Malaria cell detection: comparison of methods

Camilo Sanabria
camiloandres.sanabriavanegas@studenti.unipd.it

Flavio Agostini
Flavio.agostini.1@studenti.unipd.it

Fabio Pimentel
fabiomanuellenin.pimentelcaminero@studenti.unipd.it

Abstract

Malaria is a disease caused by Plasmodium parasites and remains one of the main global health concerns. The state-of-the-art approach for parasite detection and stage determination, to this day, remains manual inspection of microscopy data by trained miscroscopists, due to the current unreliability of automated systems. From a computational perspective, the problem is defined as object detection. The development of adequate computational tools to facilitate, or completely automate, this task would be highly beneficial. In this study, we assess how different deep learning approaches perform on this task and try to determine if state-of-the-art AI tools can effectively solve this problem. For this purpose, we selected four methodologies and tested their performance on a manually annotated dataset of infected and uninfected blood cells: U-Net, a convolutional network for image segmentation; a RCNN based on resnnet50; a RT-DETR object detection transformer; YOLO, a CNN based real time object detector.

1. Introduction

Starting from the pioneering work of Antonie van Leeuwenhoek, who first observed bacteria by microscopy, optical microscopy has become an indispensable tool for biological research [12]. Malaria is a life-threatening disease that is spread to humans by some types of mosquito. It is found mainly in tropical countries and is both preventable and curable. According to the last Global Malaria Report in 2023, there were estimated 263 million malaria cases and 597,000 malaria deaths [21]. So far, the main diagnostic method relies on manual inspection of blood smear samples by trained microscopists. This approach is slow and prone to human error. Therefore, it is imperative to develop accurate and reliable automatic diagnostic systems, and in this sense AI is gaining traction in the scientific community [10]. To explore the potentialities of AI tools, we applied four different methods to a publicly available dataset of malaria-infected cell images, the BBBC041v1 image set available from the Broad Bioimage Benchmark Collection [9].

2. Related work

2.1. Traditional methods

Before the advent of AI cell imaging tasks such as cell identification and counting have been tackled with segmentation techniques. The most widely used in the context of cell imaging is watershed segmentation. This technique is able to delineate cell boundaries based on intensity gradients [20] [11]. The algorithm treats the image's intensity as a topographic surface, identifying cells as ”basins” separated by ridges. This method often struggles with over-segmentation in noisy images and under-segmentation in cases where intensity differences between adjacent cells are low.

2.2. Deep Leaning approaches

In the past years different studies have explored a variety of deep learning architectures to detect and classify malaria-infected cells. A key reference for our work is [5]. Here, the researches evaluate the performance of an R-CNN network implemented with Keras. They achieved a mean average precision of 78%, evaluated uniquely on the model ability to locate cells, and compared the performance with CellProfiller, a traditional segmentation tool [19]. They highlighted the importance of dataset quality and generalization capabilities and argued that treating cell identification as an object detection problem, rather than a segmentation one, provides several advantages, especially in the annotation process, where drawing bounding boxes proves to be several times faster than providing pixel-level annotations. We used a similar approach with our R-CNN, but opted to implement it using pytorch.

U-net is a CNN architecture, first introduced in [17], developed specifically for image segmentation with biomedical data. While we haven't found direct applications of this to malaria datasets, we were curious to compare its performance to its competitors, who do not perform segmentation.

In [14], a number of Deep Learning architectures (VGG16, VGG19, XCeption among many) have been tested against our same dataset. Differently from us, they didn't perform object detection, extracting images representing the single cells in preprocessing, and focused their efforts in the classification task, obtaining excelent results: 96.85 F1-score with a transfer learning approach (DCNNs trained on ImageNet dataset). Avoiding object detection makes the task easier, and avoids the noise induced by annotation and image quality (eg, unannotated cells). Other similar studies, [6] [8] use datasets where training images are already segmented. These approaches are not easily applicable as they do not take into account the possibility of overlapping cells and differences in staining techniques. To overcome this, more sophisticated methods were developed and tested on malaria datasets, such as faster R-CNN [4] [18] and YOLO algorithms [1], with YOLO algorithms appearing to perform better [16]. These last methods allow end-to-end object detection pipelines, and are thus better suited to face real-world applications.

