Leukaemia Cancer Detection using Image Classification

Project report submitted for
Industrial Project Based Learning in Data Science and Machine Learning

Under the Supervision of

P.Mohan

Director, Gyan Astra IT Solutions

&

Y.V.N Phani Kishore

Director, Gyan Astra IT Solutions

By

Team Members Name	**(Roll Number)**
P Pranavi	21R11A6742
C Manognasri	21R11A6729
A. Sai Srujana	21R11A05A8
P Moulika	21R11A05E0

Department of Computer Science & Engineering

Nov 2023 - May 2024

Geethanjali College of Engineering & Technology

Accredited by NBA (UGC Autonomous)

(Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi, NAAC - A+)

Geethanjali College of Engineering & Technology

(Affiliated to J.N.T.U.H, Approved by AICTE, NEWDELHI.)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that Mr./Ms. C.Manognasri, A.Sai Srujana, P Moulika , P Pranavi bearing roll no 21R11A6729,21R11A05A8,21R11A05E0, 21R11A6742 has successfully completed Industrial Project Based Learning in Data Science and Machine Learning held during November 2023 to May 2024 at Geethanjali college of Engineering and Technology.

Coordinator	**H.O.D - C.S.E**
Mr. E. Mahender	**Dr. A. Sree Lakshmi**
Assistant Professor	**Professor & Head**
Dept of C.S.E	**Dept of C.S.E**

ABSTRACT

Leukemia, a devastating form of cancer impacting individuals of all ages, necessitates swift and accurate diagnosis to improve therapeutic outcomes and elevate survival rates. However, the current diagnostic process, reliant on manual analysis of blood samples through microscopic images, is plagued by sluggishness, labor intensiveness, and suboptimal accuracy. To address these shortcomings, the field is witnessing the emergence of automated machine learning algorithms designed to detect leukemia with greater efficiency and precision. In this project, we propose the implementation of an intelligent deep learning algorithm leveraging Convolutional Neural Network (CNN), ResNet, and VGG architectures to discern leukemic cells from healthy blood cells.

Our dataset encompasses an extensive collection of images depicting leukemic blood cell patterns, serving as the foundation for training and assessing image classification models. Our primary objective is to cultivate a robust classifier model capable of aiding in the early detection of leukemia, thereby contributing to improved patient care and prognosis. Through the fusion of advanced machine learning techniques and comprehensive datasets, we endeavor to propel the field of leukaemia diagnosis towards enhanced effectiveness and reliability.

.

TABLE OF CONTENTS

S.NO	TOPICS	PAGE NUMBERS
i	Abstract	i
1	Introduction	3
2	Literature survey	4
3	Problem statement	4
4	Objectives	4
5	Methodology	5
5.1	Data Collection	5
5.2	Importing and required packages	5
5.3	Data Pre-Processing	6
5.3.1	Auto-orientation of pixel data	6
5.3.2	Resize to 640x640	7
5.4	Data Augmentation	9
5.4.1	50% probability of horizontal flip	9
5.4.2	50% probability of vertical flip	10
5.4.3	Random rotation of between -30 and +30 degrees	11
5.4.4	Random brightness adjustment of between -15 and +15 %	12
5.4.5	Random Gaussian blur of between 0 and 2 pixels	13
6	Model Implementation	15
7	Web Application	18
7.1	Result	19
8	Conclusion	22
9	Future Scope	23
10	References	23

LIST OF FIGURES

Fig No	Text	Page No
5.3.1.1	Sample Image	6
5.3.1.2	Auto-oriented Image	7
5.3.2.1	Original Image	8
5.3.2.2	Resized Image	8
5.4.1.1	Original Image	9
5.4.1.2	Horizontal Flipped Image	9
5.4.2.1	Original Image	10
5.4.2.2	Vertically Flipped Image	10
5.4.3.1	Original Image	11
5.4.3.2	Rotated Image	11
5.4.4.1	Original Image	12
5.4.4.2	Brightness Adjusted Image (-15)	12
5.4.5.1	Original Image	13
5.4.5.2	Blurred Image	13
6.1.3.1	Training Parameters Details	16
6.1.3.2	Epoch	16
6.1.4.2	Metrics	17
7.0.1	Web screen for leukaemia cancer detection	18
7.1.1	Running Flask application on localhost	19
7.1.2	Web address of Flask application	19
7.1.3	Uploading image	19
7.1.4	Result 1	20
7.1.5	Result 2	20
7.1.6	Result 3	21
7.1.7	Result	21

1. INTRODUCTION

Leukemia, a type of cancer affecting blood cells and bone marrow, represents a formidable challenge to public health globally. The conventional method of diagnosing leukemia through manual microscopic analysis of blood samples is fraught with drawbacks, including sluggishness and susceptibility to errors. However, the advent of automated machine learning algorithms holds promise for revolutionizing leukemia detection by offering improved speed and accuracy.

In this project, we embark on a journey to harness the power of deep learning techniques, specifically Convolutional Neural Networks (CNN), ResNet, and VGG neural networks, to tackle the complexities of leukemia diagnosis. By leveraging these sophisticated algorithms, we seek to enhance the efficiency and accuracy of classifying leukemic cells from normal cells using microscopic images. This endeavor is driven by the overarching goal of improving patient outcomes through timely and precise diagnosis.

Our exploration of learning techniques represents a pivotal advancement in the field of leukemia diagnosis, offering the potential to overcome the limitations of traditional diagnostic methods. By capitalizing on the capabilities of CNN, ResNet, and VGG neural networks, we aim to unlock new avenues for identifying and categorizing leukemic cells with unprecedented accuracy.

Through this project, we not only aim to contribute to the scientific understanding of leukemia but also aspire to translate our findings into tangible benefits for patients and healthcare practitioners. By augmenting the capabilities of medical professionals with advanced machine learning algorithms, we envision a future where leukemia diagnosis is faster, more reliable, and ultimately, more conducive to improved patient outcomes.

2. LITERATURE SURVEY

Leukemia is a devastating disease with significant implications for patient health and well-being. Early detection is crucial for effective treatment and improved survival rates. Previous research has explored various approaches to leukemia detection, including manual microscopic analysis and automated machine learning algorithms. Learning techniques, such as CNNs, ResNet, and VGG neural networks, have shown promise in accurately classifying leukemic cells from healthy cells in microscopic images. By reviewing existing literature, we aim to build upon the foundation of knowledge in this field and contribute to the development of more effective diagnostic tools for leukemia.

3. PROBLEM STATEMENT

The current manual approach to leukemia diagnosis through microscopic analysis of blood samples is slow, time-consuming, and prone to inaccuracies. To address this challenge, there is a need for automated machine learning algorithms capable of accurately classifying leukemic cells from healthy cells in microscopic images. By developing robust image classification models using deep learning techniques, we aim to improve the efficiency and accuracy of leukemia diagnosis, ultimately leading to better patient outcomes and reduced mortality rates.

4. OBJECTIVES

The primary objective of this project is to develop intelligent image classification models using CNN, ResNet, and VGG neural networks to classify leukemic cells from healthy blood cells in microscopic images. Specific objectives include importing the image dataset into Python, pre-processing the images using data augmentation methods, training classifier models, evaluating metrics for all classifier models, identifying the best image classifier model, and building a web user interface using Flask API integration with the best image classifier model. Through these objectives, we aim to contribute to the advancement of leukemia detection methods and ultimately improve patient care and outcomes.

5. METHODOLOGY

5.1 Data Collection

- **Description of data**

Source: The dataset was fetched from the website 'https://universe.roboflow.com/custom-yolov5-o0hdb/leukemia-cancer-detection'.

Contents: The dataset contains a total of 7535 images and labels.

Data Split:

- Training Data: 6591 images and labels.
- Testing Data: 312 images and labels.
- Validation Data: 622 images and labels.

