Volume: 11 Issue: IV Month of publication: April 2023

DOI: https://doi.org/10.22214/ijraset.2023.50459

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue IV Apr 2023- Available at www.ijraset.com

Generation of RBC, WBC Subtype, and Platelet Count Report Using YOLO

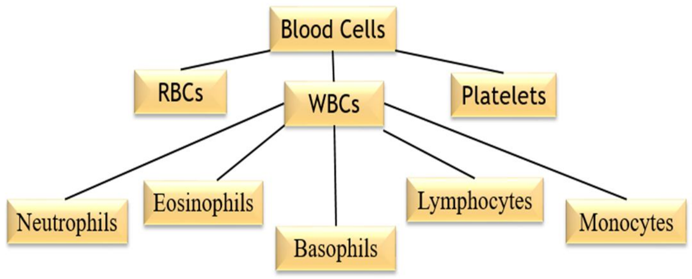
Vaishnavi Abhyankar[1], Rashmi Kene[2]
[1, 2]Computer Engineering Department, Datta Meghe College of Engineering, Navi Mumbai, Maharashtra, India

Abstract: Artificial intelligence introduced a way to combine machines' computing ability with human intelligence. Machine learning is a sub-branch of AI that consists of different algorithms to implement concepts of AI in practical terms. But when a machine has to process numerous amounts of data, deep learning algorithms come into the picture. It is observed that when a computing system has to deal with image data then neural network algorithms give efficient ways to process them and draw unique patterns from them. Object detection is the task of identifying required objects from an image. This type of technology plays a crucial role in medical image processing. Some algorithms can efficiently identify and classify objects from an image. It is observed that Fast R-CNN and Faster R-CNN, mask R-CNN, have given pretty good accuracy while performing such tasks. But when time is a concern for a system, such methods put an obstacle of training time and architectural complexity. The You Only Look Once (YOLO) object detection method has introduced a new way of processing images in a single pass. This algorithm is famous because of its speed and correctness. There are numerous models of YOLO developed now which include YOLOv1 to YOLOV8. This paper gives the performance of the latest version of YOLO on the blood cell dataset.

***Keywords:** YOLO, CBC, YOLOv5, YOLOv7, YOLOv8, Deep Learning*

I. INTRODUCTION

Blood is a fluid connective tissue. It is an important parameter of a patient's health check-up. Human blood is composed of 55% of liquid and 45% of solid components. The liquid part is called plasma which consists of proteins, dissolved salts, and nutrients whereas the solid part consists of cellular structures which include White Blood Cells, Red Blood Cells Platelets, etc. White blood cells are also called attacking cells as they invade unwanted or foreign bodies that entered the bloodstream. Several types of WBCs attack differently on foreign bodies. Their count helps doctors to understand the overall disease spread in the body. RBCs play an important role in supplying oxygen to all body organs. Platelets are also known as thrombocytes which generate clots and stop bleeding. They are important to surviving and fighting cancer, chronic diseases, and traumatic injuries.

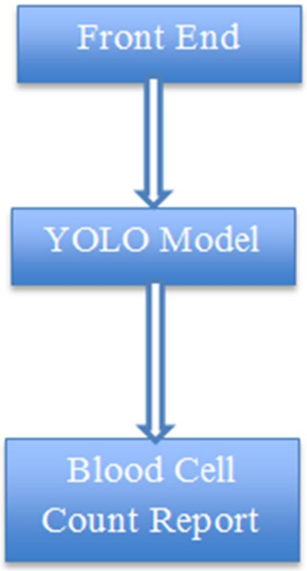


In today's busy lifestyle people are quite unaware of their health and it causes serious health problems. Most of the time many diseases are concerned with blood, so doctors often ask patients to do their blood tests. In this test, we get the report which consists of the exact count of blood cells present in the blood smear. This count helps physicians in diagnosing diseases, deciding therapy plans, and predicting the stage of a particular disease like cancer. To perform these kinds of tests generally patients visit pathological labs for blood tests, where nurses or medical specialists take a drop of the blood, and then pathologists analyze and perform counting of blood cells by placing slides of blood under a microscope. This process is quite time-consuming and needs 4 to 5 hours for final report generation as pathologists manually perform counting. So, to make this process easier and more efficient we have developed a model which will take the image of a blood smear, process it automatically and give the report of blood cell count in just a few minutes. There are several models which worked on this idea but their architectural and implementational complexity made it time-consuming and tedious. Previously developed models have worked individually on specialized blood cells.

