Cervical Cancer Model

November 3, 2025

1 Methods

1.1 Dataset

A total of 24 SVS files containing diagnostic annotations from certified cervical cytology specialists were used as the raw dataset for model development. The expert labels were exported to CSV format using QuPath, and each SVS image was subsequently divided into non-overlapping 1024×1024 pixel patches. After processing, the dataset consisted of 591 Lesion patches and 1302 NILM (Normal) image patches, which were used for model training and evaluation.
To further enhance dataset quality, two additional reinforcement cycles were performed. In each cycle, high-confidence predictions generated by the trained model on unlabeled image patches were reviewed by professional cytologists, who confirmed or corrected the predicted labels. The verified data were then merged back into the original dataset, forming progressively larger and more refined training sets for subsequent model iterations. The number of labels in each version of the dataset after these refinement cycles is summarized in Table below.

Table 1: Label distribution across different dataset versions.

Dataset	**Train (Lesion)**	**Train (Normal)**	**Val (Lesion)**	**Val (Normal)**	**Total (Lesion)**	**Total (Normal)**	**Total Labels**
Initial	1168	2551	271	683	1439	3234	4673
After_First_Merge	1245	2561	271	683	1516	3244	4760
After_Second_Merge	1547	2624	271	683	1818	3307	5125

1.2 Preprocessing

The dataset was provided as whole-slide images (WSIs) in `.svs` format, with corresponding annotations stored in QuPath `.qpdata` files. We first converted the `.qpdata` annotations into a tabular format using a Groovy script, which produced `.csv` files containing the following fields for each annotation: `type`, `boundary_center_x`, `boundary_center_y`, `height`, and `width`.

For each WSI, we extracted non-overlapping image patches of size 1024×1024 pixels. Since the slide dimensions are not always multiples of 1024, the final row and column of patches were allowed to overlap slightly to cover the full image.

We only kept a patch was retained if it contained at least 40% of an original annotation bounding box and otherwise, it was discarded. The retained 1024×1024 patches were subsequently downsampled to 256×256 pixels for training efficiency.

Finally, the processed patches and their corresponding annotation boxes were converted into YOLO format. The dataset was split randomly into 80% training and 20% validation sets.

1.3 Model Architecture

We employed the YOLOv12m object detection framework (Ultralytics), initialized with pretrained COCO weights. The network architecture consisted of a CSPDarknet backbone with an anchor-free detection head. The loss function combined classification, bounding box regression, and objectness terms. Optimization was performed using AdamW with weight decay, and the learning rate was set to 1×10^{-4} with a cosine annealing schedule.

1.4 Iterative Pseudo-Labeling

Because only a portion of the dataset was annotated, we implemented a multi-generation pseudo-labeling strategy:

1. Train YOLO on the labeled subset.
2. Run inference on the unlabeled patches.
3. Select predictions using *class-specific* thresholds and caps (see Table 1): retain detections with confidence $\geq$ the per-class threshold and keep up to the top-k highest-scoring detections per class per image (0.90 and top-5 for Lesion; 0.90 and top-3 for NILM).
4. Merge the resulting pseudo-labels with the training set.
5. Retrain YOLO on the expanded dataset.
6. Repeat for $N = 20$ generations.

This iterative procedure allowed the model to progressively leverage unlabeled data while reducing the impact of noisy predictions.

Table 2: Per class thresholds.

Class	`per_class_conf`	`per_class_topk`
Lesion	0.90	5
NILM	0.90	3

1.5 Training Procedure

Each generation was trained for 400 (with early stop) epochs with images resized to 256×256. Training ran on `device`=0 with `workers`=4 and early stopping `patience`=40. Rectangular batching was enabled (`rect=true`), and the first `freeze`=10 layers were kept fixed during fine-tuning. The batch setting `batch`=-1 lets the framework choose the largest batch that fits GPU memory.

Optimization. A cosine learning–rate schedule (`cos_lr=true`) was used with initial rate `lr0` $= 8 \times 10^{-4}$ and final factor `lrf` $= 0.02$. Label smoothing of `label_smoothing` $= 0.10$ was applied.