Format: Images are in standard image formats (e.g., JPG,), while labels are in a format compatible with YOLOv5 object detection framework.

Annotation: Training Data, Testing Data and Validation Data contains 4 types of images and labels such as Benign, Malignant Early, Malignant Pre, Malignant Pro. Labels include class labels representing the type of leukaemia and coordinates representing the location of cancer cells in the images.

5.2 Importing required packages

- **TensorFlow**: TensorFlow is an open-source machine learning framework developed by Google. It provides tools for building and training various machine learning models, including neural networks.

- **tensorflow.keras.models import Sequential:** Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. Sequential is a type of model in Keras that allows you to create neural networks layer by layer in a sequential manner.

- **tensorflow.keras.layers:** This module provides various types of layers that can be added to a neural network model.

- **NumPy**: NumPy is a popular Python library for numerical computations. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

- **CV2:** OpenCV (Open-Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. It provides tools for image processing, feature detection, object detection, and more.

- **OS:** The '**OS**' module provides a portable way of using operating system-dependent functionality. It allows you to interact with the operating system, such as manipulating file paths, directories, and environment variables.

5.3 Data Pre-processing

➢ **5.3.1 Auto-orientation of pixel data**

In training data, we have tilted, rotated images so we need to auto-rotate them for efficiency. First, I take a picture as shown below.

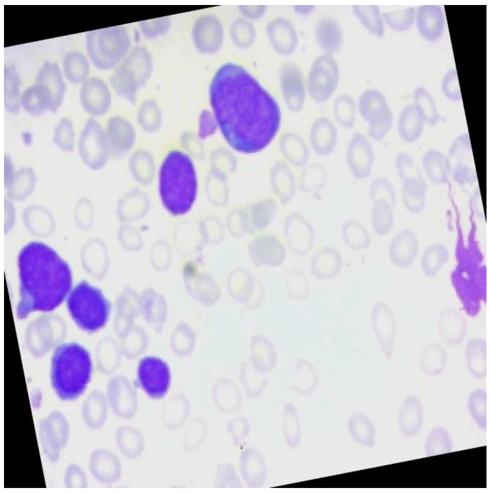


Fig 5.3.1.1 Sample Image

- Then we applied auto-orientation step and we got this:

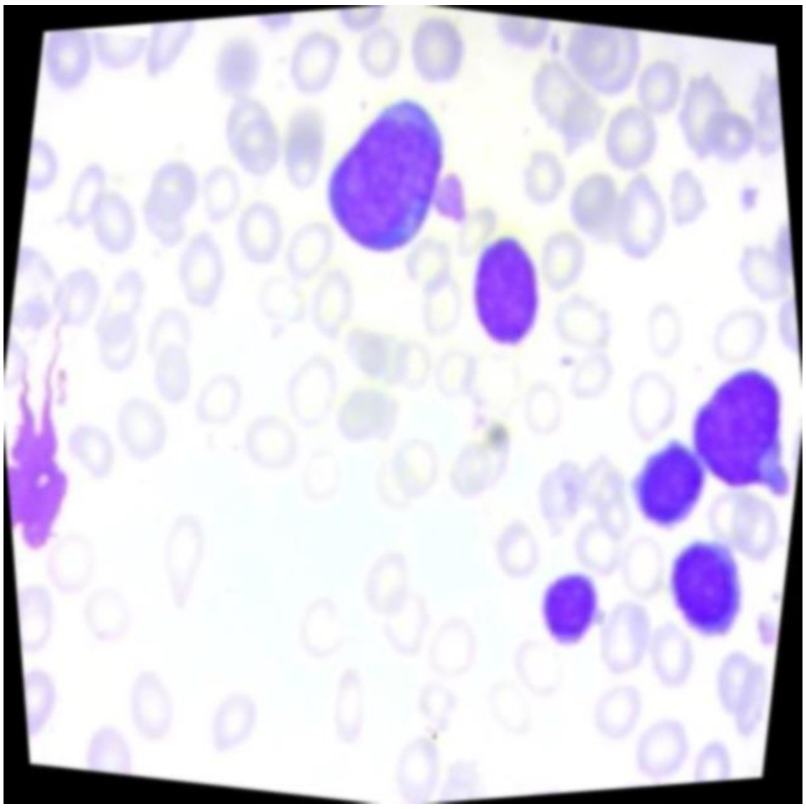


Fig 5.3.1.2 Auto-oriented Image

In this way we are rotating each and every image to improve the efficiency. And the auto-orientation method in order to improve the accuracy and efficiency of the model.

5.3.2 Resizing image (640 X 640)

In the training data, images come in varying sizes, necessitating conversion to a specified size for uniformity. This standardization is crucial as models process images more efficiently when they are of consistent dimensions rather than varied sizes.

We used image resizing method from open cv package to resize the images.

