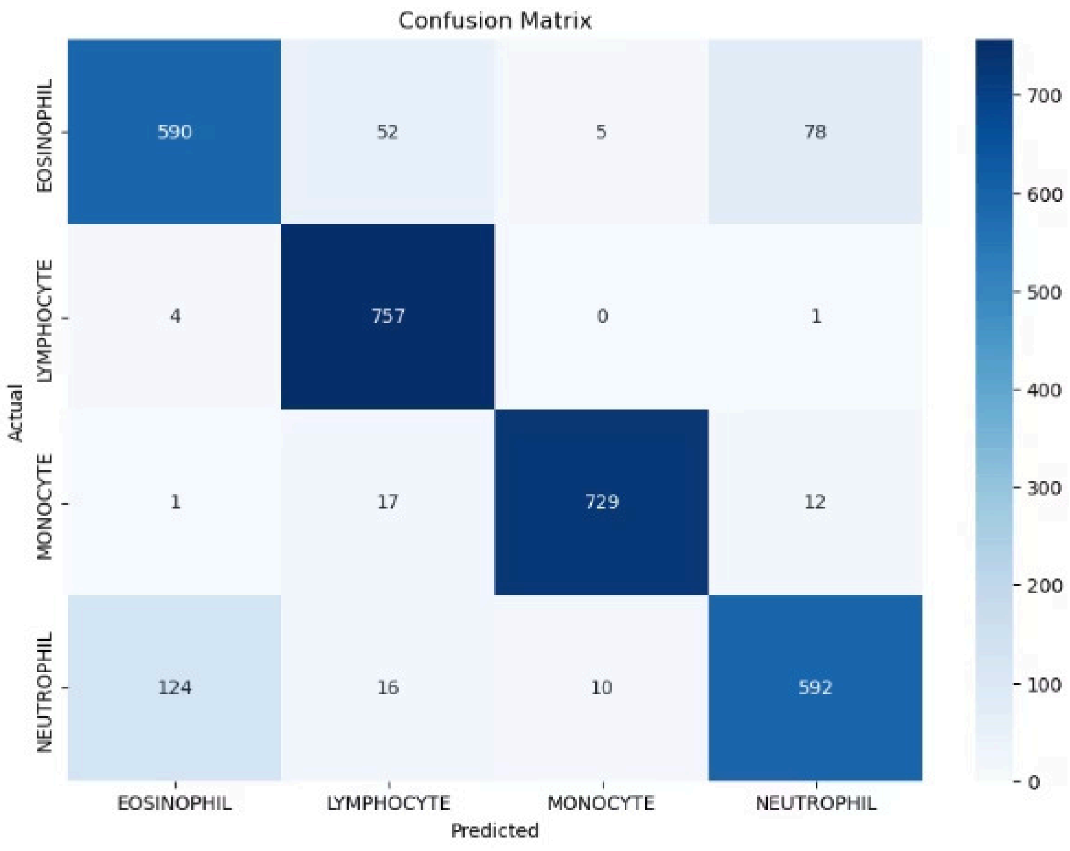
A new classifier head is appended, consisting of a Flatten layer, a Dropout regularization), and a Dense output layer with Softmax activation.

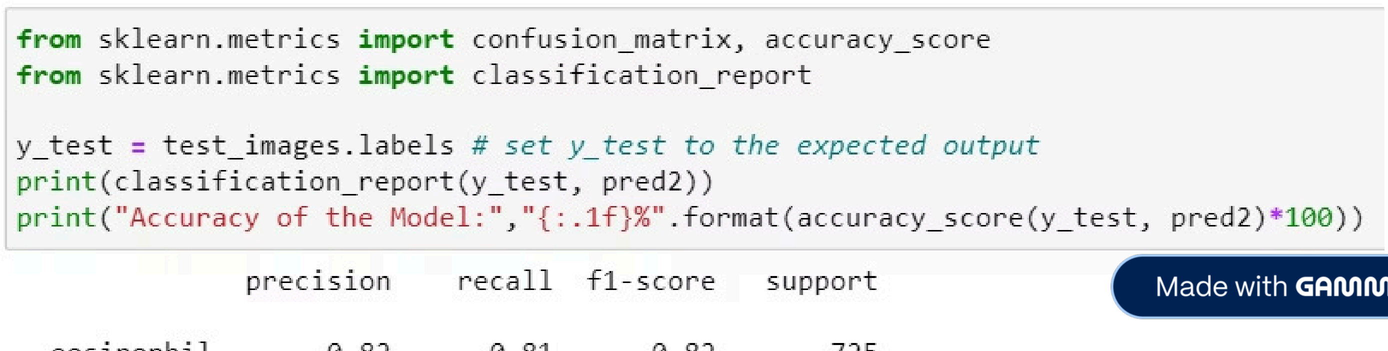
Testing Model & Data Prediction

Evaluating the modelHere we have tested with the Mobilenet V2 Model With the help of the predict () function.



```
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()
```





