Research Associate Implementation Task

Fine-tuning YOLOv3 on a Given Dataset

Prepared by

Ali Akram

To

Dr Waqas Sultani

Overview

In this report, we used a provided dataset of malaria to train and fine-tune the YOLOv3 model for object detection. The goal was to train the model to detect specific objects and evaluate its performance.

Dataset

Data overview

The dataset consists of four folders having images in test, train, validation and annotations in the annotation folder in Pascal VOC format. There are four classes named Gametocyte, ring, schizont, and trophozoite which are different stages in the life cycle of Plasmodium responsible for causing malaria in humans. Data is split into 51 images of trains, 10 images as val data and 18 images as test data.

Data conversion

Since data is in Pascal VOC format we need to convert it into yolo format to start training the model. Pascal VOC format is an XML-based annotation format. Each image's annotations are stored in separate XML files, containing information about object bounding boxes, class labels, and image metadata. YOLO uses a different annotation format stored in a text file with each image. Each line in the text file represents specifying class label and bounding box coordinates.
We used Roboflow to convert the data from Pascal VOC to yolo format as it is convenient to convert data and use them directly in google colab.

Data pre-processing

For pre- processing we have applied Auto-Orient which discards EXIF rotations and standardise pixel ordering. When an image is captured, it contains metadata that dictates the orientation by which it should be displayed relative to how the pixels are arranged on disk stored in the EXIF orientation field. Auto-orient strips images of their EXIF data so that we see images displayed the same way they are stored on disk.

Data augmentation

As our dataset is very small, we applied data augmentation to increase the train data 3 times. As the more data we have the better the model learns to detect that object. For augmentation we flipped the images horizontally and vertically, we rotated the image by 90 degree clockwise and counterclockwise. We also applied Mosaic augmentation to solve the problem in our dataset.

Problems in our Dataset

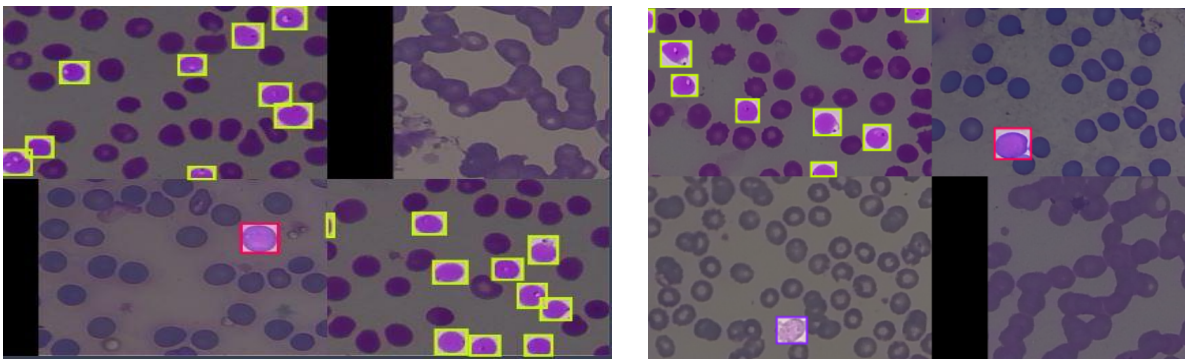
In addition to the dataset being very small, we have other problems in our dataset. We have 79 images and a total 387 annotations which comes to 4.9 per image. However in these total 387 annotations we have 281 annotations of ring which makes it as over represented in our data, trophozoite have 87 annotations, gametocytes have 15 annotations and schizont have only 4 annotations in our data. Another problem is that some images have only annotations of one class example; some images have only annotations of ring class, however it is really important for a model to perform better on all classes to have a dataset having annotations of more than one class in one image.

Solutions to Dataset problem

To have better results we should have 1500 or more images per class to make the model more accurate. The obvious solution is to gather more data for each class.

Second we need more information and expert advice on annotations. We can apply other techniques such as increasing/decreasing brightness, contrast, blurring, adding noise etc to further increase our dataset. As applying these augmentation without medical expert advice can lead to wrong detection.

Third, for the problem that some images have only annotations of one class we applied Mosaic augmentation which takes four images and combines them together into one, this makes an image having annotation of more than one class to make a model better in detecting these classes.



