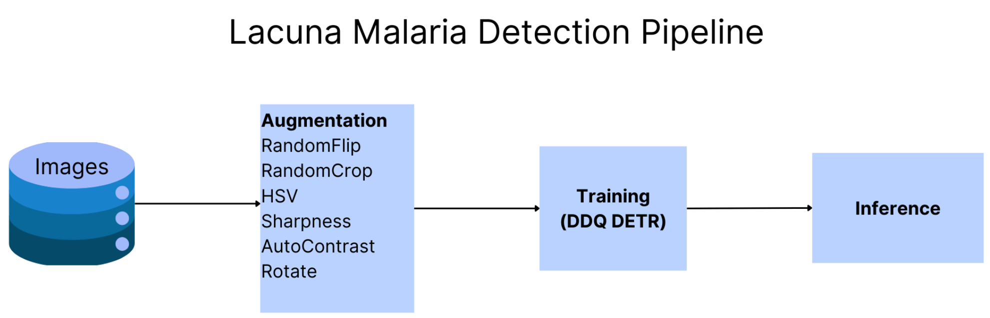
Lacuna Malaria Detection Challenge - 3rd Place solution

1. Overview

The objective of the challenge was to develop a multiclass object detection and classification model capable of accurately localizing and classifying malaria parasites in blood slide images.
The solution employs a single DDQ DETR model, trained on the complete training dataset using the MMDetection library.

2. Architecture diagram



3. ETL process

The data was extracted to mirror the directory structure below:

```
.
└── /workspace/
    └── mmdetection/
        └── data/
            ├── images/
            ├── Test.csv
            ├── Train.csv
            └── SampleSubmission.csv
```

The data was transformed into the COCO annotation format required by the MMDetection library. This included mapping bounding boxes and class labels to COCO-compliant JSON files.

4. Data modeling

The following augmentations were applied to the training dataset during training:

- *RandomFlip*
- *RandomChoiceResize*
- *YOLOXHSVRandomAug*
- *Sharpness*
- *AutoContrast*
- *Rotation*

The Swin-L variant of the Dense Distinct Query for End-to-End Object Detection (DDQ-DETR) model was selected based on its promising performance on object detection tasks. The model was trained for 30 epochs with the following parameters:

Training Parameters

- **Epochs:** 30
- **Batch size:** 2
- **Optimizer:** AdamW (learning rate = 0.0002, weight decay = 0.05)
- **LR Schedule:**
    - **Base LR:** 1e-4
    - **Warmup:** LinearLR warm-up for the first 2000 iterations, increasing learning rate from 0.0001 × base_lr to base_lr.
    - **Multi-step Decay:** Constant learning rate until epoch 20, reduced by gamma = 0.1 at epochs 20 and 26.

5. Inference

The model was deployed using the MMDetection framework on a GPU-enabled machine for efficient inference.

Deployment workflow:

1. Load trained model weights (`epoch_30.pth`) and configuration file (`cfg`) with `init_detector`.
2. Sequentially process test images from a DataFrame (`df_test`) containing image paths and IDs.
3. Perform object detection using `inference_detector` to generate bounding boxes, class labels, and confidence scores.

4. Apply post-processing:
    - Filter predictions with a confidence threshold (`minconf = 0.05`).
    - Record a default entry with a class label of `NEG` if no detections are found.
    - Structure and save detected bounding boxes, scores, and labels in a dictionary format.

Model updates and retraining:

- New data can be incorporated through fine-tuning or full retraining.
- If fine-tuning, modify the `load_from` parameter in the configuration to use the current model weights.

6. Run time

- Training: 12 hours
- Inference: 15 minutes

7. Performance metrics

- **Disk space:** The model required ~100 GB to store checkpoints after each epoch during training.
- **GPU VRAM usage:** Peak usage was ~45 GB during training.
- **Evaluation scores:**
    - **Public leaderboard mAP@0.5:** 0.92529841
    - **Private leaderboard mAP@0.5:** 0.921717525