2.3. Transformers applications

In [3], Guemas et al. evaluated RT-DETR for Plasmodium species detection in thin blood smear images using a dataset of 24,720 images. Differently form us, their study focused on performance on individual patient data, achieving an overall accuracy of 79.4%. They also compared RT-DETR with YOLOv8, finding that YOLO performed slightly better but at a higher computational cost. Similarly from us, they also evaluated the performance of YOLOv8 to compare the results. The comparison indicated that YOLO performed slightly better, at the cost of being more computationally expensive.

3. Dataset

The dataset [9] comprises 3 sets of images, for a total of 1364 images, gathered by three different researches, located in different countries: from Brazil (Stefanie Lopes), from Southeast Asia (Benoit Malleret), and from in vitro samples cultivated at the University of Oxford (Gabriel Rangel). The researchers have also manually curated the annotation of each image, indicating bounding boxes and class labels for each cell. In total, the dataset contains 86.035 individual cells, belonging to two classes of uninfected cells (RBCs and leukocytes) and four classes of infected cells at different stages of development (gametocytes, rings, trophozoites, and schizonts). The dataset is split in training and test set (Table 1). As we can see, it is heavily unbalanced, with uninfected cells constituting about 97% of all cells. The annotators were allowed to mark some cells as "difficult" if it was impossible to give a precise label to specific cells. Cells marked as "difficult" are, in any case, considered as "infected" for the purposes of our analysis. To this regard, since the dataset is already extremely unbalanced, we decided to limit our analysis to "infected" vs. "uninfected" cells, performing a pre-processing step to adjust the annotations to this new scheme. This choice is common in similar studies (eg, [6] [14]), while other studies attempt to classify the various stages of parasitic development as well ([3]). Images collected by different researches and instruments have different characteristics of illumination, exposure, color schemes and contrast. Taking inspiration from [5], we decided to address this problem with two pre-processing steps: firstly, we rescaled the intensities of each channel of an image between its maximum and minimum values. This helped mitigate the variation introduced by different lighting and exposure between pictures. We then applied CLAHE (Contrast Limited Adaptive Histogram Equalization) [13] to enhance local contrast in the images, making features more prominent. Finally, images also presented different resolutions (1600x1200, 1944x1383). We addressed this problem, when necessary, differently in each model.

Set	Images	Uninfected	Infected	Inf/tot
Train	1208	77523	2590	0,032
Test	120	5922	308	0,052
Total	1328	83445	2898	0,033

Table 1: Dataset composition

4. Methods

4.1. U-Net Segmentation

The segmentation model implemented is based on the U-Net architecture with a ResNet-50 encoder pre-trained on ImageNet.

4.1.1 Dataset and Preprocessing

The dataset used for training consists of paired images and masks, where each mask encodes class information using pixel values. To ensure consistency, images and masks were resized and aligned according to the bounding box annotations provided in the dataset. Any size mismatches were resolved using OpenCV's nearest-neighbor interpolation to preserve data integrity. To improve generalization, various data augmentations were applied, including image and mask resizing, horizontal and vertical flips, normalization, and adjustments in brightness and contrast.

4.1.2 Model Architecture and Loss Function

The chosen U-Net model incorporates a ResNet-50 encoder due to its strong feature extraction capabilities and widespread application in biomedical image segmentation [17]. U-Net follows an encoder-decoder structure: the encoder extracts feature representations, while the decoder reconstructs the segmentation map.

ResNet-50, serving as the encoder, consists of 50 layers with residual connections, which facilitate deep network training by mitigating the vanishing gradient problem.

For segmentation, a combined loss function was used to address class imbalance and improve accuracy. Weighted Cross-Entropy Loss was employed to handle the imbalanced class distribution, while Dice Loss was used to enhance the overlap between predicted and ground-truth masks.

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{Dice} \tag{1}$$

4.1.3 Implementation

The model was implemented using PyTorch library and was optimized for three classes (background, uninfected, and infected cells). A custom loss function was implemented with class weights [0.1, 1, 0.7], adjusting for dataset imbalance. We used Adam optimizer with a lr=0.001 and StepLR scheduler (decay factor 0.1 every 5 epochs), and training was conducted for 10 epochs. Epoch loss and execution time were logged for monitoring.

4.2. Faster R-CNN

Another approach for classifying infected and uninfected cells involved using a Faster R-CNN model. This method enables both the localization of individual cells within images through bounding boxes and their classification.