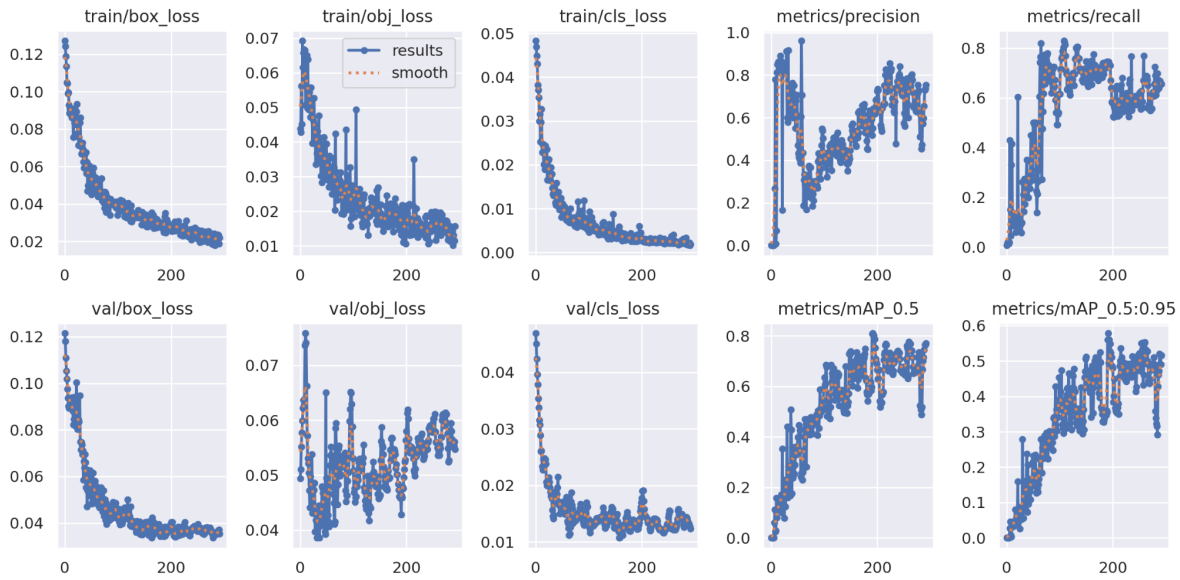
Mosaic augmentation

Procedure

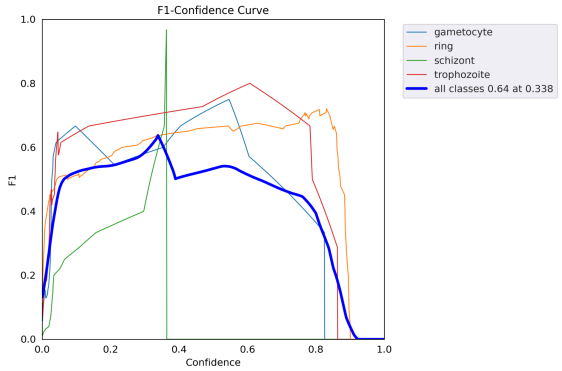
We downloaded yolov3 model and we will this pre-trained model to train on our custom dataset

Initial Training

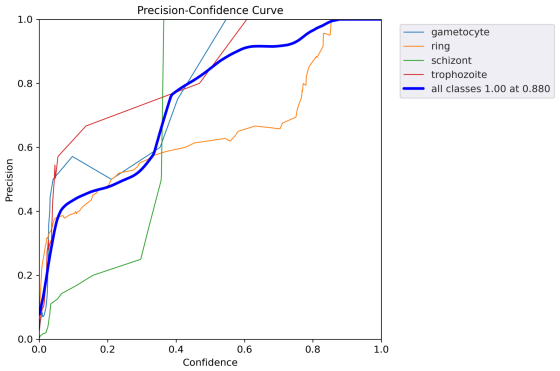
In our initial training phase, we employed the YOLOv3 model on a dataset consisting of 79 images. We trained the model for 250 epochs, using a learning rate of 0.01, an image size of 640, and a batch size of 16. Notably, no data augmentation or preprocessing techniques were applied during this stage. The training process completed in approximately one hour, and the results are below.