But in this paper, we are introducing a way of detecting and counting all WBCs subtypes along with platelets and RBCs using the latest version of the You Only Look Once, Model

II. METHODOLOGY

This section will give a detailed explanation of the flow of the system and its implementation



Let us now understand each section in detail:

A. Front End

This is nothing but a user interface that hides the complexity of the model. This is implemented in the python flask framework. This section will also accept peripheral blood smear images in the intended format.
The image should be in JPG, JPEG, and PNG format. In this system, the user is nothing but a lab technician. This user interface will accept some more details like the patient’s full name, reference doctor name, age gender, etc. Input from this stage will serve as input to the YOLO framework.

B. YOLO Model

You Only Look Once (YOLO) is one of the widely used computer vision models. The YOLO family of models has continued to evolve since its initial release in 2016. The latest version of YOLO is version 8. The original YOLO model was the first object detection network to combine the problem of drawing bounding boxes and identifying class labels in one end-to-end differentiable network. YOLO is a single-stage detector, handling both object identification and classification in a single pass of the network. YOLO is not the only single-stage detection model but it is generally more preferment in terms of speed and accuracy. Before YOLO, R-CNNs were among the most commonly used method for detecting objects, but when accuracy and speed is a things of concern then YOLO is the winner of this competition. Each updated version of YOLO has some unique improvements to the previous one. This upgradation may be in terms of the backend neural network or way of training the model.
Let us now have a look at different versions of YOLO models

1) YOLOv1

It was first established in the year 2015 by Joseph Redmon et al. Its main characteristic was its Single stage unified detection. It was the first single-stage object detector approach that treated detection as a regression problem. The detection architecture only looked once at the image to predict the location of the objects and their class labels. it uses a single neural network that predicts class probabilities and bounding box coordinates from an entire image in one pass. YOLOv1 has achieved 63.4 mAP

2) YOLOv2

It was established in the year 2016 by Joseph Redmon and Ali Farhadi. YOLOv2 was trained on detection datasets like Pascal VOC and MS COCO. YOLOv2 has achieved 76.8 mAP. Its characteristic was the introduction of anchor boxes and multi-scale training.

3) YOLOv3

It was established in the year 2018 by Joseph Redmon and Ali Farhadi. They introduced a new network architecture called Darknet-53. This was a more efficient and fast neural network architecture. YOLOv3 has achieved 28.2 mAP. Its characteristic was a feature pyramid network (FPN)

4) YOLOv4

It was established in the year 2022 by Meituan. This kind of model consists of a backbone, neck, and head. The backbone can be a pre-trained convolutional neural network such as VGG16 or CSPDarkNet53 trained on COCO or ImageNet data sets. The backbone of the YOLO v4 network acts as the feature extraction network that computes feature maps from the input images. The neck connects the backbone and the head. The neck concatenates the feature maps from different layers of the backbone network and sends them as inputs to the head. The head processes the aggregated features and predicts the bounding boxes, objectness scores, and classification scores.

5) YOLOv5

It was established in the year 2020 by Ultralytics. YOLOv5 offers a family of object detection architectures pre-trained on the MS COCO dataset. YOLOv5 was implemented in the PyTorch environment which eliminated the Darknet framework's limitations. YOLOv5 was one of the official state-of-the-art models in the Torch Hub. There were different sub-models of YOLOv5 like YOLOv5l, YOLOv5m, YOLOv5s, YOLOv5x, and YOLOv5n which can be used depending on the size of the dataset.

6) YOLOv6

It was established in the year 2022 by Meituan. YOLO v6 differed from YOLOV5 in terms of the CNN architecture used. YOLO v6 used a variant of the EfficientNet architecture called EfficientNet-L2. In YOLO v6 new methods of anchor boxes called Dense Anchor boxes were introduced.

