Blood Cell Detection

Angel Macwan
armacwan@gmail.com
angelmacwan.github.io

Introduction

Classifying different types of blood cells is a critical task in medical imaging, which can help in the diagnosis and treatment of various blood-related diseases. In recent years, computer vision techniques, particularly deep learning models, have been widely used to perform this task. Two popular models in object detection and image classification are **YOLO (You Only Look Once)** and **R-CNN (Regions with Convolutional Neural Networks)**.

This project aims to compare YOLO and R-CNN on a blood cell classification dataset and evaluate their performance in terms of accuracy and efficiency. The study provides valuable insights into the suitability of these models for blood cell classification tasks and helps in selecting the most appropriate model for similar applications.

Data

Data can be downloaded from the following link.
https://www.kaggle.com/datasets/draaslan/blood-cell-detection-dataset/download?datasetVersionNumber=1

This dataset contains 100 labeled images of White Blood Cells (WBC) and Red Blood Cells (RBC) combined. A separate file named `annotations.csv` is provided that contains labeling for each object in the given image.

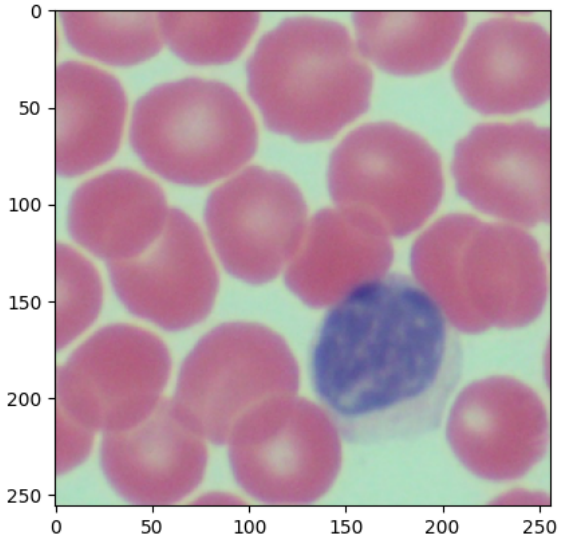

fig 1 Sample Image file (Unlabeled)

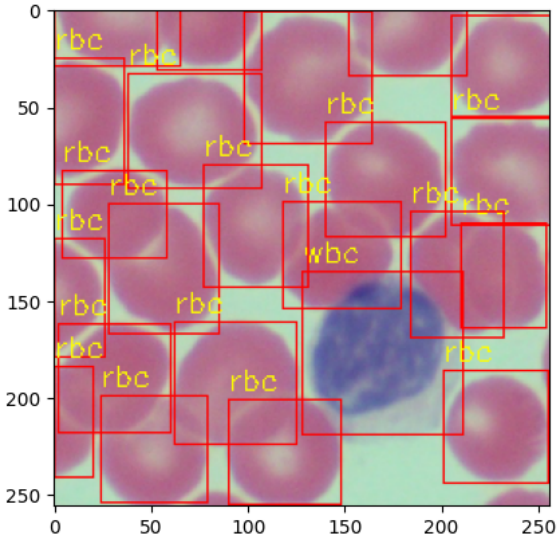

fig 2 Sample Labeled image

To train various deep learning models, this data has to be converted to the format that is expected by the model. This can be done by a python script or by using tools like robot flow. For demonstration purposes, both methods have been illustrated. The python script is used to generate data in YOLOv5 format. and RoboFlow is used to convert data to YOLOv8 and COCO format.

YOLOv8 format is used to train YOLOv8 model and the COCO format dataset is used to train the RCNN model

YOLO

YOLO (You Only Look Once) is a real-time object detection model designed for real-time object detection. The key idea behind YOLO is that it **only performs one forward pass** through the network, making it much faster than other object detection systems that may require multiple forward passes.

For this project, YOLO version 8 was used which is a newer/revised version of YOLO. YOLO on its own is much faster compared to traditional methods such as R-CNN but it is also less accurate. Although YOLO has made a significant improvement and is closing the gap.

Transfer learning was used to retrain or fine-tune the model to the new dataset. The model was trained on 80% data (80 images out of 100) for 30 epochs (20 epochs would be efficient). As the model was trained and then finetuned on a specific dataset, not a lot of data is required.

There are various versions of YOLO available based on the number of parameters as seen below. For this demonstration, **YOLOv8m** was used which is a medium size model with 25.9 million parameters.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

fig 3 YOLO Models

YOLO takes care of image transformations and augmentations internally, resulting in the application of various modifications such as rotation, scaling, and others to the images during the training process. This helps to enhance the robustness of the model by exposing it to a diverse range of variations.