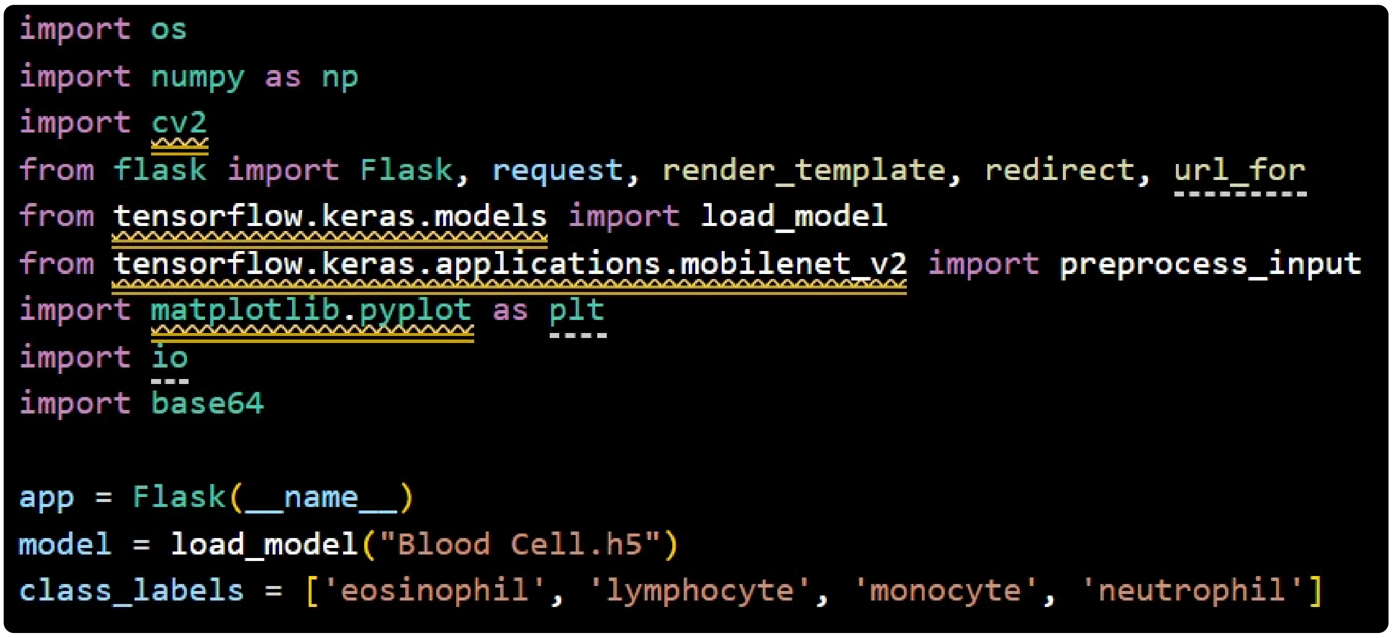
```
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import classification_report

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
```

Application Development and Deployment

Build Python code:

