Introduction

My approach leverages a two-stage modeling pipeline to effectively detect and classify white blood cells. Stage 1 is a universal White Blood Cell Detection. This stage employs an object detection model to identify white blood cells, irrespective of their underlying class. The goal is to robustly locate and distinguish white blood cells from other elements in the image. Stage 2 is classification once white blood cells are detected in the first stage, a second-stage model classifies them into specific categories. Alternatively, this stage can directly learn class labels when working with full images in the absence of explicit white cell detections. This modular pipeline ensures accurate localization and precise classification, optimizing performance for varying data scenarios and improving the adaptability of the solution.

My approach differs from all-in-one models like YOLO, which combine object detection and classification. By introducing a second stage focused on classification, my method provides greater control over the most challenging aspect of this task - accurate classification - whereas detecting white blood cells is relatively straightforward. I am confident in this approach, as it secured me a spot among the top three winners in a similar competition last year.

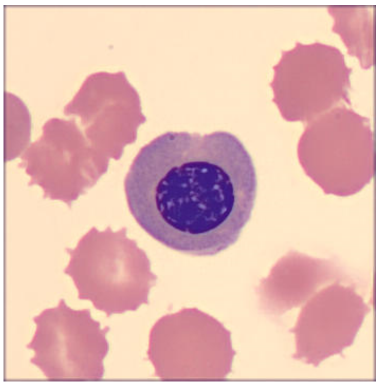
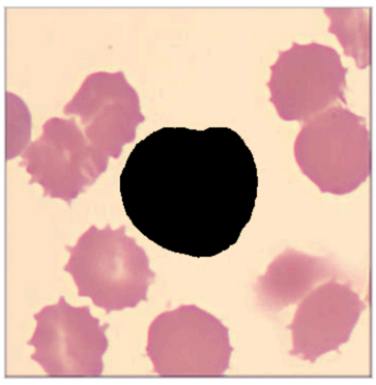
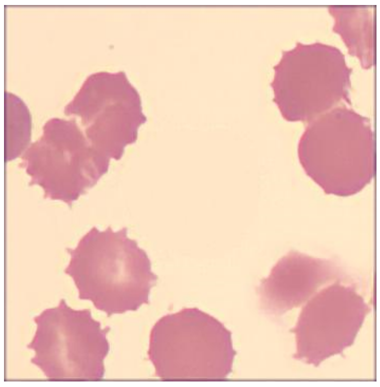
Training

Object detection

Two object detectors have been trained: The first one with a purpose of maximizing GIOU score by including all training data without any cleaning (noisy boxes kept) assuming that noise distribution on test data will be the same. The second one with a purpose of extracting clean boxes in order to train the next classification stage. Both object detectors are single class (white blood cell) only. A few object detector models have been evaluated (YOLO, YOLOX, RT-DETRv1/v2) but I've selected YOLOX because it provided good performance and an open source license. (YOLO from Utralytics provided nice results too but its license could restrict usage when included in a full solution). Notice that I've modified the YOLOX implementation to use GIOU loss instead of IOU loss.