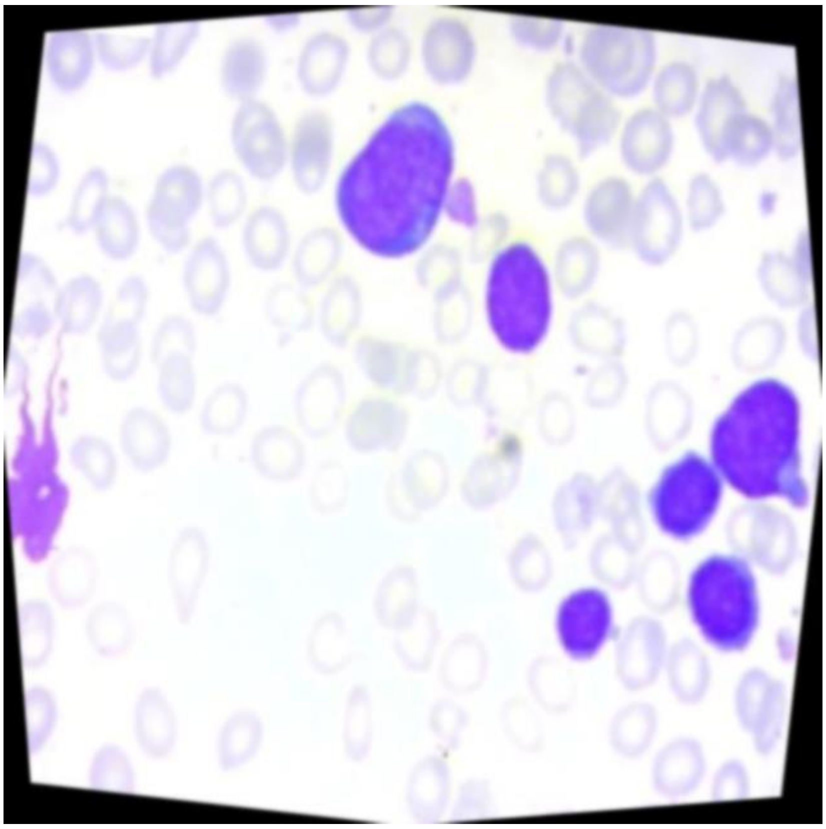


Fig 5.3.2.1 Original Image

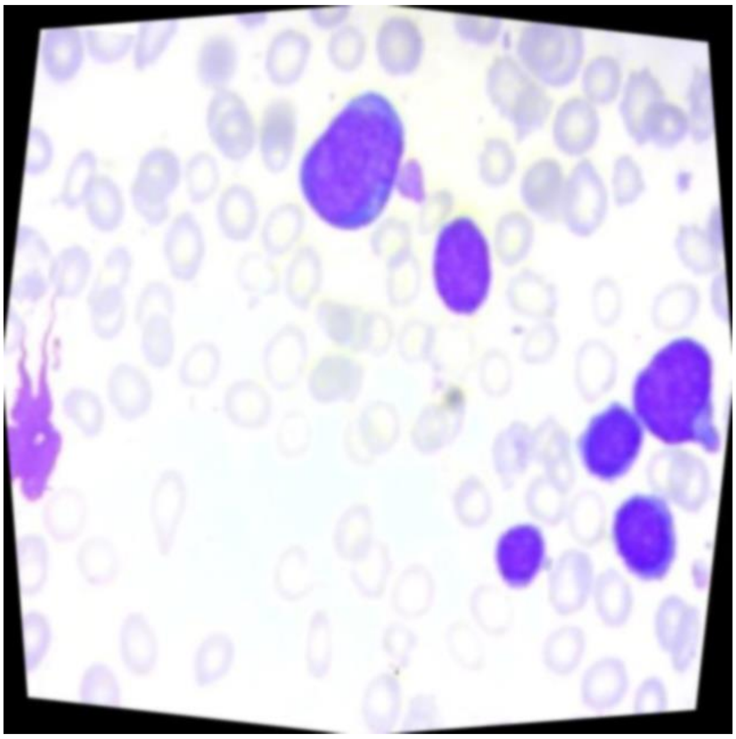


Fig 5.3.2.2 Resized Image

5.4 Data Augmentation

- **5.4.1 50% probability of horizontal flip**

"50% probability of horizontal flip," it means that for each image augmentation, there's an equal chance (50%) that the image will be flipped horizontally. This process introduces randomness into the augmentation pipeline, allowing for variation in the training data.

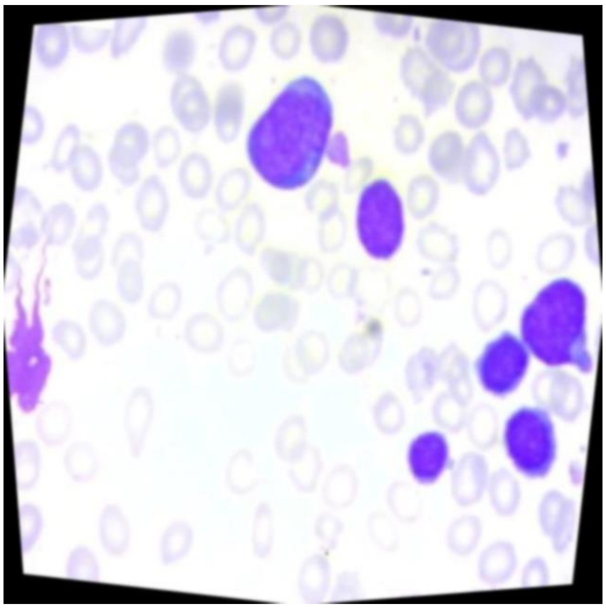


Fig 5.4.1.1 Original Image

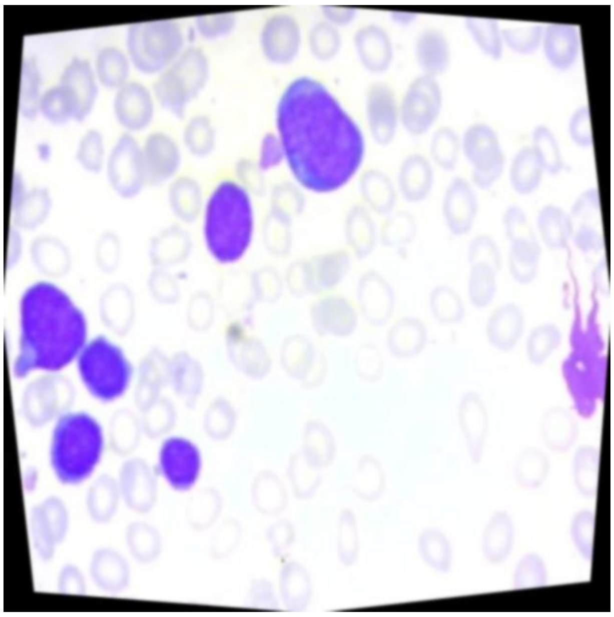


Fig 5.4.1.2 Horizontal Flipped Image

➢ 5.4.2 50% probability of vertical flip

"50% probability of vertical flip" means that for each image in the dataset undergoing augmentation, there's a 50% chance that the image will be flipped vertically.

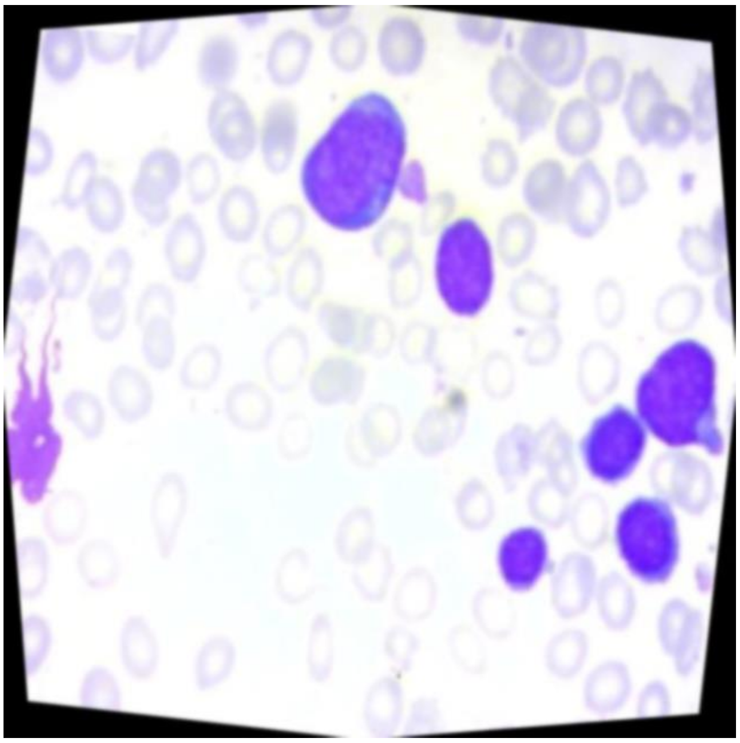


Fig 5.4.2.1 Original Image

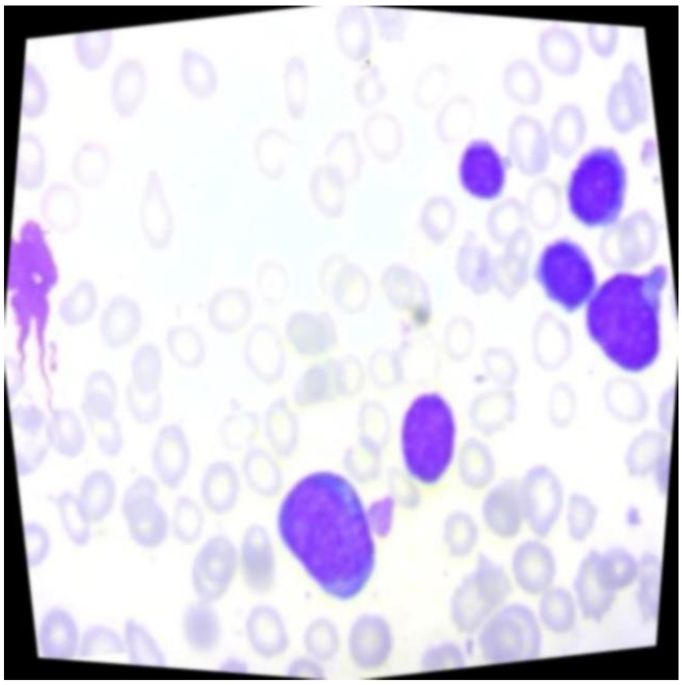


Fig 5.4.2.2 Vertically Flipped Image

➢ 5.4.3 Random rotation of between -30 and +30 degrees

Random rotation between -30 and +30 degrees involves randomly selecting an angle within the specified range and applying it to rotate the image during augmentation by affine transformation.