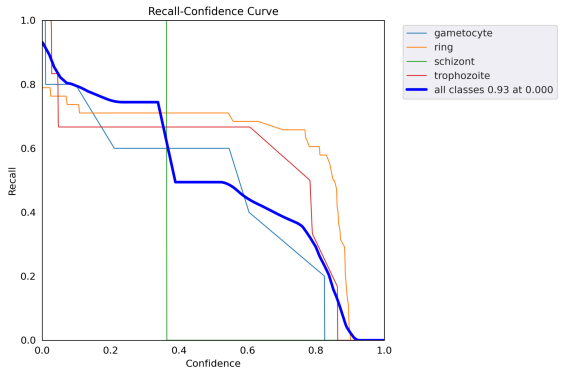
Loss curve



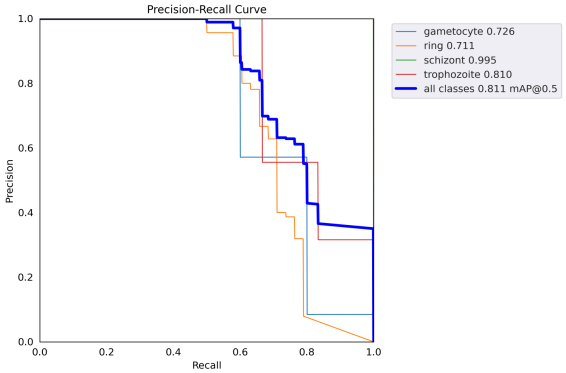
F1 curve



Precision curve



Recall curve



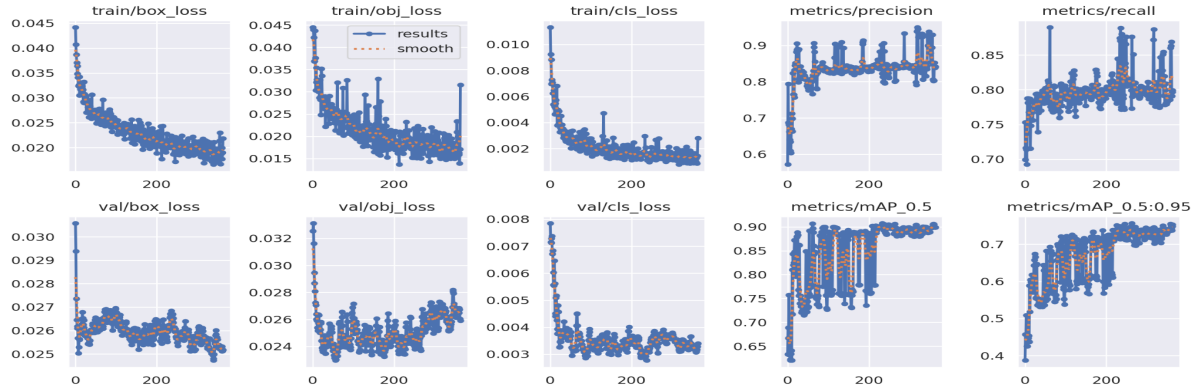
Precision-Recall curve

Model performance on validation dataset:

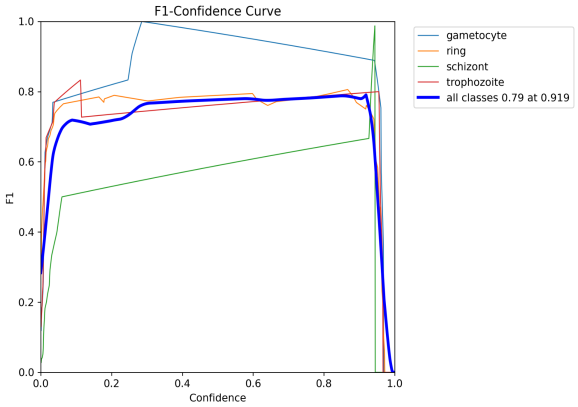
Class	Images	Instances	P	R	mAP50	mAP50-95:
all	10	50	0.682	0.721	0.648	0.41
gametocyte	10	5	0.788	0.753	0.678	0.508
ring	10	38	0.87	0.632	0.72	0.501
schizont	10	1	0.23	1	0.497	0.398
trophozoite	10	6	0.84	0.5	0.698	0.231