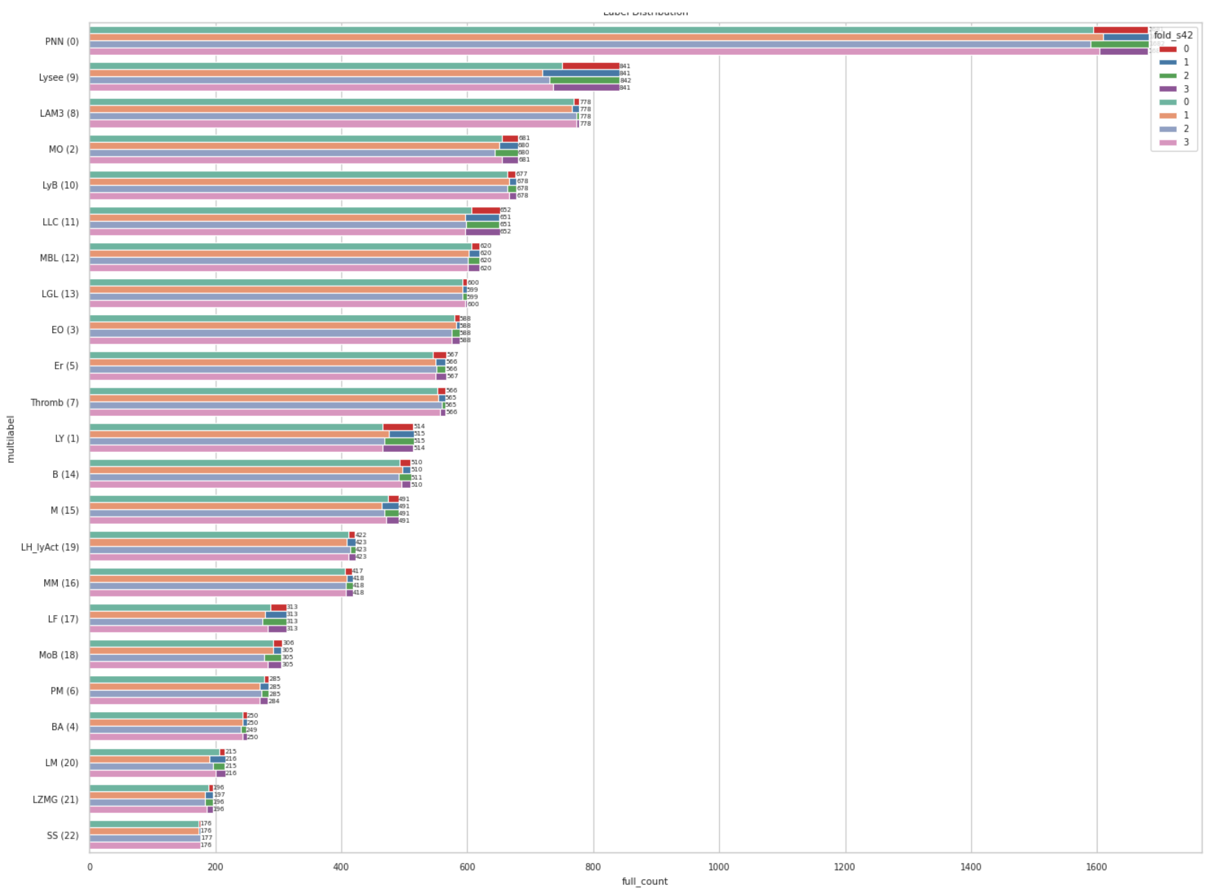
Background images

Object detection models perform better when background images (no white blood cell) are included in the training, it reduces the false positives. I've generated around 3.5k background images based on boxes detected followed by an OTSU thresholding to define a mask and custom color filler.

 Original image 00716a0e-d	 Mask detected (with OTSU thresholding)	 Background image (Mask filled in with main color)

Cross Validation

A good cross validation scheme is key in any training. As my stage 2 will consider multi-classes and multi-labels models, I've split my train/valid datasets based on a multi-labels stratification. Each fold includes a similar distribution of single label, multiple labels, small/regular/medium/big/large cells and colors (mean/std binarization). I've opted for 4 folds (75%/25%) to limit the overall training time.



Classification

Input of my Stage 2 classification is the output of my Stage 1 cross validation. Boxes detected are cropped from each image with an additional margin of 16 pixels to capture more cell's context. During the cross validation I mapped each box detected to ground truth based on GIOU best score to associate the related white cell class. We're now back to a regular multi-classes classification task. The power of CNN, transformers and foundation models is available to make experiments. We can play with pre-trained models, image size and augmentations to find and finetune some good SOTA performing models.