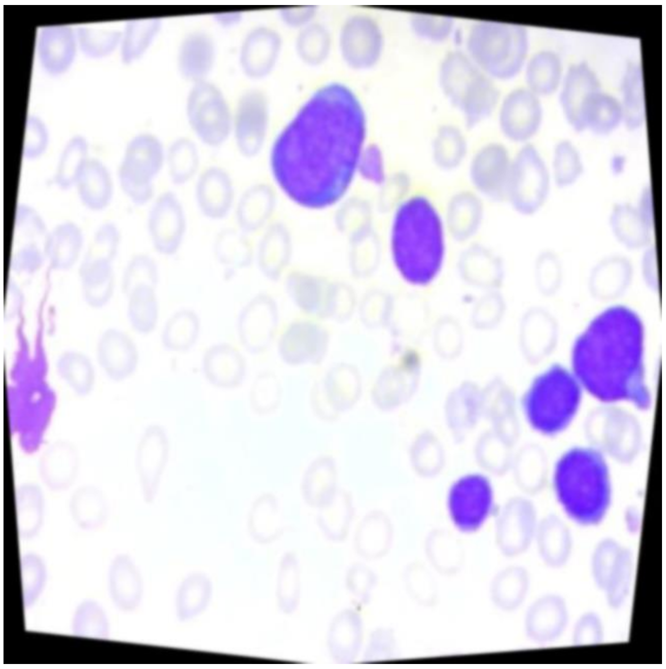


Fig 5.4.3.1 Original Image

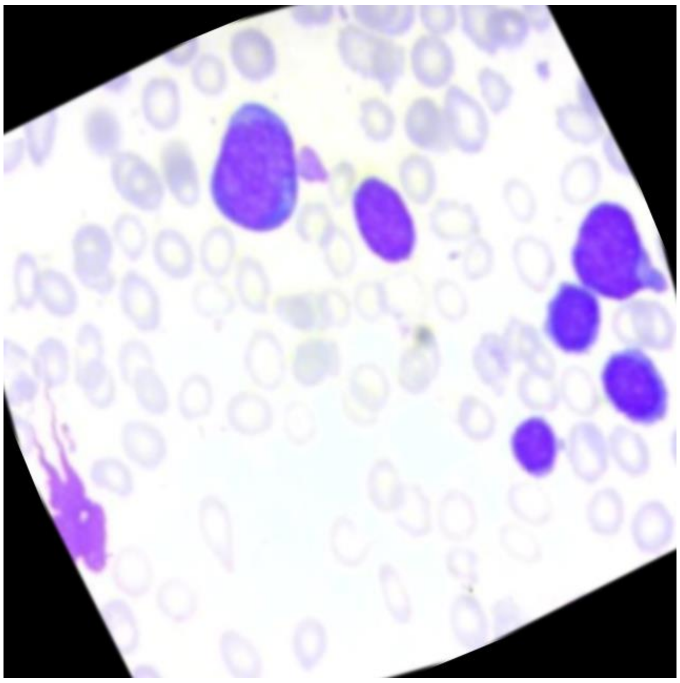


Fig 5.4.3.2 Rotated Image

➢ 5.4.4 Random brightness adjustment of between -15 and +15 percent

Random brightness adjustment between -15% and +15% involves randomly selecting a brightness factor within the specified range and applying it to adjust the brightness of the image during augmentation.

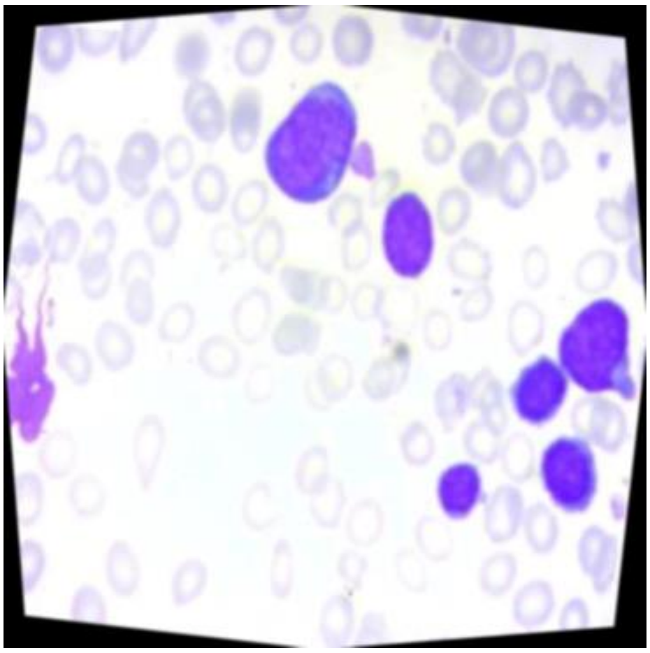


Fig 5.4.4.1 Original Image

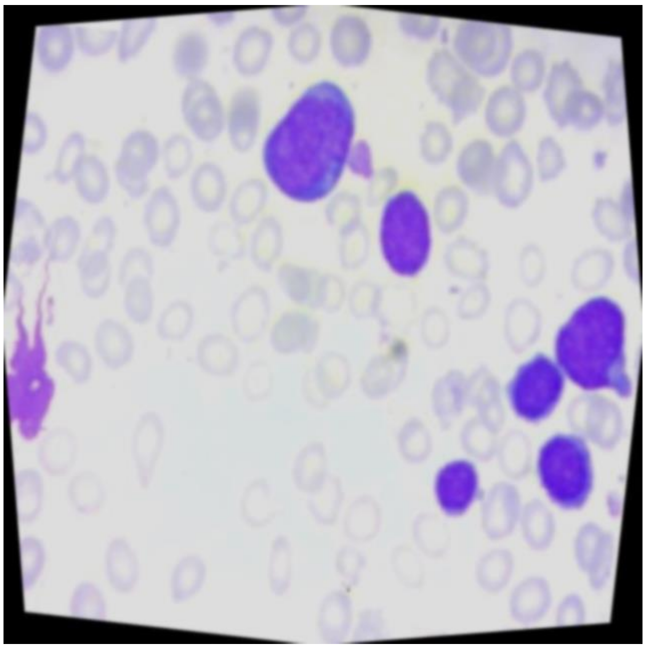


Fig 5.4.4.2 Brightness Adjusted Image (-15)

- During image augmentation, a brightness factor is randomly chosen from a uniform distribution ranging from -15% to +15%.
- This factor represents the percentage by which the brightness of the image will be adjusted.
- The brightness adjustment is applied to the image using a linear scaling operation, where pixel values are multiplied by the brightness factor.

➢ 5.4.5 Random Gaussian blur of between 0 and 2 pixels

Random Gaussian blur between 0 and 2 pixels involves randomly selecting a blur amount within the specified range and applying Gaussian blur to the image during augmentation.