Intermediate Training

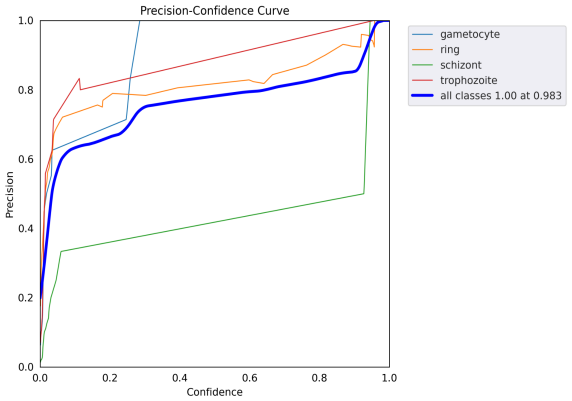
To enhance the quality and quantity of our dataset, we implemented data preprocessing and augmentation techniques, effectively expanding our dataset to 181 images. We continued the training process using the previously saved model, running for 350 epochs with a reduced learning rate of 0.001. During this phase, we increased the image size to 800 pixels and adjusted the batch size to 8. This extended training period took approximately 1.6 hours to complete. The results achieved during this stage are below.



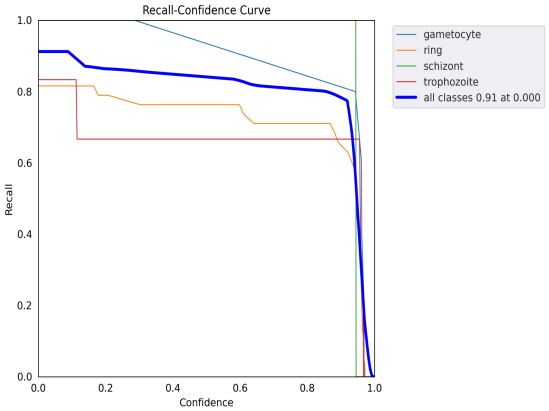
Loss curve



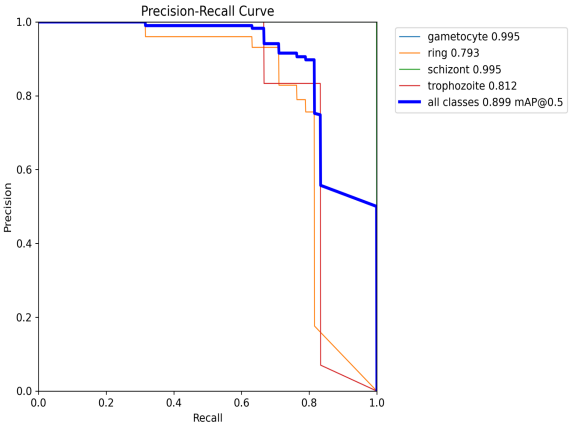
F1 curve



Precision curve



Recall curve



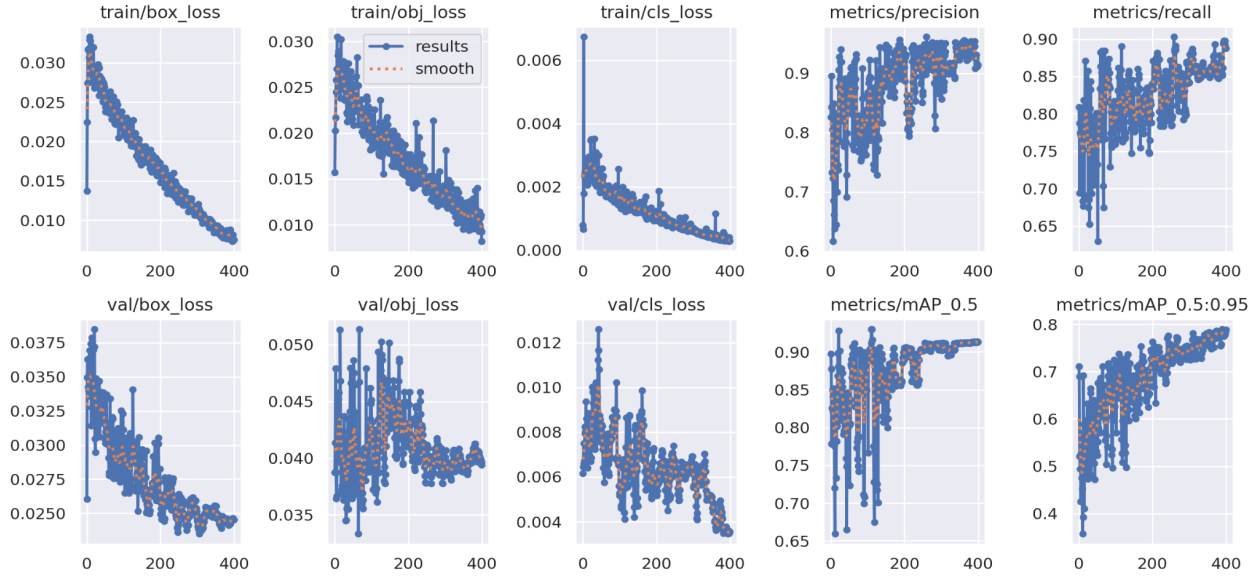
Precision-Recall curve

Model performance on validation dataset

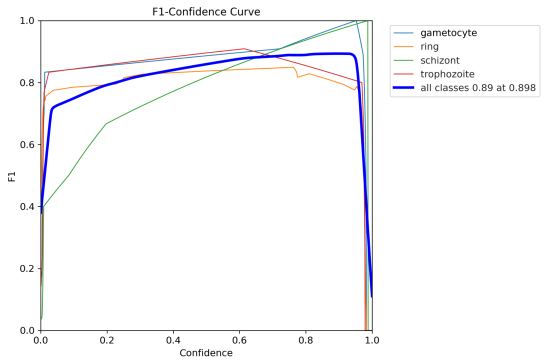
Class	Images	Instances	P	R	mAP50	mAP50-95:
all	10	50	0.791	0.833	0.897	0.739
gametocyte	10	5	1	0.956	0.995	0.815
ring	10	38	0.835	0.711	0.786	0.606
schizont	10	1	0.422	1	0.995	0.995
trophozoite	10	6	0.905	0.667	0.812	0.54

Final Training