4.2.1 Dataset and Preprocessing

The annotations give the bounding-box's class and localization. A custom class was implemented to load the images and extract the corresponding bounding-box coordinates and class labels using the Pascal VOC format. Labels are then mapped to numerical values. To handle class imbalance, a set of augmentations was implemented including horizontal and vertical flips, brightness and contrast modifications, and random resized cropping. Augmentations were applied to images and boxes to ensure consistency during training.

4.2.2 Model architecture and loss

The detection model is based on a Faster R-CNN with a ResNet 50 feature Pyramid Network (FPN) as the backbone, described in detail in [15]. The model consists of three main components: the backbone, that allows extraction of the features form the input image; the RPN (Region Proposal Network) that identifies the regions that are likely to contain objects; the ROI Head, a component that allows the classification and refinement of the bounding boxes. The use of the FPN allows to generate a multi-scale feature hierarchy, improving detection across large and small objects. Moreover, the RPN helps to reduce computation times by generating bounding-box candidates, without using a separate region proposal stage. While in the ROI the proposals are classified and refined by updating the box coordinates.

$$\mathcal{L}_{\text{RPN}} = \mathcal{L}_{\text{cls}} + \lambda \mathcal{L}_{\text{bbox}} \tag{2}$$

$$\mathcal{L}_{\text{ROI}} = \mathcal{L}_{CE} + \gamma \mathcal{L}_{\text{bbox}} \tag{3}$$

$\mathcal{L}_{CE}$: Cross-entropy loss for classification, $\mathcal{L}_{\text{bbox}}$: Smooth L1 loss for bounding box adjustment.$\mathcal{L}_{\text{cls}}$: Binary cross-entropy loss for object/no-object classification, $\mathcal{L}_{\text{bbox}}$: Smooth L1 loss for bounding box regression.

The final classification layer was modified to output the three possible classes, and the default predictor is changed with a custom one that allows to apply Weighted Cross-Entropy Loss to address class imbalance.

4.2.3 Implementation

We implemented our model using PyTorch libraries. We fine-tuned a Faster R-CNN model using a ResNet-50 FPN backbone pretrained on COCO, imported from torchvision's detection models. The classification head was replaced to match our number of classes, and its cross-entropy loss function was modified to incorporate class weights ([0.1, 1.0, 7.0]) to address class imbalance. For optimization, we implemented AdamW optimizer with a lr=0.0001 and StepLR scheduler reducing LR by 0.1 every 5 epochs. Training was carried out for 6 epochs with training batches of size 8. Epoch loss, duration, and learning rate were logged for monitoring.

4.3. RT-DETR Object Detection Transformer

We implemented the Real-Time DEtection TRansformer (RT-DETR) [22], a Transformer-based end-to-end object detector designed for real-time performance. Unlike previous CNN-based detectors, RT-DETR processes images as a sequence of features, enabling direct object detection.

4.3.1 Dataset and Preprocessing

Our dataset annotations were converted into COCO format to ensure compatibility with RT-DETR. Input images were resized to 640×640, consistent with the model's pretraining setup [22]. As before, data augmentation techniques were applied to mitigate class imbalance and improve model generalization. Image preprocessing was performed using the AutoImageProcessor, which normalizes pixel values and adjusts bounding box formats to match the model's requirements. A custom collate function was used to make data compatible with PyTorch.

4.3.2 Model architecture and loss

RT-DETR consists of three main components. Backbone: We use ResNet-50vd, a modified ResNet-50 variant optimized for enhanced feature extraction. This backbone processes input images and generates multi-scale feature representations. Transformer Encoder-Decoder: The hybrid encoder decouples intra-scale feature interaction from cross-scale fusion, reducing computational overhead while maintaining accuracy. The decoder refines object queries using uncertainty-minimal query selection, ensuring robust detection by prioritizing high-confidence predictions. Prediction Heads: Fully connected (FC) layers predict object categories and bounding boxes directly from the refined queries. The loss function consists of three components. The standard cross-entropy loss is used for classification, while bounding box regression relies on an L1 loss combined with the generalized IoU loss to ensure accurate alignment between predicted and ground-truth shapes. Additionally, RT-DETR optimizes query selection by explicitly minimizing uncertainty in both classification and localization, improving the quality of detected objects.