7) YOLOv7

It was introduced in the year 2022 by Chien-Yao Wang et all. Improvement in YOLO v7 was the use of an efficient loss function called focal loss. It has been observed that higher resolution than the previous version. YOLO v7 was efficient because of its speed as it can process images at a rate of 155 frames per second

8) YOLOv8

This model is the latest version of YOL introduced in the year 2023 by ultralytics. Its characteristic is that the head portion of the model is free from anchor boxes. It is observed that it takes less time for training. One of the remarkable changes in this version is that while training if the model founds that there is no change in the weights then early stopping is carried out by the system itself. This is the key benefit of using
YOLOv8. Such improvement helps in preventing the model from overfitting.
The YOLOv8-based system can be broken down into major four steps:

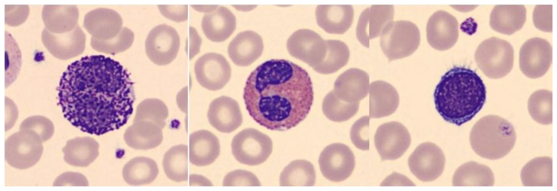
a) Creating and annotating the dataset
b) Model Training
c) Saving the best weights
d) Validation and deployment

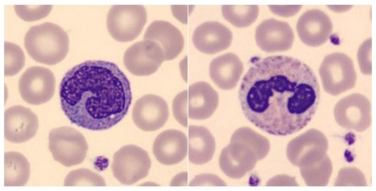
Let us understand these steps in brief:

- Creating and annotating dataset-

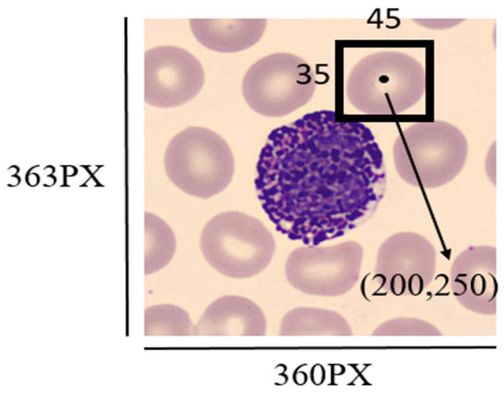
In this, we first created a dataset from a set of images by adding some noise and rotating the images to different angles. As initially we have a lesser number of images to use as the dataset, we have done this step.

Images from Dataset


Basophil Eosinophil Lymphocyte


Monocyte Neutrophil

To ease the entire process of training and understanding and optimizing the performance we have created annotations in YOLO format only instead of going with the standard XML format. In the case of XML or CSV format, it first needs to be converted into a YOLO-like structure which was quite a tedious process. There is a library called LabelIMG in Python itself to ease the process of the labeling image dataset. But to save time we have used an online available tool for image annotation tasks.
YOLO Data Annotation format is:
<object-class> <x> <y> <width> <height
Here object classes are referred to as classes that are needed to be identified e.g. dog, cat, etc.
x is referred to as the center x-coordinate of the bounding box. It is calculated as actual_center_x_coordinate_of_bounding_box /width_ of_the_image.
y is referred to as the center y-coordinate of the bounding box. It is calculated as actual_center_y_coordinate_of_bounding_box /height_ of_the_image.
width is referred as width_of_the_bounding_ box/width_of_the_image
height is referred as height_of_the_bounding_box/height_of_the_image



SO HERE
x=200/360 y=250/363 width=45/360 height=35/363

- *Model Training*

There are two modern techniques are introduced to improve object detection and classification: Mosaic data augmentation and Class-specific anchor boxes. Mosaic data augmentation is a data augmentation technique that mixes up images to create new training images. This ultimately helps to increase the diversity of the training data resulting in the prevention of overfitting and improving generalization. Mosaic data augmentation combines 4 training images into one in random proportions. The algorithms are the following:

- Take 4 images from the train set;
- Resize them to the same size;
- Integrate the images into a 4x4 grid;
- Crop a random image patch from the center.

Class-specific anchor boxes are anchor boxes that are specific to each class, allowing the model to better match the shape and aspect ratio of objects in the target class. In YOLO v8 new model architecture called SPP-YOLO Spatial Pyramid Pooling YOLO is introduced which helps to handle objects at different scales.