Import the libraries



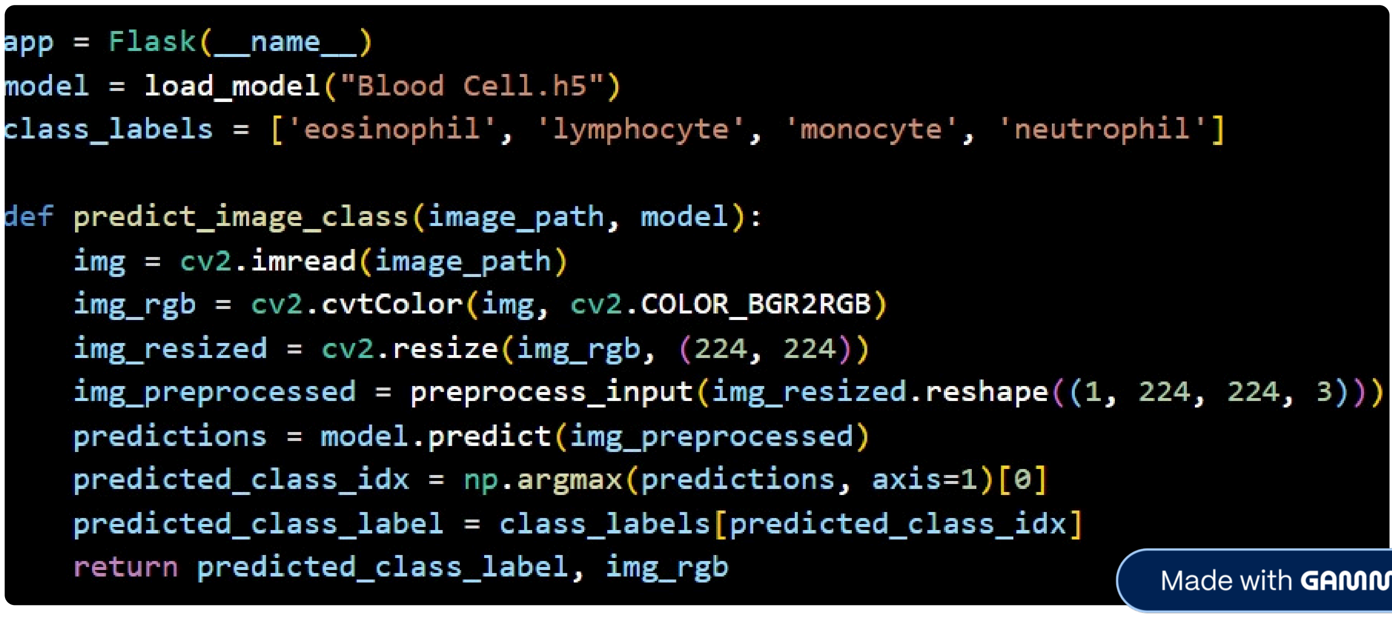
```python
import os
import numpy as np
import cv2
from flask import Flask, request, render_template, redirect, url_for
from tensorflow.keras.models import load_model
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
import matplotlib.pyplot as plt
import io
import base64

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

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:

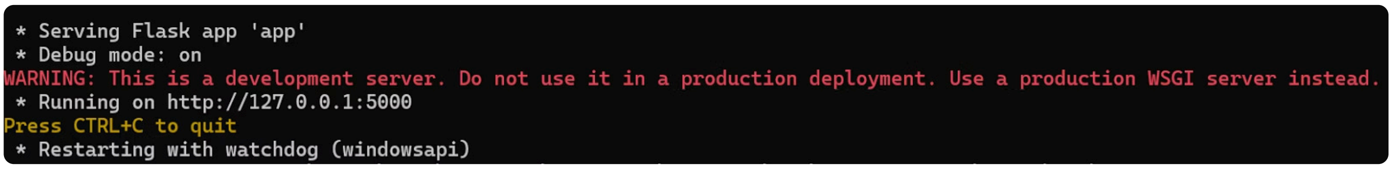


```python
app = Flask(__name__)
model = load_model("Blood Cell.h5")
class_labels = ['eosinophil', 'lymphocyte', 'monocyte', 'neutrophil']