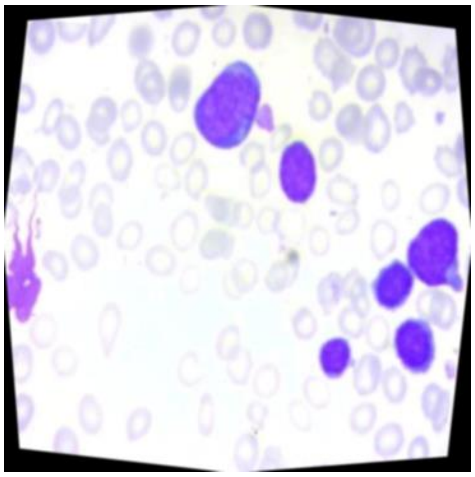


Fig 5.4.5.1 Original Image

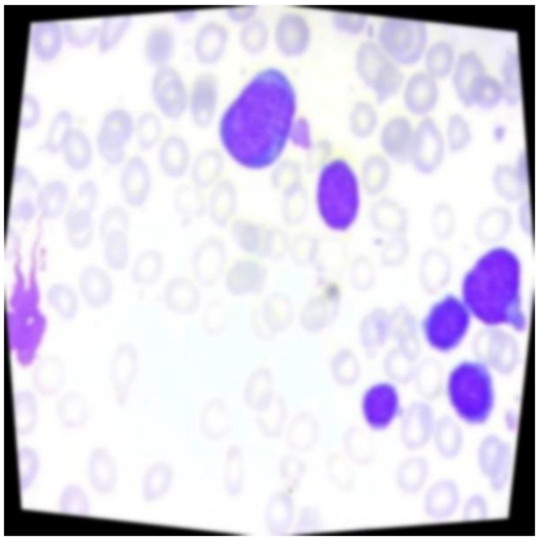


Fig 5.4.5.2 Blurred Image

- During image augmentation, a blur amount is randomly chosen from a uniform distribution ranging from 0 to 2 pixels.
- This amount determines the standard deviation of the Gaussian kernel used for blurring.
- Gaussian blur is then applied to the image using the selected kernel size.

By applying these processing and augmentation steps we created 3 versions of each source image. Which will be useful for improving the model accuracy and performance.

Overall, these augmentation techniques introduce variability into the training data, allowing the model to learn from a diverse set of images and improve its ability to generalize to unseen data.

6. MODEL IMPLEMENTATION

6.1 BUILDING CNN MODEL ARCHITECTURE

1. **Defining Input Parameters:**

   Input Shape specifies the dimensions of the input images the model expects. In this case, images are assumed to be 224 pixels high, 224 pixels wide, and have 3 color channels (RGB). A smaller size can reduce memory usage during training.
   Number of Objects defines the maximum number of objects (bounding boxes) the model will predict for each image in our dataset.

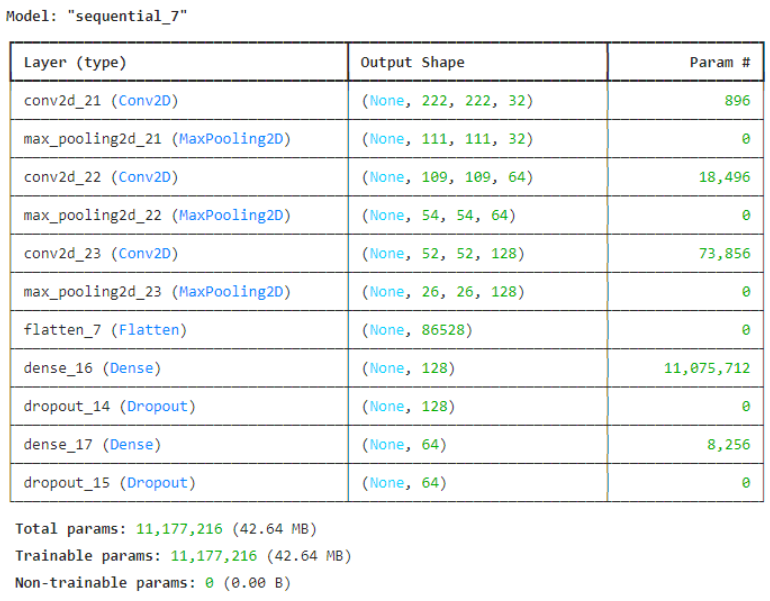
2. **Creating the CNN Model**:

   Convolutional Layers: These layers extract features from the input images. They apply filters of a specific size (e.g., 3x3) to small regions of the image, generating feature maps that capture patterns and edges. The model uses multiple convolutional layers with increasing numbers of filters to progressively learn more complex features.
   Activation Functions: These functions introduce non-linearity into the model, allowing it to learn more complex relationships between features. The code uses the ReLU (Rectified Linear Unit) activation function.
   Pooling Layers: These layers downsample the data, reducing its dimensionality. This can help control overfitting and make the model more efficient. Here, max pooling is used, which takes the maximum value from a specific window size (e.g., 2x2) to create a smaller output.
   Flatten Layer: After the convolutional layers, the data is flattened into a 1D vector before feeding it into dense layers.
   Dense Layers: These fully connected layers learn more intricate relationships between the extracted features. The code uses two dense layers with dropout (a regularization technique) to prevent overfitting. generalize to unseen data.

3. Training the Model

Now, we made a CNN model Architecture, the next step is to train and validate the model using training data and validation data.

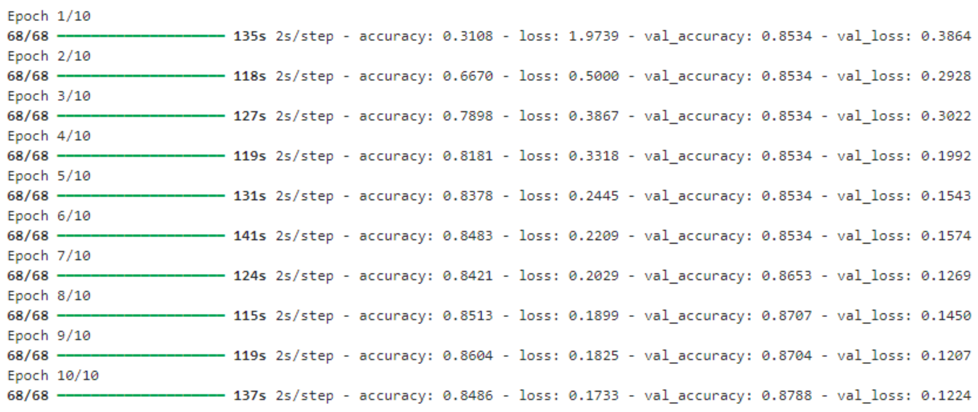
We passsed the directory path to the model and it loads the images and labels, preprocesses it by using preprocessing method. Then, the training and validiation images and labels are given to the model by using model.fit() function. Here, the size of the images are very large, to optimize it we performed batch processing operation.



```
Model: "sequential_7"
┌──────────────────────────────────┬─────────────────────────────┬──────────────┐
│ Layer (type)                     │ Output Shape                │      Param # │
├──────────────────────────────────┼─────────────────────────────┼──────────────┤
│ conv2d_21 (Conv2D)               │ (None, 222, 222, 32)        │          896 │
│ max_pooling2d_21 (MaxPooling2D)  │ (None, 111, 111, 32)        │            0 │
│ conv2d_22 (Conv2D)               │ (None, 109, 109, 64)        │       18,496 │
│ max_pooling2d_22 (MaxPooling2D)  │ (None, 54, 54, 64)          │            0 │
│ conv2d_23 (Conv2D)               │ (None, 52, 52, 128)         │       73,856 │
│ max_pooling2d_23 (MaxPooling2D)  │ (None, 26, 26, 128)         │            0 │
│ flatten_7 (Flatten)              │ (None, 86528)               │            0 │
│ dense_16 (Dense)                 │ (None, 128)                 │   11,075,712 │
│ dropout_14 (Dropout)             │ (None, 128)                 │            0 │
│ dense_17 (Dense)                 │ (None, 64)                  │        8,256 │
│ dropout_15 (Dropout)             │ (None, 64)                  │            0 │
└──────────────────────────────────┴─────────────────────────────┴──────────────┘
 Total params: 11,177,216 (42.64 MB)
 Trainable params: 11,177,216 (42.64 MB)
 Non-trainable params: 0 (0.00 B)
```

Fig 6.1.3.1 Training Parameter details

- Model is trained for 10 epochs with a batch size of 32 using training and validation data.