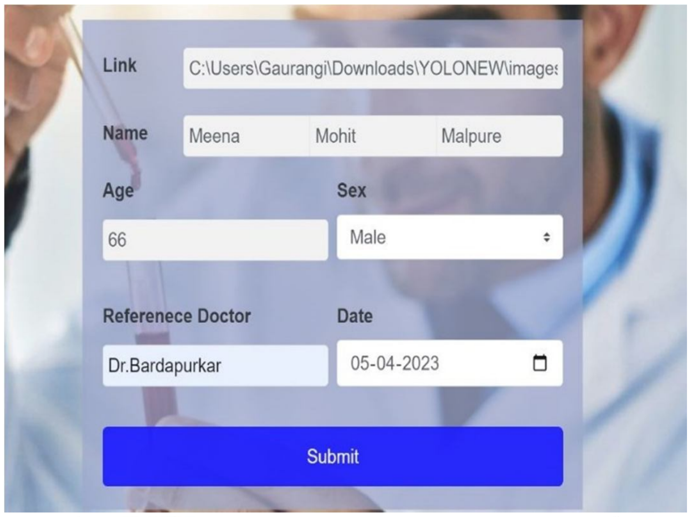
- *Saving Best Weights*

After training the pre-trained YOLO model for 93 epochs weights are automatically get saved in best.pt file.

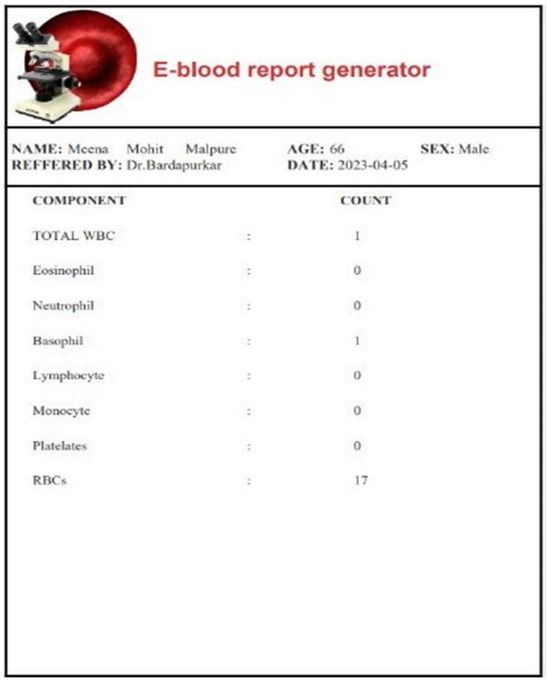
- *Validation And Deployment*

In YOLOV8 we have performed validation using images saved in the val directory. For that model .val() method have used. Here we have used best weights for further classification.

- *User Interface and Final Result*



Link: C:\Users\Gaurangi\Downloads\YOLONEW\images
Name: Meena Mohit Malpure
Age: 66
Sex: Male
Referenece Doctor: Dr.Bardapurkar
Date: 05-04-2023



E-blood report generator

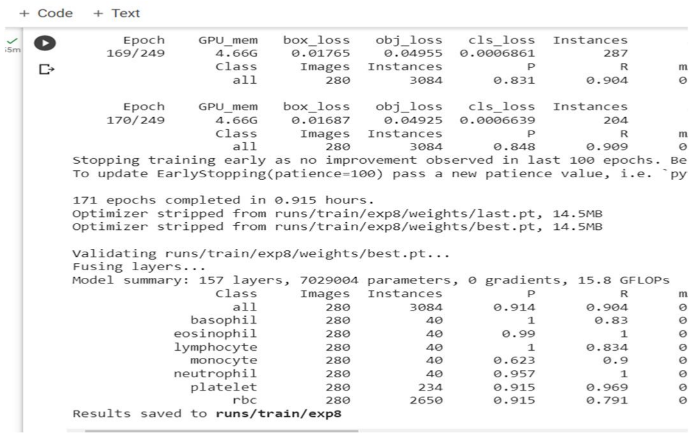
NAME: Meena Mohit Malpure **AGE:** 66 **SEX:** Male
REFFERED BY: Dr.Bardapurkar **DATE:** 2023-04-05