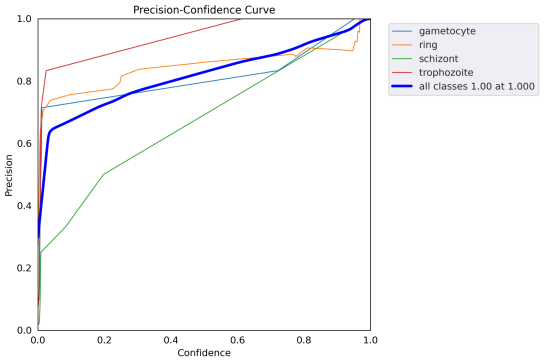
In the final training iteration, we utilised the last model and extended the training process to 400 epochs. We maintained a learning rate of 0.01 as our loss was almost the same in previous training, an image size of 800 pixels, and a batch size of 8. This training cycle, similar in duration to the initial one, lasted approximately one hour. The results obtained from this training are below.



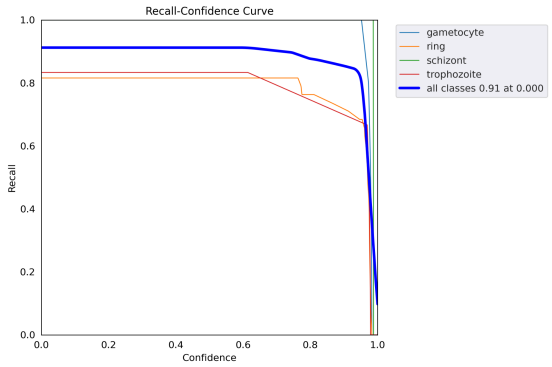
Loss curve



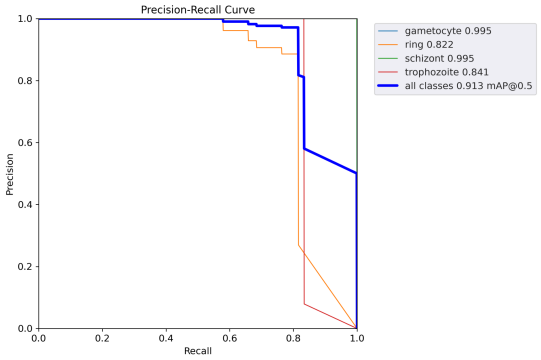
F1 curve



Precision curve



Recall curve



Precision-Recall curve

Model performance on validation dataset

Class	Images	Instances	P	R	mAP50	mAP50-95:
all	10	50	0.914	0.888	0.913	0.786
gametocyte	10	5	0.894	1	0.995	0.869
ring	10	38	0.907	0.773	0.82	0.656
schizont	10	1	0.855	1	0.995	0.995
trophozoite	10	6	1	0.778	0.842	0.624

Hyperparameters

Batch Size: Batch size determines the number of samples used in each forward and backward pass during training. During first training we used batch size of 16 which is commonly used, however we reduced our batch size after