Data augmentation. Geometric transforms were minimal. Scaling was limited to `scale`=0.10. Horizontal flip was used with probability `fliplr`=0.5, while vertical flip was disabled (`flipud`=0). Color jitter in HSV space used `hsv_h` $= 0$, `hsv_s` $= 0.2$, and `hsv_v` $= 0.2$. Mosaic and mixup were disabled (`mosaic`=0, `mixup`=0; `close_mosaic`=0).

Loss / task weights. The detection losses used weights `box` $= 8.0$, `cls` $= 0.3$, and `dfl` $= 1.2$. The IoU target was set to `iou` $= 0.7$.

Table 2 lists the exact configuration.

Table 3: Training configuration.

General	
`epochs`	400
`imgsz`	256
`batch`	-1
`device`	0
`workers`	4
`patience`	40
`rect`	`true`
`freeze`	10
Optimization	
`cos_lr`	`true`
`lr0`	0.0008
`lrf`	0.02
`label_smoothing`	0.10
Data augmentation	
`degrees`	0.0
`shear`	0.0
`perspective`	0.0
`translate`	0.0
`scale`	0.10
`fliplr`	0.5
`flipud`	0.0
`hsv_h`	0.0
`hsv_s`	0.2
`hsv_v`	0.2
`mosaic`	0.0
`mixup`	0.0
`close_mosaic`	0
Loss / task weights	
`box`	8.0
`cls`	0.3
`dfl`	1.2
`iou`	0.7

1.6 Evaluation

Validation was performed by sweeping the confidence threshold from (0.10) to (0.60) in increments of (0.05) and saving the confusion matrix for each setting. For every generation within each sweep, per-class metrics were computed from the confusion matrix. To formalize these computations, the following confusion-matrix notation is adopted.

Let cm_{ij} denote a three class confusion matrix with *predicted* class $i \in \{L, N, B\}$

on rows and *true* class $j \in \{L, N, B\}$ on columns, where L = Lesion, N = NILM, and B = Background. In the figures the x–axis (columns) is True and the y–axis (rows) is Predicted. Define the *true* column totals $T_j = \sum_i cm_{ij}$ and the *predicted* row totals $P_i = \sum_j cm_{ij}$.

Aggregate metrics over non background truths.

$$D = \sum_{j\in\{L,N\}} \sum_{i\in\{L,N,B\}} cm_{ij},$$

$$\text{Total\_Acc} = \frac{cm_{LL} + cm_{NN}}{D},$$

$$\text{Total\_Background\_rate} = \frac{cm_{BL} + cm_{BN}}{D}.$$

Per class column normalised quantities (conditioning on the true class). Here R_L is the Lesion recall (sensitivity), F_L^{BG} is the probability that a true Lesion is predicted as Background (Lesion→Background), and F_L^{X} is the probability that a true Lesion is predicted as NILM (Lesion→NILM); analogously R_N, F_N^{BG}, and F_N^{X} are defined for NILM.

$$R_L = \frac{cm_{LL}}{T_L}, \qquad F_L^{\text{BG}} = \frac{cm_{BL}}{T_L}, \qquad F_L^{\text{X}} = \frac{cm_{NL}}{T_L},$$

$$R_N = \frac{cm_{NN}}{T_N}, \qquad F_N^{\text{BG}} = \frac{cm_{BN}}{T_N}, \qquad F_N^{\text{X}} = \frac{cm_{LN}}{T_N}.$$

Thresholds were specified for eight metrics and were refined after each validation run to select the best model for this application. The update policy followed a fixed priority: first, minimize the false label rates for each class $(F_L^{\text{X}}, F_N^{\text{X}})$; second, increase the aggregate accuracy Total_Acc; third, minimize the background misclassification rates, both the aggregate Total_Background_rate and the per class terms $(F_L^{\text{BG}}, F_N^{\text{BG}})$. The final selection balances these three goals. The final thresholds for each matrix are shown below:

1. Accuracy ≥ 0.5
2. Total False Background Rate ≤ 0.5
3. Lesion Recall ≥ 0.37
4. Lesion False Background Rate ≤ 0.65
5. Lesion False Label Rate ≤ 0.02
6. Normal Recall ≥ 0.61
7. Normal False Background Rate ≤ 0.40
8. Normal False Label Rate ≤ 0.02

1.7 Final Model Selection and Performance Evolution

Final Model Selection Based on the evaluation metrics discussed above, the final model selected corresponds to the configuration initialized with the random seed `seed1390368751`. This model achieved the best validation performance after 13 self training cycles, where each cycle progressively incorporated high-confidence pseudo-labeled data into subsequent training rounds without human supervision. The model trained under this configuration demonstrated the most stable and accurate predictions, producing the best overall output when the validation confidence threshold (`val_conf`) was set to 0.15.