COMPONENT		COUNT
TOTAL WBC	:	1
Eosinophil	:	0
Neutrophil	:	0
Basophil	:	1
Lymphocyte	:	0
Monocyte	:	0
Platelates	:	0
RBCs	:	17

Comparison of YOLOv5, YOLOv7 and YOLOv8
Comparison based on training time and accuracy

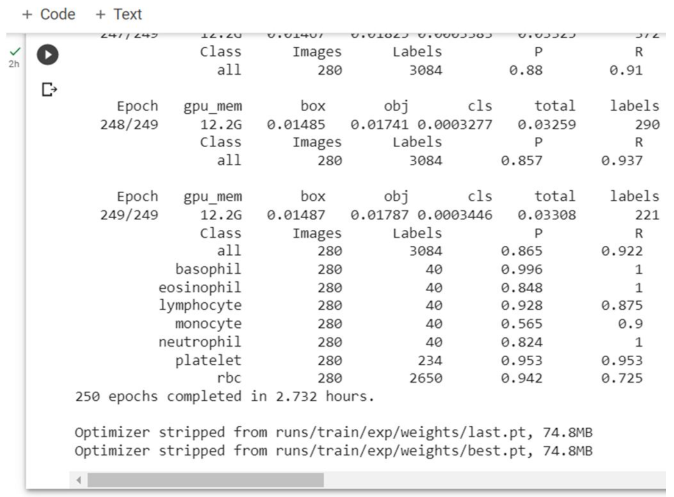
- *Training time for 250 epochs*

➢ *YOLOv5:* 0.915 hours, It Stopped training early because no improvement was observed in the last 100 epochs. Hence although we have run it for 250 epochs, it completed the training at 171 epochs.



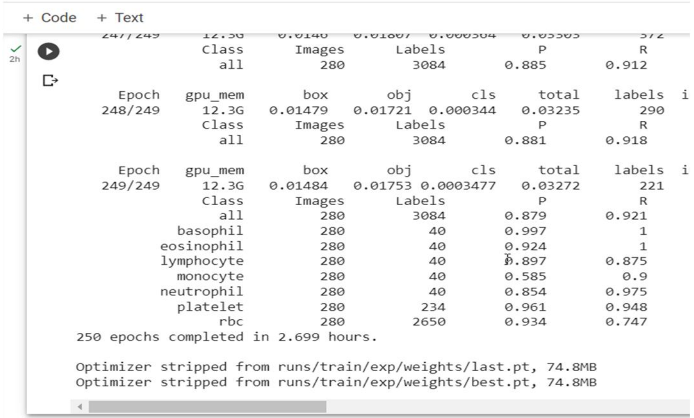
```
      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances
    169/249      4.66G    0.01765    0.04955  0.0006861        287
                 Class     Images  Instances          P          R      m
                   all        280       3084      0.831      0.904      0

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances
    170/249      4.66G    0.01687    0.04925  0.0006639        204
                 Class     Images  Instances          P          R      m
                   all        280       3084      0.848      0.909      0
Stopping training early as no improvement observed in last 100 epochs. Be
To update EarlyStopping(patience=100) pass a new patience value, i.e. `py

171 epochs completed in 0.915 hours.
Optimizer stripped from runs/train/exp8/weights/last.pt, 14.5MB
Optimizer stripped from runs/train/exp8/weights/best.pt, 14.5MB

Validating runs/train/exp8/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7029004 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      m
                   all        280       3084      0.914      0.904      0
              basophil        280         40          1       0.83      0
            eosinophil        280         40       0.99          1      0
            lymphocyte        280         40          1      0.834      0
              monocyte        280         40      0.623        0.9      0
            neutrophil        280         40      0.957          1      0
              platelet        280        234      0.915      0.969      0
                   rbc        280       2650      0.915      0.791      0
Results saved to runs/train/exp8
```