Here are some results from the YOLO model

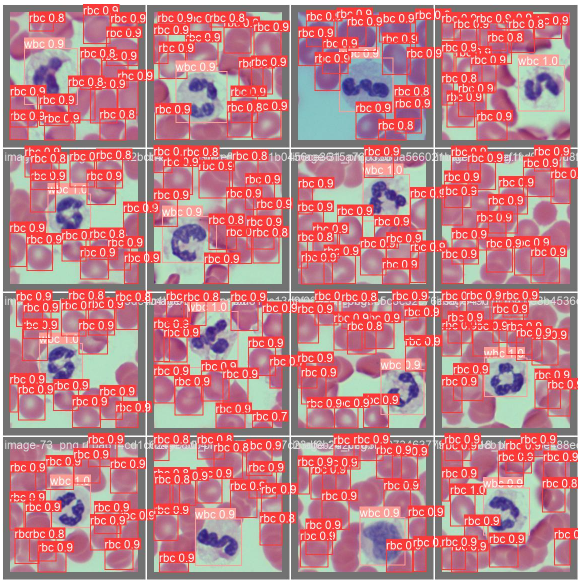

fig 4

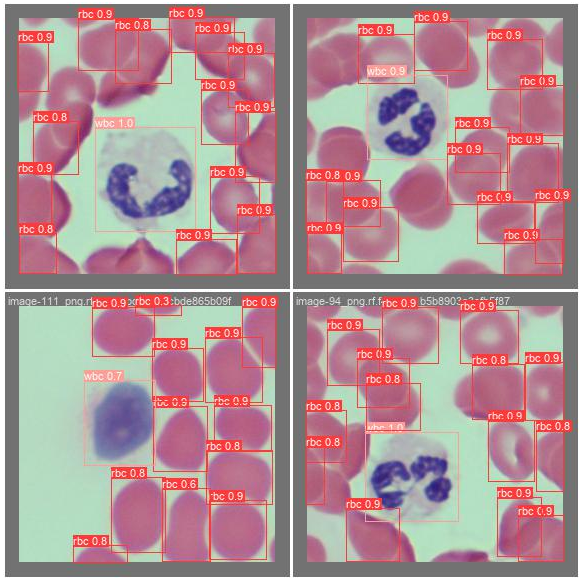

fig 5

As seen, the model is accurately able to identify the labels in the images.

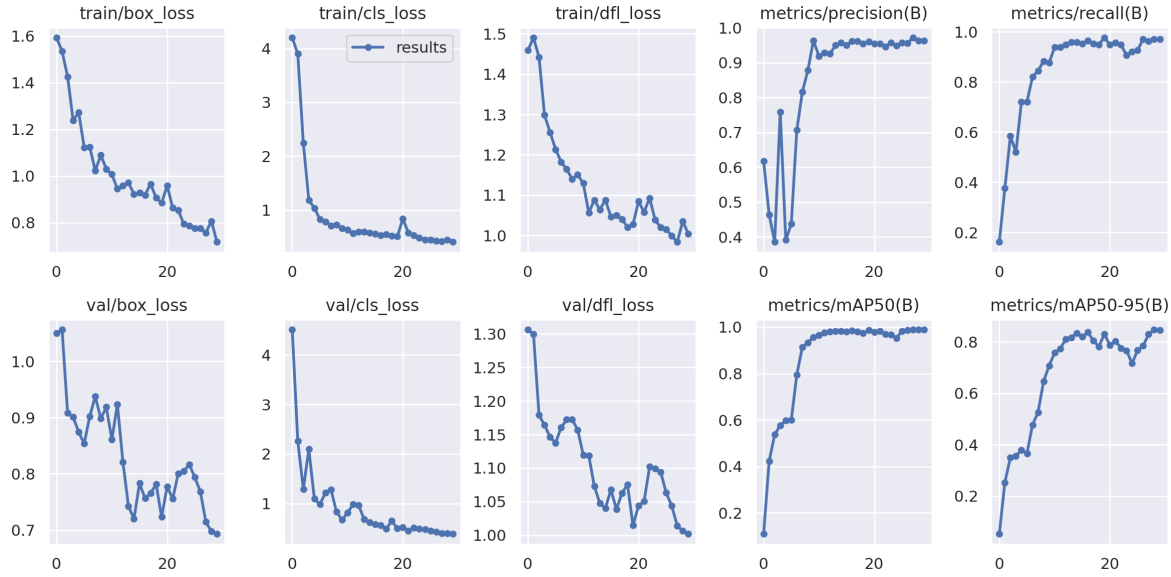

fig 6 Trains and val metrics YOLO

The training log for the YOLO model is displayed in the above image (fig 6).

R-CNN

R-CNN (Regions with Convolutional Neural Network) is a popular object detection system used in computer vision. The idea behind R-CNN is to use region proposals to identify potential object locations in an image, and then **apply a separate neural network to each proposal** to classify the objects and refine the bounding box coordinates.

One of the advantages of R-CNN is its high accuracy. By using region proposals, R-CNN can be more selective in its predictions, which improves its accuracy compared to other object detection systems. R-

CNN also has the ability to detect multiple objects within a single image, making it well suited for crowded scenes with multiple objects.