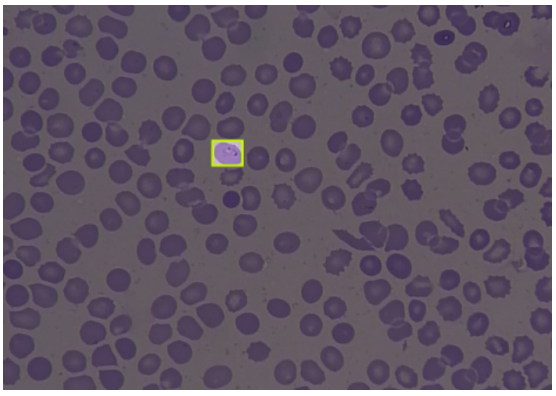
as we changed other parameters which resulted in ‘Cuda of memory’ error so we needed to reduce batch size at the expense of computation time.

Image size: YOLOv3 uses fixed-size input images. The image size can significantly impact both training speed and detection accuracy. We increased our image size from 640 to 800 as we are detecting very small objects in the image. We first used 1023 image size with batch size 16 but got an error on colab after a few epochs, so we reduced the image size to 800 and batch size 8 and trained the model on them.

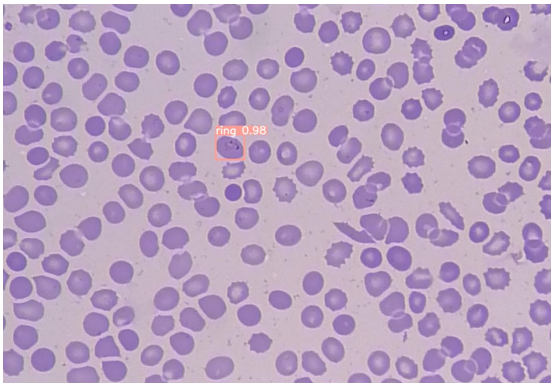
Learning rate: The learning rate controls how much the model's weights are updated during each training iteration. During first training we used lr 0.01, but after data augmentation in second training we reduced learning rate to 0.001, as we are using a pre-trained model so we used small learning rate on our augmented data. After 350 epochs we observed that model loss is not reducing much, so we increased our lr to 0.01 again and trained it on 400 epochs

Epochs: The number of training epochs specifies how many times the entire dataset is passed forward and backward through the neural network. We trained the model on a total of 1000 epochs with different parameters.