➢ *YOLOv7:* It took 2.732 hours for training the model.



```
               Class      Images      Labels           P           R
                 all         280        3084        0.88        0.91

     Epoch   gpu_mem       box       obj       cls     total    labels
   248/249     12.2G   0.01485   0.01741 0.0003277   0.03259       290
               Class      Images      Labels           P           R
                 all         280        3084       0.857       0.937

     Epoch   gpu_mem       box       obj       cls     total    labels
   249/249     12.2G   0.01487   0.01787 0.0003446   0.03308       221
               Class      Images      Labels           P           R
                 all         280        3084       0.865       0.922
            basophil         280          40       0.996           1
          eosinophil         280          40       0.848           1
          lymphocyte         280          40       0.928       0.875
            monocyte         280          40       0.565         0.9
          neutrophil         280          40       0.824           1
            platelet         280         234       0.953       0.953
                 rbc         280        2650       0.942       0.725
250 epochs completed in 2.732 hours.

Optimizer stripped from runs/train/exp/weights/last.pt, 74.8MB
Optimizer stripped from runs/train/exp/weights/best.pt, 74.8MB
```

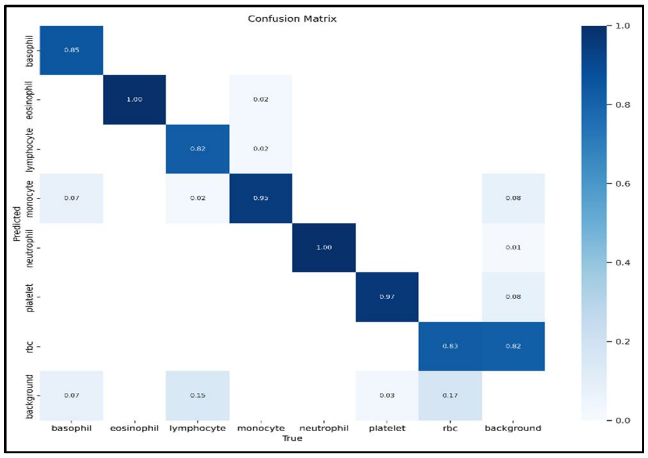
➢ *YOLOv8:* It took 2.699 hours for training the model.



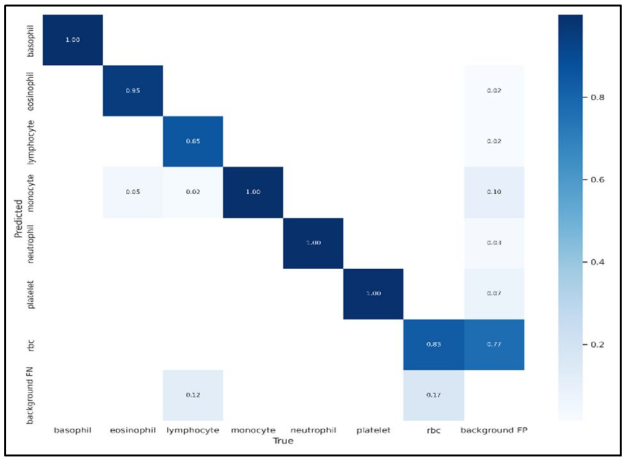
```
               Class      Images      Labels           P           R
                 all         280        3084       0.885       0.912

     Epoch   gpu_mem       box       obj       cls     total    labels  i
   248/249     12.3G   0.01479   0.01721  0.000344   0.03235       290
               Class      Images      Labels           P           R
                 all         280        3084       0.881       0.918

     Epoch   gpu_mem       box       obj       cls     total    labels  i
   249/249     12.3G   0.01484   0.01753 0.0003477   0.03272       221
               Class      Images      Labels           P           R
                 all         280        3084       0.879       0.921
            basophil         280          40       0.997           1
          eosinophil         280          40       0.924           1
          lymphocyte         280          40       0.897       0.875
            monocyte         280          40       0.585         0.9
          neutrophil         280          40       0.854       0.975
            platelet         280         234       0.961       0.948
                 rbc         280        2650       0.934       0.747
250 epochs completed in 2.699 hours.

Optimizer stripped from runs/train/exp/weights/last.pt, 74.8MB
Optimizer stripped from runs/train/exp/weights/best.pt, 74.8MB
```

From this, we can observe that as we go with the higher versions of YOLO the training time decreases.