$$\mathcal{L}_{\text{DETR}} = \lambda_{\text{cls}}\mathcal{L}_{\text{cls}} + \lambda_{\text{box}}\mathcal{L}_{\text{box}} + \lambda_{\text{giou}}\mathcal{L}_{\text{giou}} \tag{4}$$

4.3.3 Implementation

We fine-tuned RT-DETR-R50 using the AutoModelForObjectDetection Hugging Face Transformers framework. The model was initialized from the pre-trained PekingU/rtdetr_r50vd_coco_o365 checkpoint. This model It is pretrained on COCO and Objects365 datasets. The training setup included: lr=5e-5 with gradient clipping (max_grad_norm=0.1), 300 warmup steps and batch size of 8. During training mean average precision (mAP) was tracked as the primary metric, using epoch-based evaluation and checkpointing. The best-performing model was automatically restored at the end of training. Training was conducted for 12 epochs, using Trainer API, with automated evaluation and metric computation.

4.4. YOLO

Lastly, we implemented YOLOv8 [7], a real-time object detection model that efficiently predicts both bounding boxes and object categories in a single forward pass. Its streamlined architecture enables high-speed inference while maintaining strong detection accuracy.

4.4.1 Dataset and Preprocessing

Unlike the previous methods, our implementation of YOLO through Ultralytics required annotations in a specific format given by the normalization of the bounding boxes based on image dimensions. Due to training time, and consistently to what we did with RT-DETR, the images are resized to 640x640. This step is implemented directly in the model's train function. Likewise, Augmentations were performed using YOLO´s built-in functions.

4.4.2 Model architecture and loss

We used the model YOLOv8m, a medium size version of YOLOv8 due to its balance between detection and inference speed. The model consists of three main components, the backbone that extracts the multi-scale feature map from the input. The Neck that executes the process of feature aggregation that helps to detect objects of various sizes by fusing low-level and high-level features. Finally, the head, that predicts bounding boxes, objectness scores, and class probabilities. The loss function is defined by the sum of losses related with the bounding-box coordinates, the classes error for detected objects, and the objectness.

$$\mathcal{L}_{YOLO} = \lambda_{\text{box}}\mathcal{L}_{\text{box}} + \lambda_{\text{cls}}\mathcal{L}_{\text{cls}} + \lambda_{\text{obj}}\mathcal{L}_{\text{obj}} \tag{5}$$

4.4.3 Implementation

We fine-tuned a YOLOv8-m model for real-time object detection using a pretrained (on COCO dataset) checkpoint, importing it from ultralytics library. The model was trained for 20 epochs with a batch size of 16. Other training parameters are: Box loss=7.5; class loss=3.0; Objectness loss=1.0, defined after some trials to improve performance on bounding boxes and classifications.

5. Experiments

5.1. Evaluation metrics

To assess the models' performance in object detection, we used the Intersection over Union (IoU) metric, which measures how well the predicted bounding boxes align with the ground truth using a threshold of 0.5 as presented in various references. For classification, our main focus is to evaluate the model's ability to distinguish between infected and uninfected cells separately. Due to class imbalance, relying solely on overall accuracy could introduce bias, misrepresenting the model's performance. For this reason, we instead focused on precision, recall and F1 score. Since the F1 score is typically not used for segmentation tasks, we compared the Dice score instead for the U-Net method, as it provides a more suitable evaluation metric for segmentation performance.

5.2. Training experiments

Results of training trials can be seen on the separate appendix.pdf file.

5.3. Computational resources

In order to perform this experiments we used Google Colab Pro's resources. In particular, we used T4 GPU, coming along with 12.7 GB of system RAM, 235.7 GB of disk storage and 15.0 GB of GPU RAM.

5.4. Training time and inference speed

Table 2 presents the results for training time, inference speed, and model size. Faster R-CNN (F-RCNN), trained for 7 epochs, took 45.96 minutes, reflecting a higher computational cost due to its region proposal mechanism. RT-DETR, trained for 12 epochs, required 55.50 minutes due to its transformer-based attention. YOLOv8, trained for 20 epochs, completed training in 37.56 minutes, offering a good balance of efficiency and complexity. U-Net, trained for 10 epochs, had the shortest training time at 22.71 minutes, making it the most computationally efficient.

Inference speed is key for real-world deployment, where fast processing is essential. YOLOv8 showed the best performance with a processing time of 14.6 ms. While RT-DETR is more efficient than Faster R-CNN, it still lags behind YOLO, contrary to the claims in [3], where RT-DETR was highlighted for its speed. Faster R-CNN, though accurate, has the slowest inference time, limiting its real-time applicability. U-Net has moderate speed but isn't ideal for object detection, reducing its effectiveness in practical use.