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
```

Run the web application

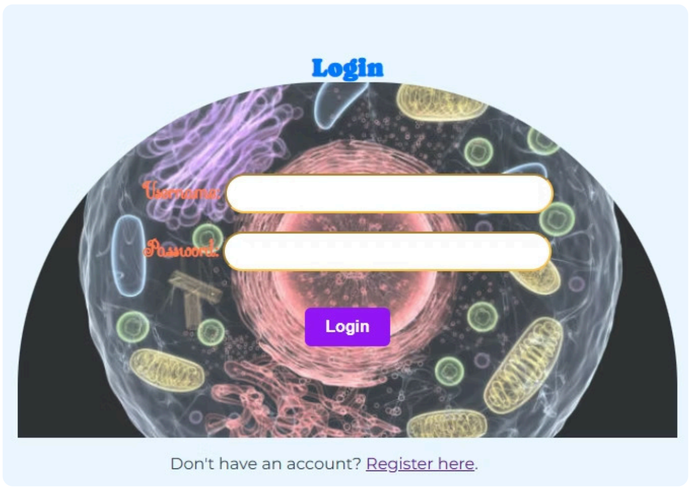
- : Run the application

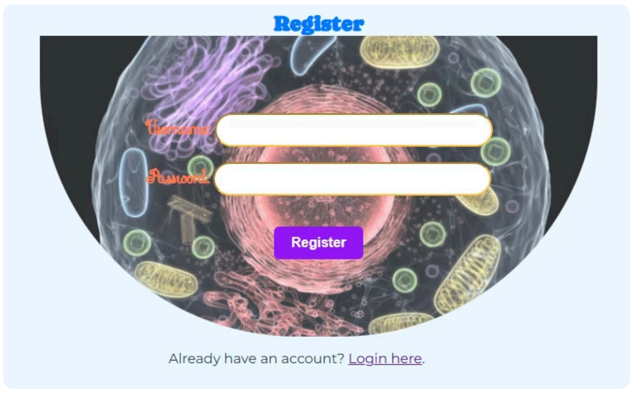
Now, Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below results

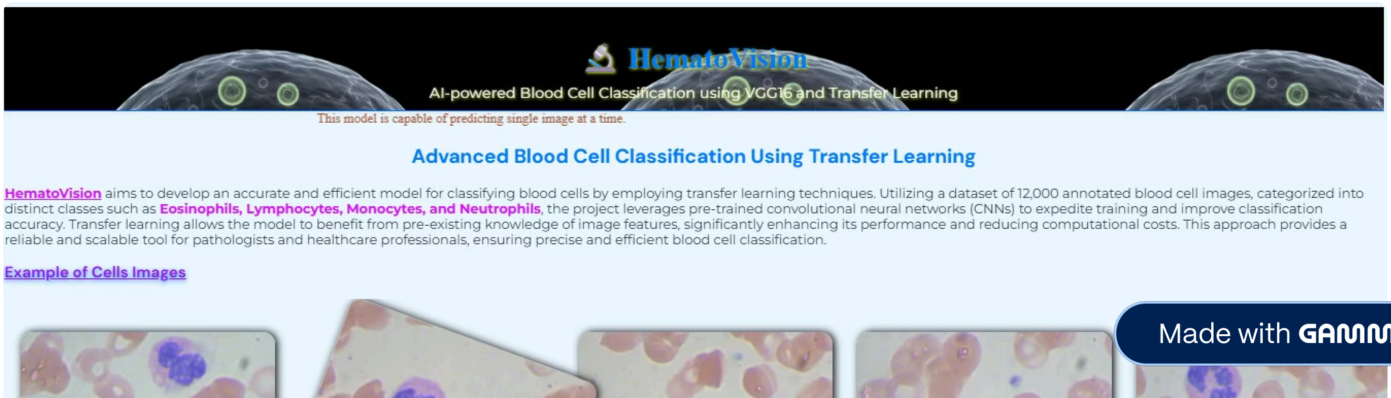


```
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
```

  - 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.
  - 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.htmlTest For Class-1 : NeutrophilTest For Class-2 : MonocytTest For Class-3 : Lymphocyte
    Test For Class-4 : Eosinophil







HematoVision

AI-powered Blood Cell Classification using VGG16 and Transfer Learning

This model is capable of predicting single image at a time.

Advanced Blood Cell Classification Using Transfer Learning

HematoVision aims to develop an accurate and efficient model for classifying blood cells by employing transfer learning techniques. Utilizing a dataset of 12,000 annotated blood cell images, categorized into distinct classes such as **Eosinophils, Lymphocytes, Monocytes, and Neutrophils**, the project leverages pre-trained convolutional neural networks (CNNs) to expedite training and improve classification accuracy. Transfer learning allows the model to benefit from pre-existing knowledge of image features, significantly enhancing its performance and reducing computational costs. This approach provides a reliable and scalable tool for pathologists and healthcare professionals, ensuring precise and efficient blood cell classification.

Example of Cells Images

Conclusion and Future Scope

HematoVision stands as a testament to the effective application of transfer learning and deep learning in addressing real-world medical diagnostic challenges. The project successfully leverages the pre-trained VGG16 model to achieve high accuracy in classifying human blood cells, providing a robust and automated solution. By integrating this model into a user-friendly Flask web interface, HematoVision effectively reduces the reliance on manual microscopic analysis, thereby mitigating human error and significantly accelerating the diagnostic process. This not only enhances efficiency in clinical settings but also supports faster decision-making for healthcare professionals, ultimately contributing to improved patient outcomes.

The system's deployable and lightweight nature makes it a valuable tool for a diverse range of users, from medical students and laboratory technicians to professionals in remote health centers where specialized expertise may be limited. Its ability to provide accurate and rapid classifications demonstrates the immense potential of computer vision in modern healthcare.

Future Scope: Enhancing HematoVision

While HematoVision delivers a significant advancement in blood cell classification, the project has immense potential for future enhancements and expansions. These future developments aim to broaden the application's utility, improve its diagnostic capabilities, and increase its accessibility across various platforms.

Multi-Cell Image Support

Implement object detection capabilities to classify multiple blood cells within a single image, enhancing efficiency and diagnostic completeness.

Expanded Dataset for Generalization

Integrate a larger and more diverse dataset to further improve the model's generalization capabilities, ensuring higher accuracy across a wider range of image variations.

Mobile App Integration

Develop a dedicated mobile application to allow on-the-go blood cell classification, making the tool accessible to a broader audience and in various clinical settings.