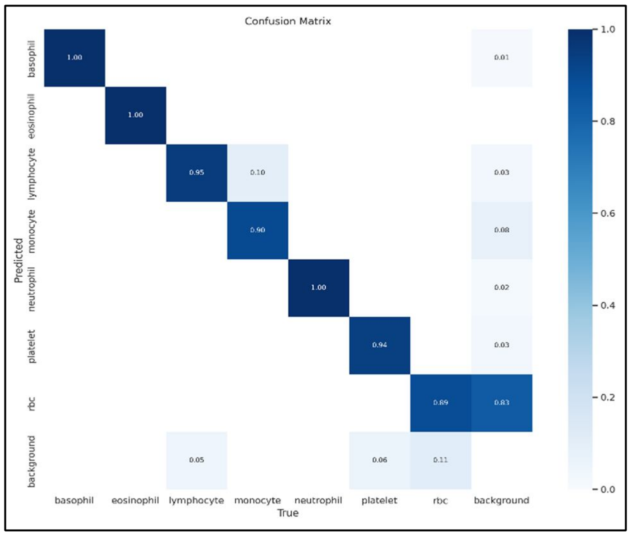
- *Accuracy for 250 epochs*
- *YOLOV5:* From the confusion matrix we have calculated its accuracy as 98.01% at 171 epoch early stop.



- *YOLOV7:* From the confusion matrix we have calculated its accuracy as 98.96%.



- *YOLOV8:* From the confusion matrix, we have calculated its accuracy as 98.52%.



III. CONCLUSIONS

This paper gives us brief views of the YOLO algorithm and its different versions. Here we have proposed a model for generating the blood cell count by using the YOLOv8 version which is the fastest and latest version of the YOLO algorithm. Here we have mentioned the step-wise description of each process of the model. The project is divided into two parts front end and the YOLO algorithm model. Here we have discussed the results of the YOLOv5, YOLOv7, and YOLOv8 versions which are compatible with the torch framework of Python. There is still room for future improvement in terms of the speed and accuracy of the YOLO model.

REFERENCES

[1] D. Kornack and P. Rakic, "Cell Proliferation without Neurogenesis in Adult Primate Neocortex," Science, vol. 294, Dec. 2001, pp. 2127-2130, doi:10.1126/science.1065467.
[2] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.
[3] R. Nicole, "Title of paper with the only first word capitalized," J. Name Stand. Abbrev., in press.
[4] K. Elissa, "Title of paper if known," unpublished.
[5] A Review of Yolo Algorithm Developments by Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, Bo Ma
[6] You Only Look Once: Unified, Real-Time Object Detection Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi
[7] Object Detection through Modified YOLO Neural Network Tanvir Ahmad, Yinglong Ma, and Amin ul Haq
[8] Real-Time Object Detection with Yolo Geethapriya. S, N. Duraimurugan, S.P. Chokkalingam
[9] Analytical Study on Object Detection using Yolo Algorithm Dawn Wilson1, Dr. Manusankar C.2, Dr. Prathibha
[10] A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector
[11] Object detection using YOLO: challenges, architectural successors, datasets and applications Tausif Diwan, G. Anirudh & Jitendra V. Tembhurne
[12] YOLOX: Exceeding YOLO Series in 2021 Zheng Ge∗ Songtao Liu∗† Feng Wang Zeming Li Jian Sun
[13] YOLOv4: Optimal Speed and Accuracy of Object Detection Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao
[14] A comparative study of YOLOv5 models performance for image localization and classification Marko Horvat, Gordan Gledec
[15] Evolution of YOLO-V5 Algorithm for Object Detection: Automated Detection of Library Books and Performace Validation of Dataset
[16] Evolution of YOLO Algorithm and YOLOv5: The State-of-the-art Object Detection Algorithm
[17] YOLOv6 v3.0: A Full-Scale Reloading Chuyi Li∗ Lulu Li∗ Yifei Geng∗ Hongliang Jiang∗ MengCheng Bo Zhang Zaidan Ke Xiaoming Xu† Xiangxiang Chu Meituan Inc.
[18] YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications
[19] YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors Chien-Yao Wang1, Alexey Bochkovskiy, and Hong-Yuan Mark Liao1 1Institute of Information Science, Academia Sinica, Taiwan