The training procedure is similar for each experiment and quite classical in computer vision problems:

- Augmentations: Horizontal/Vertical flips, rotation 90, ShiftScaleRotate, Random Brightness Contrast, Random Gamma, HueSaturationValue, ColorJitter, GaussianBlur, MotionBlur, CoarseDropout, CutMix and MixUp.
- Optimizer: AdamW
- Loss: CrossEntropy
- Learning rate: 5e-4 for CNN, 2e-4 for transformers, CosineAnnealing scheduler
- Epochs: 32 for baseline and 24 when fine tuning.
- Batch size: 32 for CNN and transformers, 256 for foundational.
- Precision: 16-mixed

Final models selected:
CNN: timm_tf_efficientnetv2_m with image size = 512
Transformers:

- timm_vit_large_patch16_224 with image size = 224
- timm_nextvit_large with image size = 384
- timm_tiny_vit_21m_512 with image size = 512
- DinoBloom (foundation, DinoV2-based), last 5 layers frozen, image size = 224

In parallel of multi-classes models, I've trained a multi-labels CNN model as 85% of images come with a single cell and 15% with more. This model will be exclusively used for later ensemble purposes. A model trained for multi-labels on full images brings additional knowledge and diversity. This model also takes into account a new class by including the background images and is trained with a focal loss to take into account the imbalance of the labels.

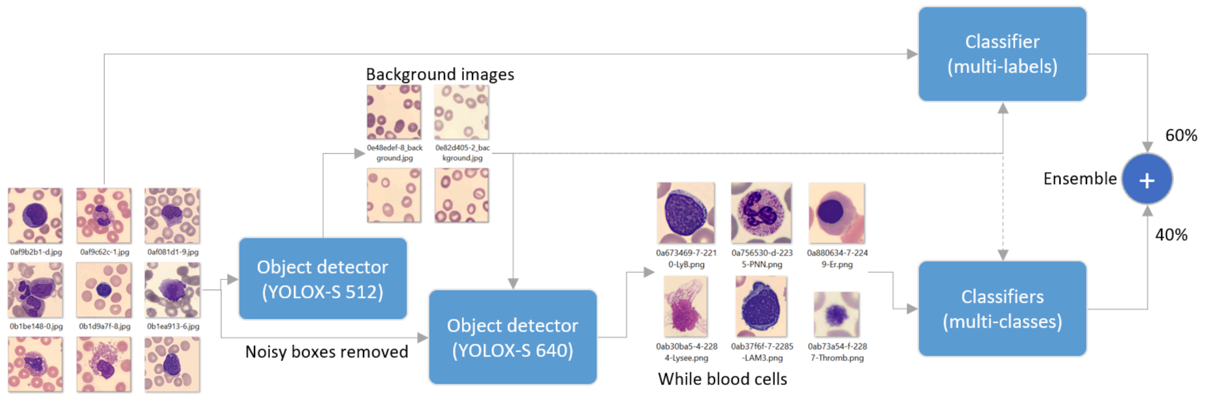
Inference

Test time augmentation (TTA - horizontal flip only) is enabled on some models to improve predictions. Background class is ignored for the multi-labels model. According to the challenge requirements the runtime for inference is limited to 500ms (GPU allowed). The object detector and classifier models are executed on the test dataset (20751 images) in around 2h40min on a single RTX3090 which fits the requirement with 462ms per image.

Ensemble

Predictions of the object detector model are ensembled with a Weighted Fusion Boxes algorithm. Predictions for the classifier models are ensemble with a simple weighted average (40%, 60%) between multi-classes models and multi-labels models limited to the images with a single box. The weights have been found by grid search on cross validation results.

The full training pipeline can be summarized as below:


Background images
Classifier (multi-labels)
60%
Ensemble
+
40%
Object detector (YOLOX-S 512)
Noisy boxes removed
Object detector (YOLOX-S 640)
While blood cells
Classifiers (multi-classes)