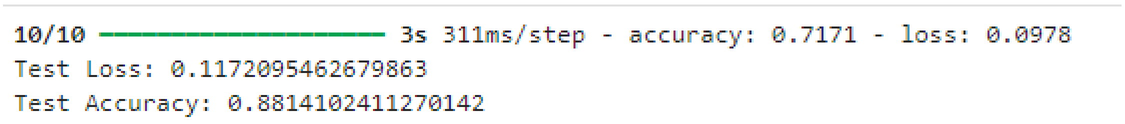
```
Epoch 1/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 135s 2s/step - accuracy: 0.3108 - loss: 1.9739 - val_accuracy: 0.8534 - val_loss: 0.3864
Epoch 2/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 118s 2s/step - accuracy: 0.6670 - loss: 0.5000 - val_accuracy: 0.8534 - val_loss: 0.2928
Epoch 3/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 127s 2s/step - accuracy: 0.7898 - loss: 0.3867 - val_accuracy: 0.8534 - val_loss: 0.3022
Epoch 4/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 119s 2s/step - accuracy: 0.8181 - loss: 0.3318 - val_accuracy: 0.8534 - val_loss: 0.1992
Epoch 5/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 131s 2s/step - accuracy: 0.8378 - loss: 0.2445 - val_accuracy: 0.8534 - val_loss: 0.1543
Epoch 6/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 141s 2s/step - accuracy: 0.8483 - loss: 0.2209 - val_accuracy: 0.8534 - val_loss: 0.1574
Epoch 7/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 124s 2s/step - accuracy: 0.8421 - loss: 0.2029 - val_accuracy: 0.8653 - val_loss: 0.1269
Epoch 8/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 115s 2s/step - accuracy: 0.8513 - loss: 0.1899 - val_accuracy: 0.8707 - val_loss: 0.1450
Epoch 9/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 119s 2s/step - accuracy: 0.8604 - loss: 0.1825 - val_accuracy: 0.8704 - val_loss: 0.1207
Epoch 10/10
68/68 ━━━━━━━━━━━━━━━━━━━━ 137s 2s/step - accuracy: 0.8486 - loss: 0.1733 - val_accuracy: 0.8788 - val_loss: 0.1224
```

Fig 6.1.3.2 Epoch

At last, at epoch 10 we gained accuracy of approx 84% and with a loss of 0.17%. And val_accuracy is approx 87% with a loss of 12%.

4. Evaluating the Model

After training the model, we have to evaluate the model performance. For this, we have test data, in which it contains the image and labels.

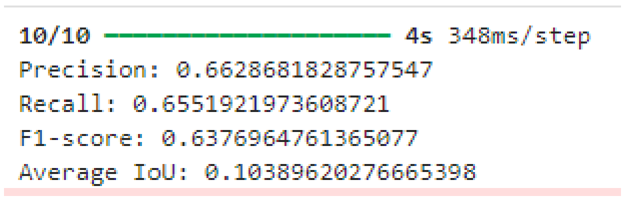


```
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 311ms/step - accuracy: 0.7171 - loss: 0.0978
Test Loss: 0.1172095462679863
Test Accuracy: 0.8814102411270142
```

Fig 6.1.4.1 Test Results

After Evaluating the model, we Test accuracy is appox 88% with a loss of approx 12%.

And we again calculated the metrics such as recall, precision, F1score.



```
10/10 ━━━━━━━━━━━━━━━━━━━━ 4s 348ms/step
Precision: 0.6628681828757547
Recall: 0.6551921973608721
F1-score: 0.6376964761365077
Average IoU: 0.10389620276665398
```

Fig 6.1.4.2 Metrics

7. WEB APPLICATION

The Web application is developed using HTML (Hyper Test Markup Language) and CSS (Cascading Style sheet). The web application was developed to detect the leukaemia cells and the type of leukaemia such as Benign, Malignant Early, Malignant Pre, Malignant pro.

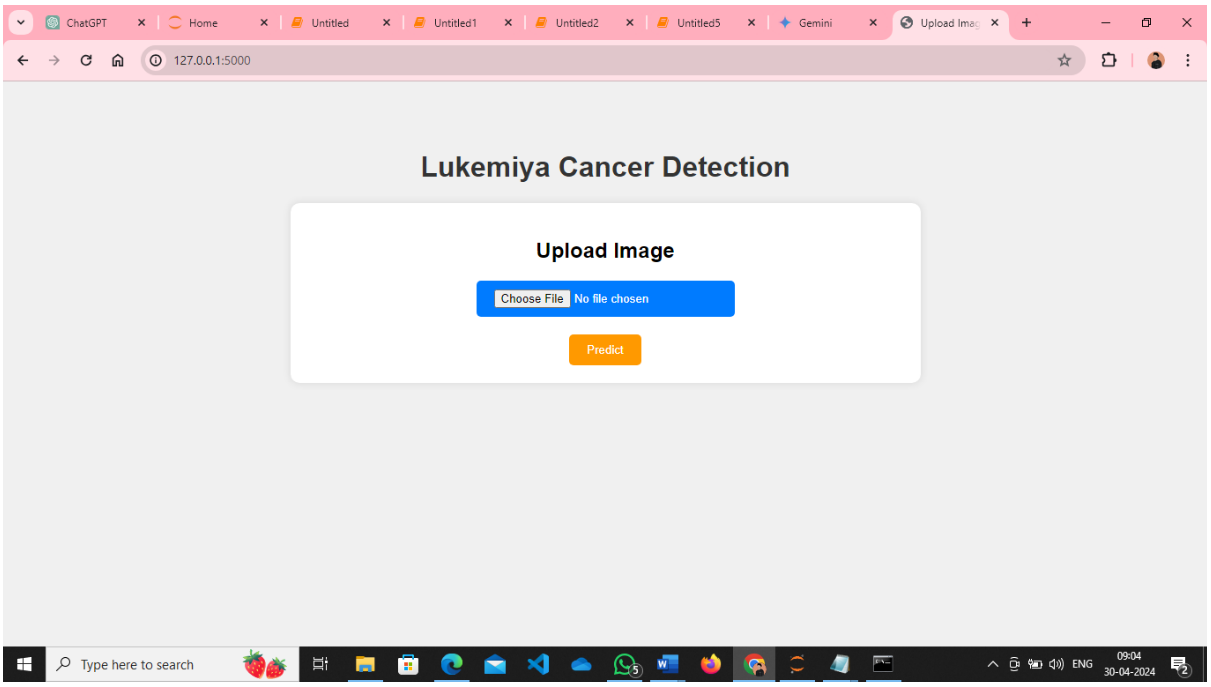
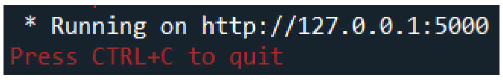


Fig 7.0.1 Web screen for leukemia cancer detection

The user uploads a microscopic leukemia cancer image, throught the html page by clicking on upload image, then the image is sent to the model, and the model predicts the type of leukemia present in that image.