Metric	U-Net	F-RCNN	RT-DETR	YOLOv8
Epochs	10	7	12	20
Training time (min)	22.71	45.96	55.50	37.56
Inference speed (ms)	49.2	127.7	33.1	14.6
Model size (Millions of parameters)	32.5	41.3	31.1	25.9

Table 2: Training time and inference speed

5.5. Results

The main results of our experiments can be viewed in Table 3. YOLOv8 shows the highest precision and recall for uninfected cells, indicating its ability to identify almost all uninfected cells while making few false positive predictions. On the other hand, RT-DETR presents the highest precision for infected cells, while Faster R-CNN provides the highest recall for this category at the trade-off of increasing the number of false positives.

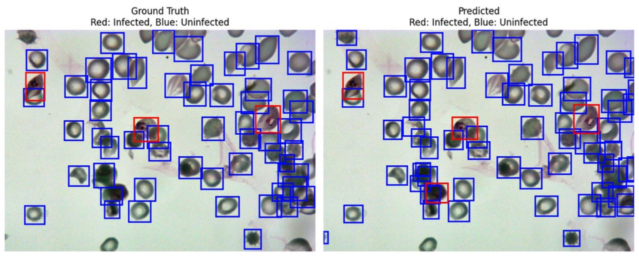
Metric	Class	U-Net	F-RCNN	RT-DETR	YOLOv8
Precision	Infected	/	0.4887	**0.7542**	0.751
	Uninfected	/	0.8015	0.7732	**0.867**
Recall	Infected	/	**0.7045**	0.5877	0.68
	Uninfected	/	0.9672	0.7512	**0.994**
F1/Dice	Infected	0.5341	0.5770	0.6606	**0.7137**
	Uninfected	0.9097	0.8765	0.7620	**0.9261**
IoU	Infected	0.3772	0.8553	0.6841	**0.9143**
	Uninfected	0.8352	0.8534	0.6799	**0.9141**
Accuracy	Infected	0.4129	**0.7045**	0.5877	0.4059
	Uninfected	0.9215	**0.9672**	0.7512	0.7257

Table 3: Comparison of evaluation metrics

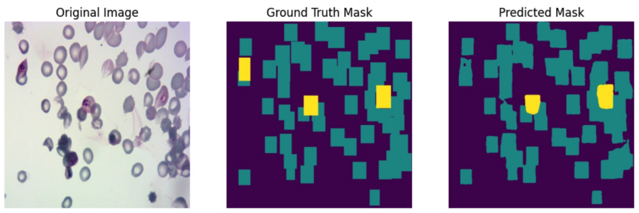
YOLOv8 also delivers the best results in terms of F1-score and IoU, indicating its strong balance between precision and recall for classification and its ability to accurately fit the spatial distribution of the predicted bounding boxes. This performance can be attributed to its Neck architecture, which facilitates the aggregation of extracted features for detecting objects of multiple sizes. Additionally, its end-to-end approach helps optimize bounding box placement over the test images, enhancing localization accuracy. From Figure 1(a), we can see how YOLO is also able to accurately predict cells that were not annotated in the ground truth, further solidifying its position as a robust detector capable of identifying overlooked instances and enhancing overall detection performance.

The best model in terms of accuracy is Faster R-CNN, which can be explained by its two-stage process. This architecture first determines region proposals and then performs classification, allowing for more precise predictions of the classes. The region proposal network (RPN) helps eliminate irrelevant regions, leading to more refined object localization and classification. In Figure 1 (c) we can see and example of failure case, where infected cells are correctly identified, but also reported an uninfected as well. This misclassification may stem from the region proposal mechanism of R-CNN. If the extracted features of an uninfected cell resemble those of infected cells, due to visual similarities, overlapping features, or suboptimal region proposals, the classifier may assign an incorrect label. Additionally, R-CNN processes each proposed region independently, which can lead to inconsistencies in detection, especially in cases where subtle visual cues differentiate infected from uninfected cells.