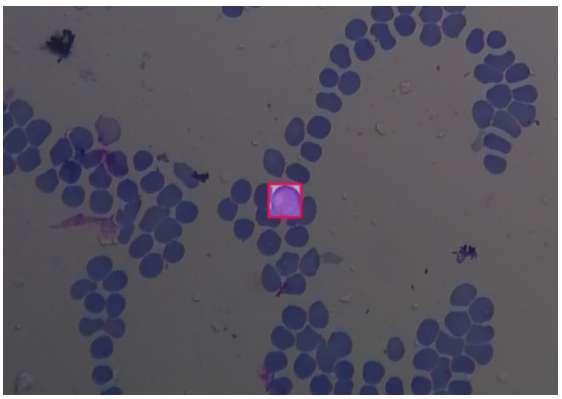
Results on test data



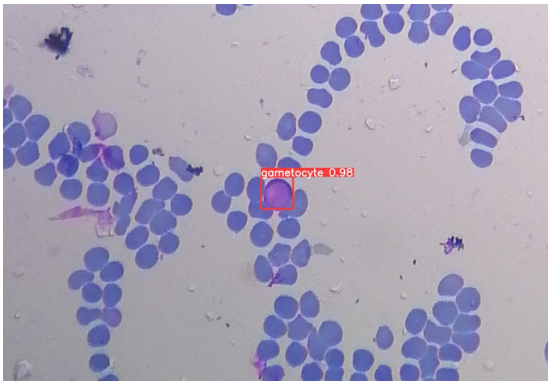
Ground truth



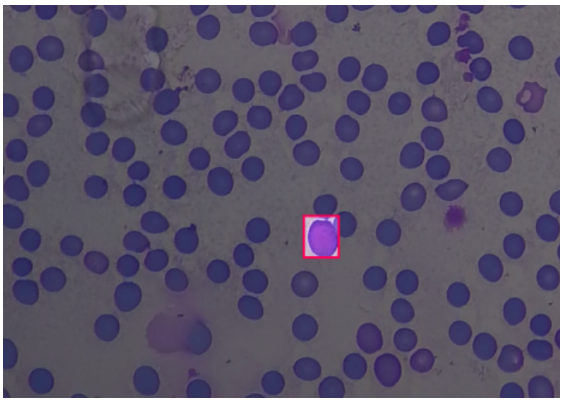
Detected



Ground truth

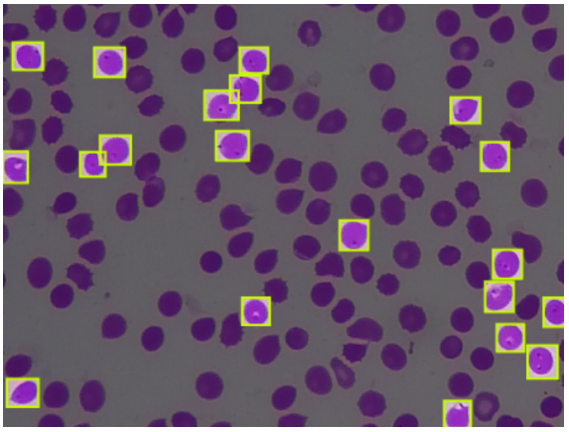


Detected

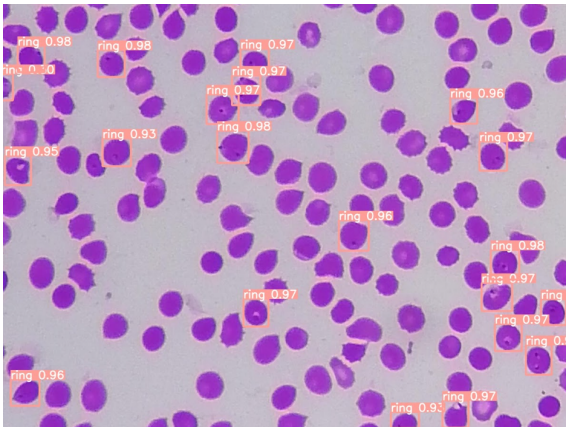


Ground truth

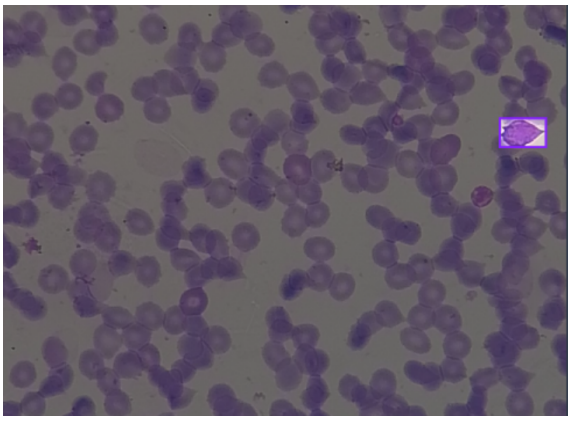
Detected



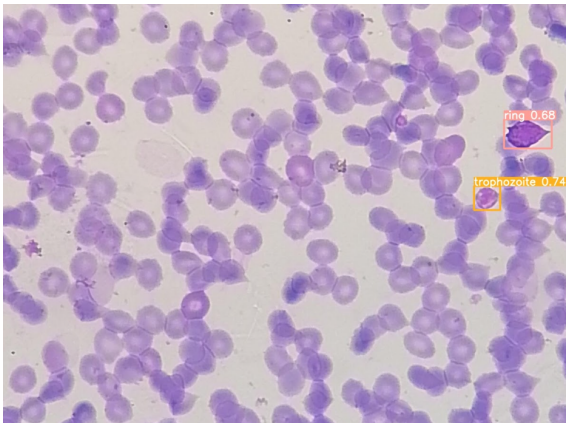
Ground truth



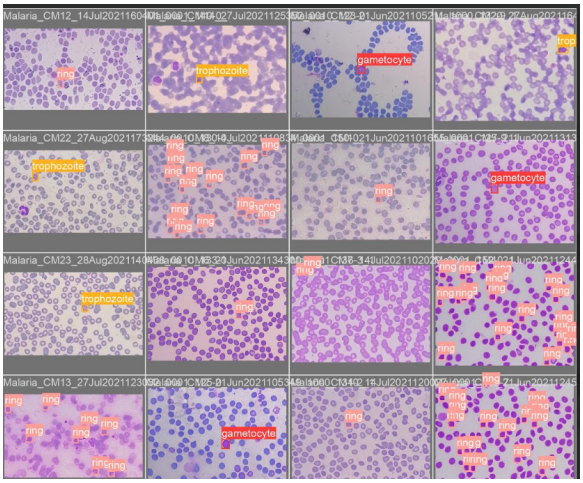
Detected



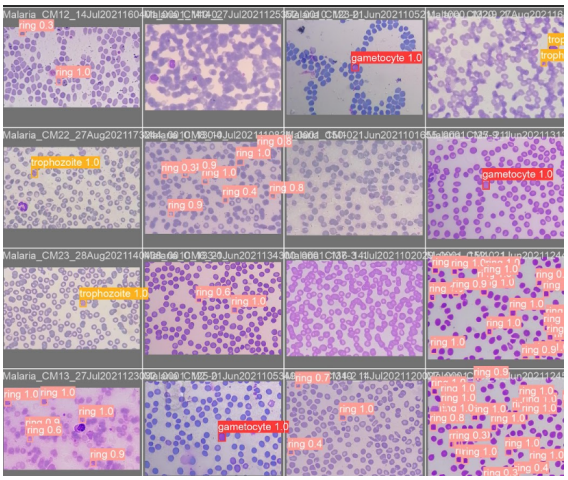
Ground truth



Detected



Ground truth



Detected

Class	Images	Instances	P	R	mAP50	mAP50-95:
all	18	87	0.97	0.727	0.839	0.708
gametocyte	18	3	0.962	1	0.995	0.897
ring	18	76	0.947	0.699	0.825	0.609
trophozoite	18	8	1	0.482	0.699	0.617

As we can see that model performs relatively better in detecting on such a small dataset. In the first 2 images it detected correctly. However, in the third image, there is an instance where the model has detected more instances of a ring than what is actually present in the ground truth annotations. In the fourth image we can see that the model detected trophozoite correctly but it also made the wrong detection of a ring which is not present in growth truth.Overall model performed well on test data.

Recommendations for further improvement.

Dataset Improvement

Our initial training dataset consisted of only 51train images. To improve model performance, we increased the number of training images to 153 by augmenting the data. However, the dataset is still relatively small at 153 images. The ideal performance requires at least 1500 images per class.

In analysing the class distribution, we found the "ring" class is overrepresented while the "schizont" class has only 4 examples. This imbalance means the model will be biased towards detecting "ring" and

struggle to recognize "schizont". A more equal class distribution is important to train an accurate multiclass detector.

Thus improvement in the dataset is expanding the dataset significantly to at least 1500 images per class, and balancing the class distribution through targeted data collection or augmentation. With sufficient training data across all categories, we can better optimize model parameters and achieve more robust performance. The model is limited by the small size and imbalance of the current dataset.