7.1 RESULT

After uploading image through web application, we created a new Python file to define Flask application and we defined routes for our application. Here one route is used to render the form for users to input their data and another route to handle form submission and generate the result. When running a Flask application, the framework automatically assigns a port number for the application to listen on. By default, Flask uses port 5000. This means that when the Flask application is started, it will be accessible through a web browser using the appropriate port number. Accessing the application in a web browser would then be by navigating to "http://localhost:5000". The port number is crucial as it enables communication between the Flask application and the web browser, allowing users to interact with the web application seamlessly.



```
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Fig 7.1.1 Running Flask application on localhost

The URL "http://127.0.0.1:5000" you would be accessing a Flask web application running on your local machine through a web browser. Upon accessing this URL, web browser would display the home page in Flask application, allowing to interact with the web application's features and functionalities.

Fig 7.1.2 Web address of Flask Application

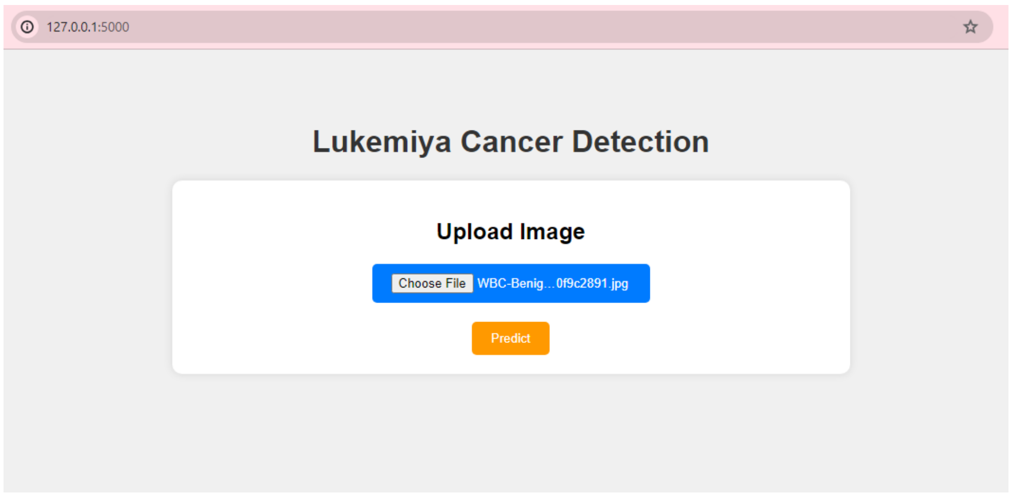
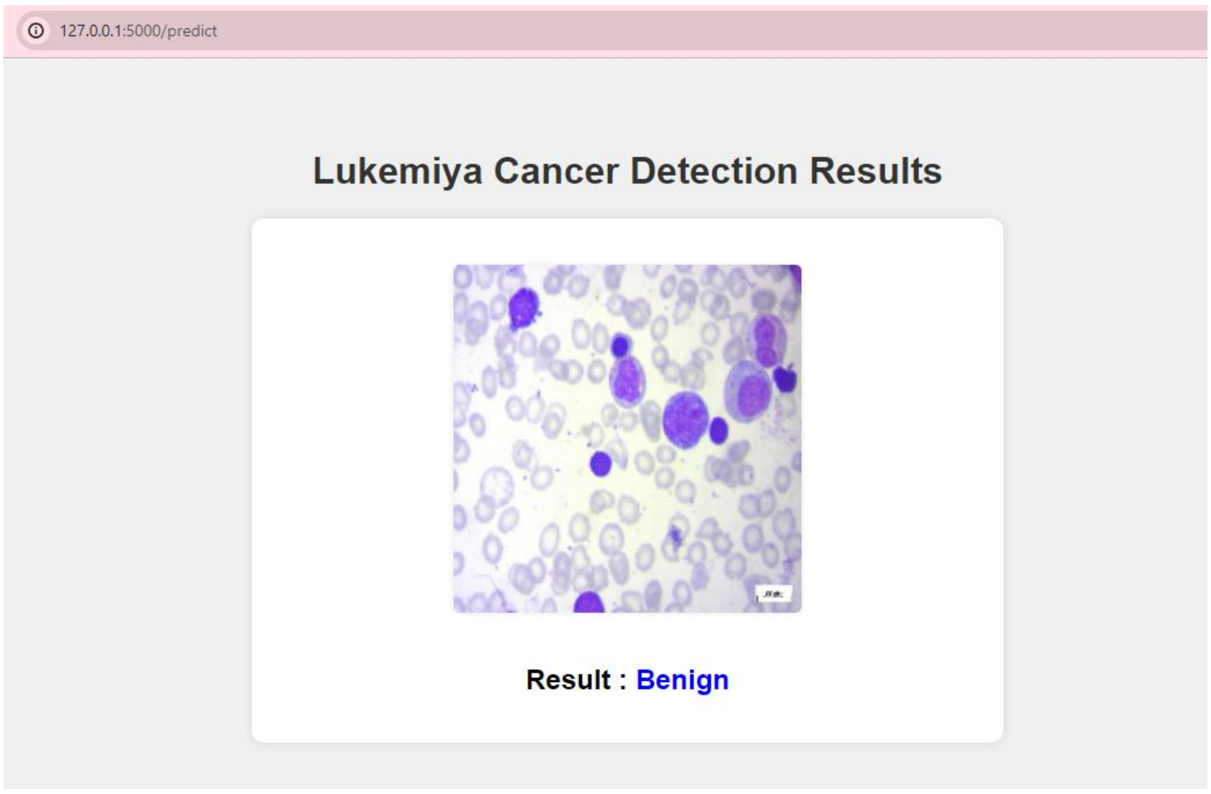


Fig 7.1.3 Uploading image

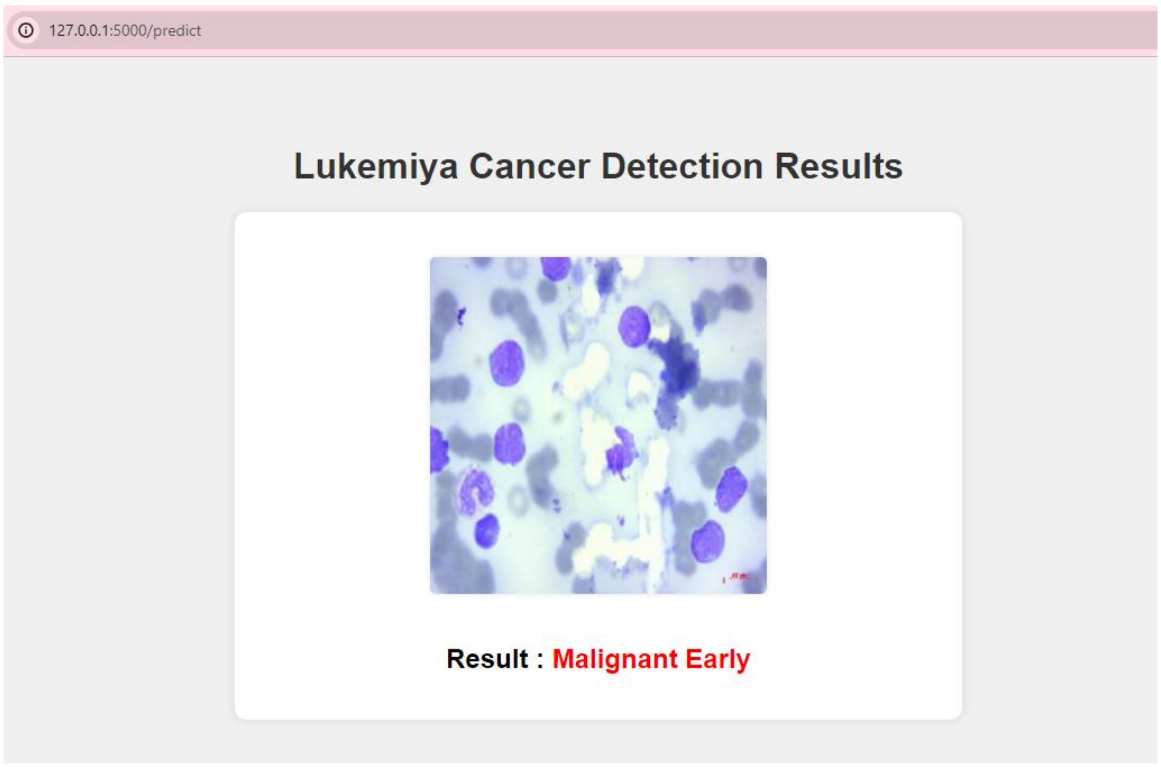
After, uploading the image, when we clicks on the predict button the type of leukemia cancer is displayed in results page.


127.0.0.1:5000/predict

Lukemiya Cancer Detection Results

Result : Benign

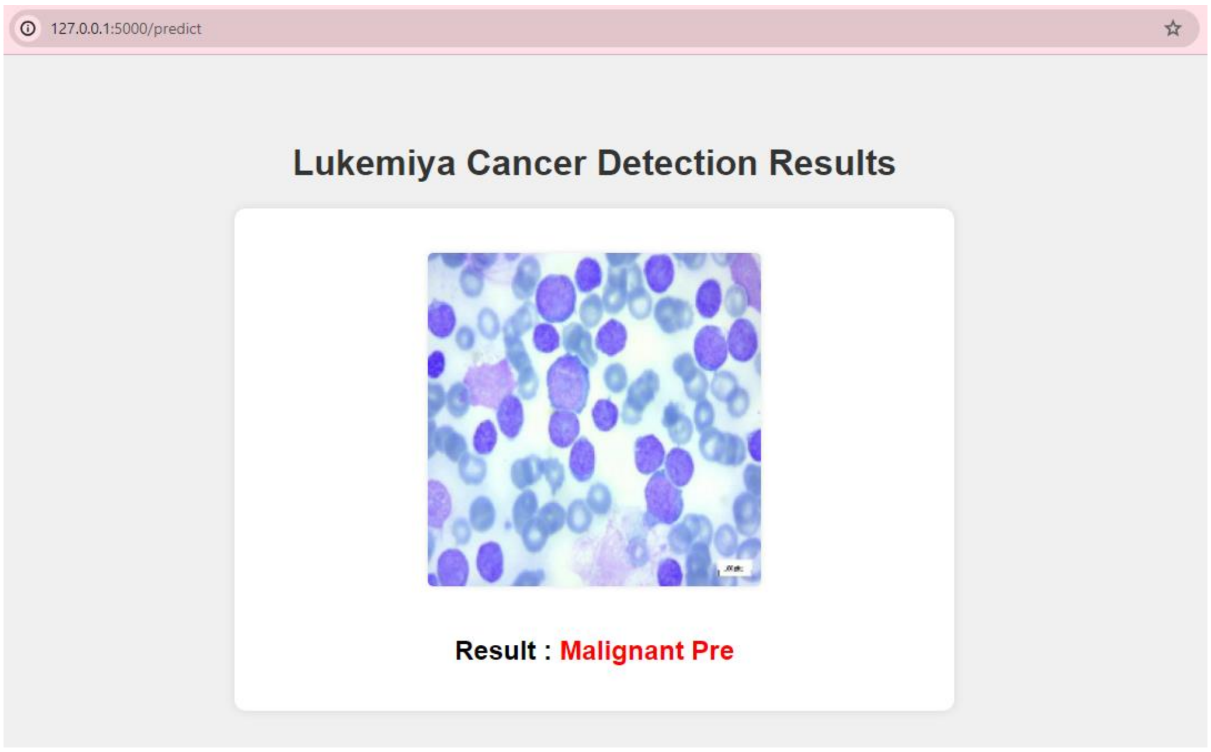
Fig 7.1.4 Result 1


127.0.0.1:5000/predict

Lukemiya Cancer Detection Results

Result : Malignant Early

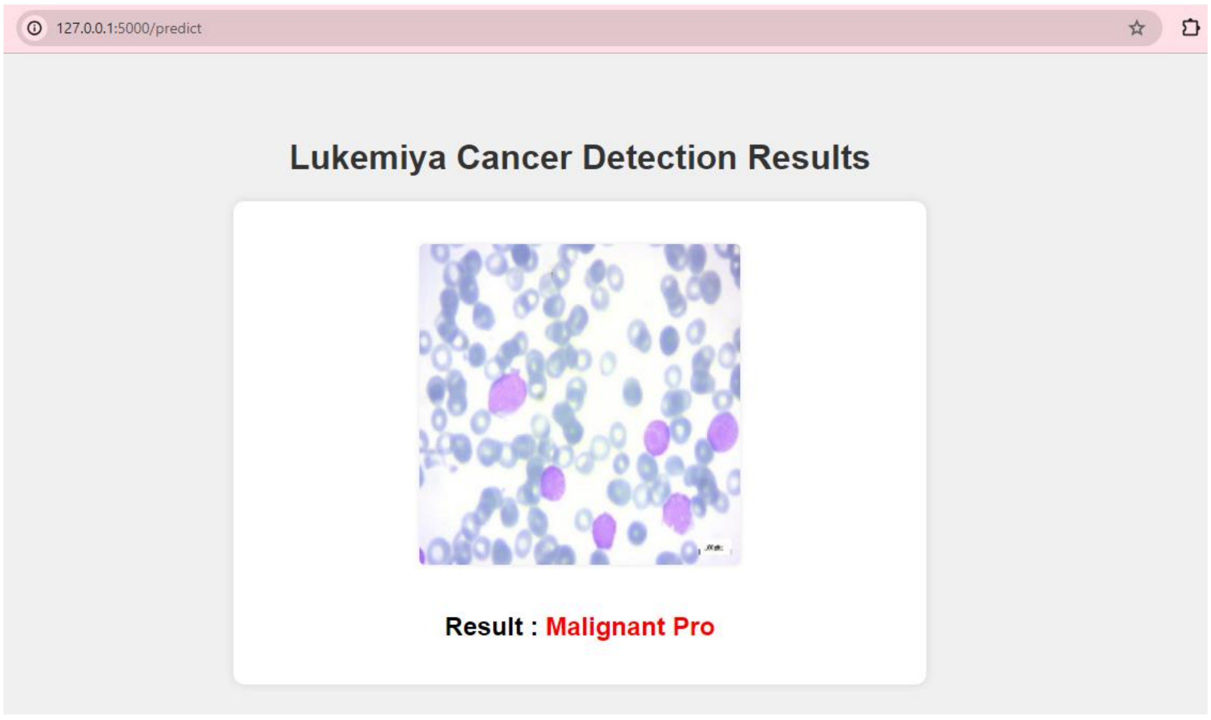
Fig 7.1.5 Result 2


127.0.0.1:5000/predict

Lukemiya Cancer Detection Results

Result : Malignant Pre

Fig 7.1.6 Result 3


127.0.0.1:5000/predict

Lukemiya Cancer Detection Results

Result : Malignant Pro