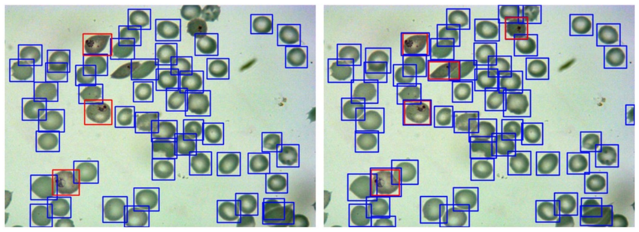
RT-DETR demonstrates high precision but exhibits lower performance in terms of recall and IoU. This suggests that while it makes confident predictions, it struggles to detect all infected cells. This could be attributed to its transformer-based spatial attention mechanism, which may not perform well in the presence of overlapping images or clusters of objects, leading to missed detections. In Figure 1 (d) it is possible to see how the model struggles with


Ground Truth
Red: Infected, Blue: Uninfected
Predicted
Red: Infected, Blue: Uninfected

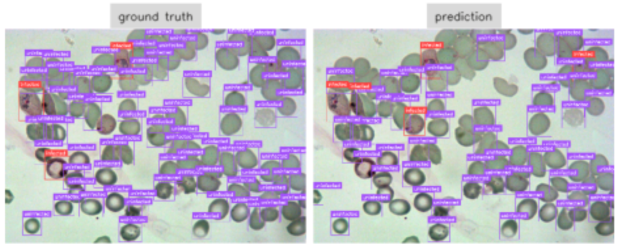
(a) Ground Truth vs YOLOv8 predictions



(b) Ground Truth vs U-Net predictions



(c) Ground Truth vs F-RCNN predictions



(d) Ground Truth vs RT-DETR predictions

Figure 1: Comparison of predictions from different models.

these types of situations. This is a known problem with transformer-based object recognition, in [2] it is indicated that this limitation stems from the fixed query set size of the transformer decoder, which restricts the model's inference capacity.

Finally, the U-Net architecture performed well in segmenting uninfected cells but struggled with infected ones. This aligns with findings from the literature [5], where semantic segmentation-based methods are generally not preferred for object detection tasks. The lack of a dedicated object detection mechanism makes U-Net less suitable for bounding box placement.

Our results align with the previously cited studies [1] and [16], which show that the YOLO method tends to achieve superior performance in both the spatial localization of cells and the F1 metric. However, unlike these studies, the accuracy obtained in our experiments is lower, whereas their results reported accuracy exceeding 90%. This discrepancy can be attributed to differences in computational resources available, as they were able to train their models for over 1000 epochs.

Although some other works suggest that R-CNN models can be the best option due to their high accuracy in predictions, such as in [4] and [18], where classification accuracy reached nearly 99%, it is crucial to take into account the number of parameters and epochs used. In [18], for example, the model consisted of over 95 million parameters and was trained for more than 1000 epochs, which significantly contributed to its high accuracy.

Additionally, our results align with the findings of [3], which compared RT-DERT and YOLO architectures. In that study, YOLOv8 also outperformed the RT-DERT in overall performance, even if the researches maintaned their position on the superiority of RT-DERT, mainly claiming that it is easier to deploy in real-time on low-cost devices.

6. Conclusion

This study compared the performance of U-Net, Faster R-CNN, RT-DETR, and YOLOv8 for the classification and detection of infected and uninfected blood cells. The results demonstrated that, for our experiment and settings, YOLOv8 is the most practical option, offering a good precision-recall trade-off while maintaining high computational efficiency. Faster R-CNN is the best choice for classification accuracy, but its computational demands are limiting, and it suffers from a high tendency of false positives for infected cells. RT-DETR shows promise with its high precision but requires further optimization to achieve better accuracy, while U-Net is the least suitable for detection tasks.Additionally, our study provides a reference starting point for the comparison of these methods, offering a structured evaluation with replicable parameters and training settings.

For future work, it would be interesting to implement hybrid approaches, for example, combining the localization advantage of YOLO with the classification head of Faster R-CNN, potentially leading to more robust detections.

References

[1] Fetulhak Abdurahman, Kinde Anlay Fante, and Mohammed Aliy. Malaria parasite detection in thick blood smear microscopic images using modified yolov3 and yolov4 models. *BMC bioinformatics*, 22:1–17, 2021.

[2] Hyeong Kyu Choi, Chong Keun Paik, Hyun Woo Ko, Min-Chul Park, and Hyunwoo J Kim. Recurrent detr: Transformer-based object detection for crowded scenes. *IEEE Access*, 2023.