Model Improvement

We can use the newer YOLOv8 model instead of YOLOv3. YOLOv8 uses a CSPResNeXt-101 backbone which is far more powerful than the Darknet-53 backbone in YOLOv3. The ResNeXt architecture leverages split attention and cardinality to improve feature extraction. YOLOv8 has a larger model size of 250 million parameters versus only 75 million in YOLOv3. This makes YOLOV8 a better choice for our dataset.Moreover YOLOv8 supports higher input image resolutions like 1280x1280 which improve small object detection.

Model ensembling can improve our predictions. Model ensembling is combining predictions from multiple models to create a unified output. We can train 2 different models having different backbone and hyperparameters and then use ensembling to achieve better results.

Siamese networks

The objective is to correctly recognize 4 classes for malaria detection. We can use the Siamese network as they are widely used for image recognition. It has two neural networks having the same weights, compare two images and determine if they are similar or dissimilar. I have used these networks on fruit recognition and they perform really well in small datasets. We can look into this approach of two neural networks and compare image embeddings.

Parameter improvements

Fine-tuning other parameters like anchor boxes can help in better model performance. In our case we are detecting small objects so using small anchor boxes can help. Similarly we can fine-tune IoU which measures the overlap between the predicted bounding box and ground truth box for an object. A higher IoU threshold means we only consider predictions that have high overlap with the target box as positive matches.

Conclusion

In conclusion, we have trained a model on detecting 4 classes for malaria detection in humans using yolov3. We have fine-tuned the model, tried different parameters and have shared its results. Further we have seen the improvement we can do in the dataset and model for achieving state of the art results.