Fig 7.1.7 Result 4

8. CONCLUSION

Leukemia diagnosis is really important but tricky. Right now, doctors use microscopes to look at blood samples, which takes a long time and can sometimes be wrong. So, we're trying to use computers to help out. We've been testing different computer methods, like using special kinds of deep learning, to see if they can spot leukemia cells in microscope pictures faster and more accurately. Our main goal has been to create a computer program that can find leukemia cells early, which would help doctors treat patients better. We've looked at a lot of research and tried out different computer techniques to make this happen. Our hope is that by combining smart computer technology with real-life medical needs, we can make leukemia diagnosis easier and help more people get better. As we look back on our work, we're excited about the progress we've made. We believe that with teamwork and dedication, we can make a real difference in how leukemia is diagnosed, making it faster, more accurate, and ultimately helping more people live healthier lives.

9. FUTURE SCOPE

- **Model Optimization**: Continuously refine architecture and hyperparameters for improved accuracy.
- **Data Augmentation Exploration:** Experiment with diverse techniques to enrich training data diversity.
- **Transfer Learning Integration**: Adapt pre-trained models to leverage knowledge for leukaemia detection.
- **Real-Time Deployment**: Develop systems for immediate feedback from live data streams.
- **Clinical Validation:** Collaborate with healthcare institutions for real-world model testing.
- **Collaborative Research Initiatives:** Partner with experts to validate and refine the model's performance.
- **Open-Source Contribution**: Share code, datasets, and findings to foster community collaboration.

10.REFERENCES

- https://universe.roboflow.com/custom-yolov5-o0hdb/leukemia-cancer-detection
- https://universe.roboflow.com
- www.youtube.com
- https://towardsdatascience.com/detecting-leukemia-with-a-cnn-af699b19ab99