[3] Emilie Guemas, Baptiste Routier, Théo Ghelfenstein-Ferreira, Camille Cordier, Sophie Hartuis, Bénédicte Marion, Sébastien Bertout, Emmanuelle Varlet-Marie, Damien Costa, and Grégoire Pasquier. Automatic patient-level recognition of four Plasmodium species on thin blood smear by a real-time detection transformer (RT-DETR) object detection algorithm: a proof-of-concept and evaluation. *Microbiology Spectrum*, 12(2), 1 2024.

[4] Jane Hung and Anne Carpenter. Applying faster r-cnn for object detection on malaria images. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 56–61, 2017.

[5] Jane Hung, Allen Goodman, Deepali Ravel, Stefanie CP Lopes, Gabriel W Rangel, Odailton A Nery, Benoit Malleret, Francois Nosten, Marcus VG Lacerda, Marcelo U Ferreira, et al. Keras r-cnn: library for cell detection in biological images using deep neural networks. *BMC bioinformatics*, 21:1–7, 2020.

[6] Tayyaba Jameela, Kavitha Athota, Ninni Singh, Vinit Kumar Gunjan, and Sayan Kahali. Deep learning and transfer learning for malaria detection. *Computational intelligence and neuroscience*, 2022(1):2221728, 2022.

[7] Glenn Jocher, Jing Qiu, and Ayush Chaurasia. Ultralytics YOLO, Jan. 2023.

[8] Saddam Hussain Khan, Najmus Saher Shah, Rabia Nuzhat, Abdul Majid, Hani Alquhayz, and Asifullah Khan. Malaria parasite classification framework using a novel channel squeezed and boosted cnn. *Microscopy*, 71(5):271–282, 2022.

[9] Vebjorn Ljosa, Katherine L Sokolnicki, and Anne E Carpenter. Annotated high-throughput microscopy image sets for validation. *Nature methods*, 9(7):637–637, 2012.

[10] Golla Madhu, Ali Wagdy Mohamed, Sandeep Kautish, Mohd Asif Shah, and Irfan Ali. Intelligent diagnostic model for malaria parasite detection and classification using imperative inception-based capsule neural networks. *Scientific Reports*, 13(1):13377, 2023.

[11] Norberto Malpica, Carlos Ortiz De Solórzano, Juan José Vaquero, Andrés Santos, Isabel Vallcorba, José Miguel García-Sagredo, and Francisco Del Pozo. Applying watershed algorithms to the segmentation of clustered nuclei. *Cytometry: The Journal of the International Society for Analytical Cytology*, 28(4):289–297, 1997.

[12] Thomas A. Nketia, Heba Sailem, Gustavo Rohde, Raghu Machiraju, and Jens Rittscher. Analysis of live cell images: Methods, tools and opportunities. *Methods*, 115:65–79, 2017. Image Processing for Biologists.

[13] Stephen M. Pizer, E. Philip Amburn, John D. Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B. Zimmerman, and Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39(3):355–368, 1987.

[14] Aimon Rahman, Hasib Zunair, Tamanna Rahman Reme, M. Sohel Rahman, and M.R.C. Mahdy. A comparative analysis of deep learning architectures on high variation malaria parasite classification dataset. *Tissue and Cell*, 69:101473, 2021.

[15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[16] Maura Rocha, Maíla Claro, Laurindo Neto, Kelson Aires, Vinicius Machado, and Rodrigo Veras. Malaria parasites detection and identification using object detectors based on deep neural networks: a wide comparative analysis. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 11(3):351–368, 2023.

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[18] Wojciech Siłka, Michał Wieczorek, Jakub Siłka, and Marcin Woźniak. Malaria detection using advanced deep learning architecture. *Sensors*, 23(3):1501, 2023.

[19] David R Stirling, Madison J Swain-Bowden, Alice M Lucas, Anne E Carpenter, Beth A Cimini, and Allen Goodman. Cellprofiler 4: improvements in speed, utility and usability. *BMC bioinformatics*, 22:1–11, 2021.

[20] Carolina Wählby. *Algorithms for applied digital image cytometry*. PhD thesis, Acta Universitatis Upsaliensis, 2003.

[21] World Health Organization: WHO and World Health Organization: WHO. Malaria, 12 2024.

[22] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. Detrs beat yolos on real-time object detection, 2023.