However, this model requires a large amount of data to train and usually, the training process takes much longer. This is mostly due to the need for multiple forward passes for a single input.

There are a few varients such as Fast R-CNN and Faster R-CNN which sacrifice a little bit of accuracy and precision for speed.

For this project, Faster R-CNN is used from the **detectron2** repo. A pre-trained model that was trained on the **COCO dataset** was loaded by the use of transfer learning it was fine-tuned to the new dataset. This allowed us to train the model faster compared to training from scratch.

Training was done using a custom loop in pytorch framework with **1500** itterations. The training took much longer compared to YOLO as expected.

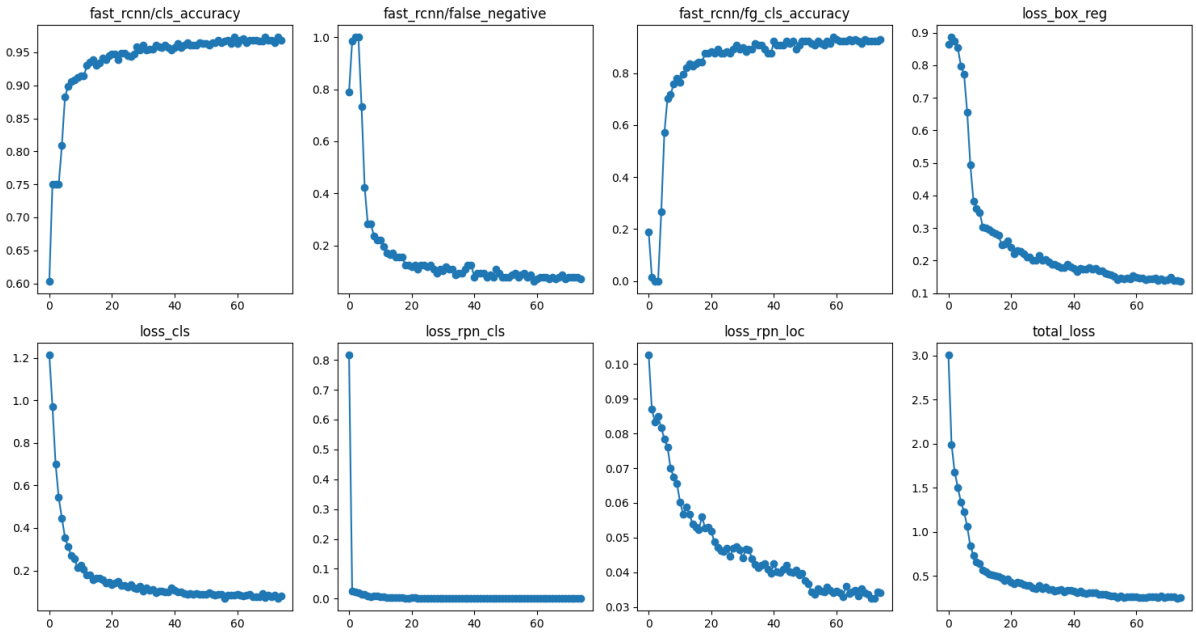

fig 7 Validation metrics R-CNN

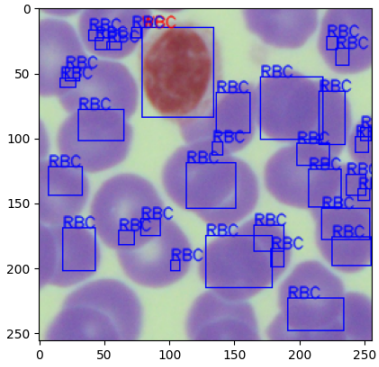

fig 8

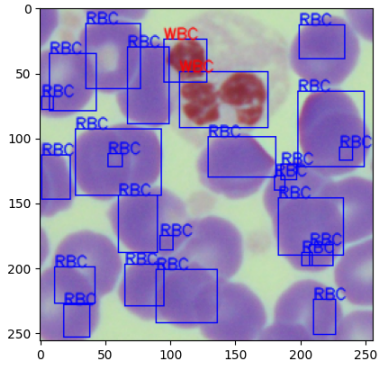

fig 9

As seen in the test images, the model doesn't perform as well as the yolo model. This is mostly because R-CNN models require more training data and takes much longer to train. With more data and better hardware this results can be improved a lot.

Conclusion

YOLO (You Only Look Once) and R-CNN (Regions with Convolutional Neural Networks) are both popular object detection algorithms, but they differ in terms of speed and accuracy. YOLO is fast and efficient, making it ideal for real-time object detection, but it may have lower accuracy compared to R-CNN. R-CNN, on the other hand, is known for its high accuracy, but it is slower due to its two-stage process. Both algorithms have their strengths and weaknesses, so the choice between YOLO and R-CNN will depend on the specific requirements of the project.