Performance Evolution Across Generations The following results illustrate the progressive improvement of the model across successive training generations. In the following trend plot, the overall accuracy, Lesion recall, and Normal (NILM) recall (shown in blue, green brown respectively) exhibit a generally increasing trend across generations, while the background and false prediction rates gradually decrease, indicating more stable and reliable model performance over time.

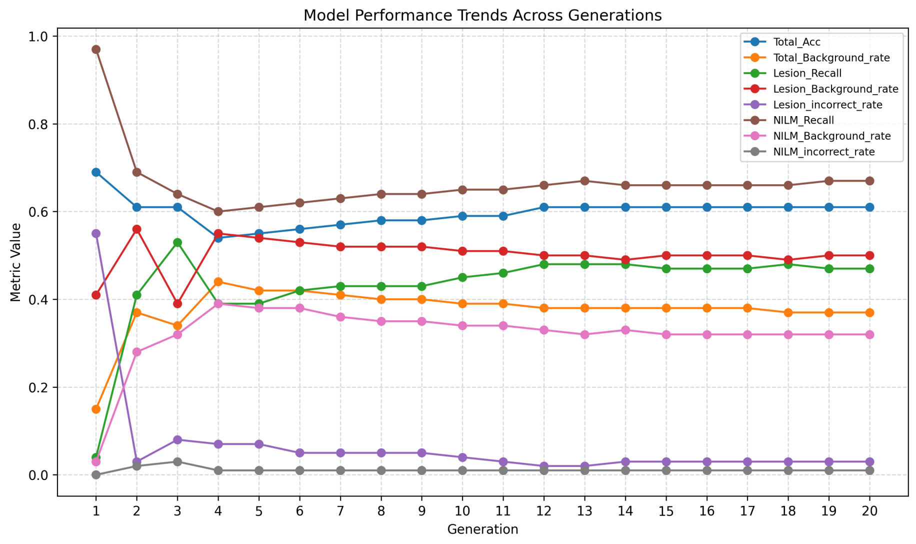
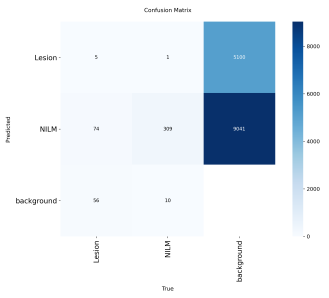
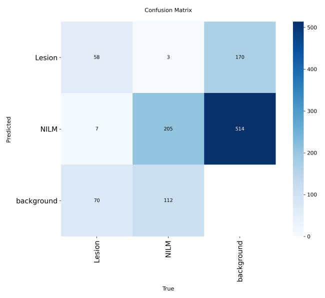


Figure 1: Trend of each matrices across training cycles

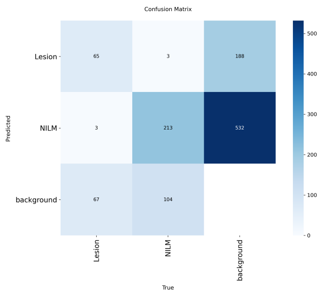
The following confusion matrices further demonstrate the progressive improvement of the model achieved through the self-training framework.



(a) Generation 1



(b) Generation 8



(c) Generation 13

Figure 2: Comparison of confusion matrices across training generations, showing progressive improvement with reduced false background and label rates.

1.8 Output and Reproducibility

All best checkpoints were stored as `.pt` files, and evaluation metrics were logged in CSV format. Precision-recall curves and training loss curves were plotted for visualization. To account for stochasticity, all experiments were repeated three to five times with different random seeds.
add github link here