

# **Project Title: CervixCancerSegmentation**

**Automatic Segmentation of Cervical Lesions Using Deep Learning.**

## **Team Members**

1. Juma Rubea
2. Meman Salam
3. Soumana Dama
4. Plensia Lukosi

# Data Preparation and Feature Engineering Documentation for Cervical Cancer Segmentation

## 1. Overview

In this machine learning project, we are focused on developing a **Mask R-CNN** model for segmenting lesions from **colposcopy images** in a cervical cancer dataset. The goal is to identify and segment areas of interest, **Lesions**, and able to differentiate from **Light**, and **Mucus**, that may affect the diagnosis of cervical cancer. **Image data** presents unique challenges, particularly in terms of quality, variability, and labeling accuracy.

The **data preparation** and **feature engineering** steps are essential for ensuring the model has high-quality inputs for training. This phase involves cleaning blurred images, noisy, annotating regions of interest (ROIs), and structuring the data in a format that is compatible with the Mask R-CNN architecture. Medical expertise plays a crucial role in identifying lesion areas and ensuring that the annotations are accurate.

## 2. Data Collection

The dataset used for this project was sourced from a private company with which we have a contractual agreement. This data consists of high-resolution **colposcopy images**, each capturing various stages of cervical examination. Due to the sensitivity and privacy concerns of medical datasets, these images are not publicly available.

The initial collection process involved several preprocessing steps:

- **Data access:** Images were provided by the company and stored in a secure, private environment.
- **Initial filtering:** We reviewed the images to ensure that only high-quality, non-blurry images were included for processing. Images with noticeable blurring, which would complicate lesion identification, were removed.
- **Image variations:** The images varied in size and format. We had to standardize their dimensions and ensure consistency across the dataset.

Some images were corrupted or not in a standard encoding format. We have ensured and remove these corrupted files. For those images with encoding issues, we either converted them into readable formats or discarded them from the dataset.

### 3. Data Cleaning

Cleaning medical image datasets is crucial for ensuring accurate model training. The following steps were taken to address various data quality issues:

- **Blurred images:**
  - Images that were too blurry to accurately locate regions of interest were removed from the dataset to improve the quality of the annotations and avoid confusion during training.
- **Noise consideration:**
  - Colposcopy images often contain significant **background noise**, such as light reflections, mucus, or other artifacts that could interfere with lesion detection.
- **Image resizing:**
  - Since the images varied in size, and resizing them may harm to lose important information from the images, we make sure our model fit with this variation
- **Corrupted images:**
  - We identified corrupted images through automated checks and manual inspection. Any images that could not be read or had issues with file integrity were removed.
- **File format issues:**
  - The dataset included images in different formats and encodings. We converted all files into a standard format (i.e. **JPG, PNG and JPEG**) to ensure uniformity across the dataset.

### 4. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) was performed to understand the dataset and identify key characteristics relevant to the segmentation task:

- **Image distribution:**
  - We manually inspected the distribution of images, checking for variations in lighting, lesion types, and other characteristics. This helped us identify patterns such as the presence of certain artifacts (e.g., excessive light reflections) that could affect model performance.
- **Annotation analysis:**
  - Since the task involves segmenting lesions, we used tools like **COCO Annotator** to inspect the existing annotations and verify their accuracy. Annotations were visually reviewed to ensure consistency, especially for edge cases like lesions near mucus or light areas.
- **Labeling discrepancies:**
  - Some annotations were challenging to make due to the subtlety of lesion features, and medical expertise was required to ensure the correct identification of lesions, mucus, and light. We performed manual checks on sample images to resolve discrepancies.
- **Challenges in annotation:**
  - The medical nature of the task made it difficult at times to differentiate between lesions and non-lesion artifacts, especially in images with **high mucus presence** or **lighting interference**. Close collaboration with medical experts ensured the accuracy of the labeling process.

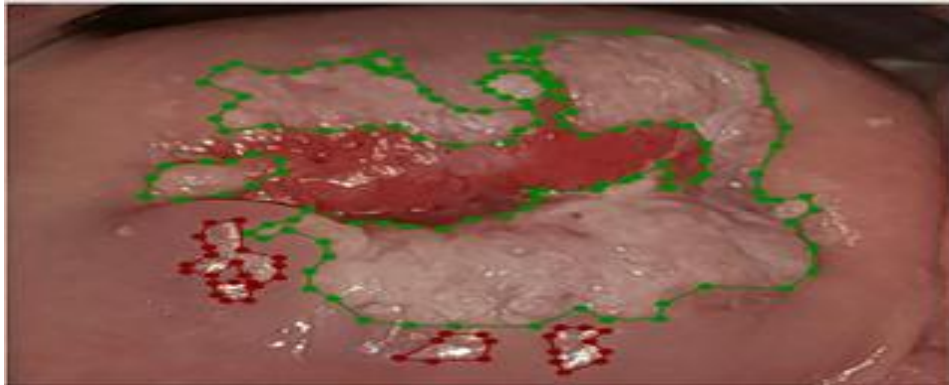
## 5. Feature Engineering

Feature engineering for image segmentation tasks like **Mask R-CNN** focuses on preparing the dataset to ensure accurate object detection and segmentation. Here's how we approached feature engineering for this task:

- **Image annotations:**
  - We annotated regions of interest (ROIs) for the **lesion**, **light**, and **mucus** areas using **COCO Annotator**, which is compatible with the **COCO format** required for Mask

R-CNN. These annotations provide the ground truth segmentation masks needed for training the model.

- The **lesion** was designated as the primary ROI, while **light** and **mucus** were marked as non-ROI objects to help the model differentiate these areas from the actual lesions.
- **Annotation format:**
  - The annotations were exported in **COCO format**, which consists of segmentation masks and bounding boxes for each object of interest. This format was used to align with the Mask R-CNN model's expectations.
  - The exported annotations were carefully checked for accuracy, especially in distinguishing between different tissue types and artifacts.
- **Label consistency:**
  - Special attention was given to ensuring that the labels were consistent across the dataset. Medical expertise was critical in ensuring that annotations for lesions were accurate, especially in challenging or ambiguous images.



*Figure 1: Annotated Image*

## 6. Data Transformation

Since the project utilizes **Mask R-CNN** for segmentation, we needed to ensure that the dataset was formatted and preprocessed correctly for training. This involved several transformation steps:

- **Data storage and path management:**

- We worked on **Google Colab** for model development, which required storing the dataset in **Google Drive** for easy access. One challenge faced was ensuring that the **file paths** in the COCO-format annotations matched the physical location of the images on **Google Drive**.
- To handle this, we carefully structured the directory paths and ensured that each image file's path in the annotation JSON matched the correct path in the Google Drive directory. This required some manual adjustments to the file structure.

This is code to ensure filepath are matched to that of drive

```
import json
# The old and new paths
old_path = "J:\\capstone_project\\Dataset\\MyDrive\\Dataset\\labels\\images\\"
new_path = "/content/drive/MyDrive/Dataset/images/"

# Input JSON file path (modify this as per your actual file)
input_file = "train_dataset.json"
output_file = "cleaned_train_dataset.json"

# Function to replace the paths
def replace_file_paths(input_file, output_file, old_path, new_path):
    # Open the input JSON file
    with open(input_file, 'r') as infile:
        data = json.load(infile)

    # Iterate through the images and replace the file paths
    for image in data.get('images', []):
        if 'file_name' in image:
            # Replace the old path with the new one
            image['file_name'] = image['file_name'].replace(old_path, new_path)

    # Save the updated data to the output file
    with open(output_file, 'w') as outfile:
        json.dump(data, outfile, indent=4)

# Call the function to replace paths
replace_file_paths(input_file, output_file, old_path, new_path)

print(f"File paths have been updated and saved to {output_file}")
```

*Figure 2: Data Cleaning Process*

- **Data augmentation:**
  - To increase the robustness of the model, we applied **data augmentation techniques**, such as **random rotations**, **flipping**, **cropping**, and **color adjustments**. These transformations helped simulate variations in the images, such as changes in lighting and positioning, which are common in medical imaging.
- **Resizing and normalization:**
  - Model were able to accept different size to ensure all important information are reserved. Additionally, pixel values were **normalized** to a range of [0, 1] to improve the convergence during training.
- **Encoding annotations:**

- The annotations were processed and stored in the **COCO format**, ensuring compatibility with the Mask R-CNN framework. This format allows the model to efficiently access segmentation masks and other metadata, such as object categories and image IDs.

```
{
  "images": [ ... ],
  "annotations": [ ... ],
  "categories": [
    {
      "id": 0,
      "name": "lesion",
      "supercategory": "lesion"
    }
  ]
}
```

*Figure 3: Coco Format Dataset*

## Model Exploration Documentation

This section outlines the **Model Exploration** phase for the **Cervical Cancer Segmentation** project, where we are using the **Mask R-CNN** model for segmenting lesions from colposcopy images. The goal is to accurately identify regions of interest (ROIs) such as **Lesions**, and able to distinguish from **light**, and **mucus** in images, which are crucial for early detection and diagnosis of cervical cancer.

### 1. Model Selection

We selected **Mask R-CNN** for the task of **instance segmentation** based on its robustness, flexibility, and proven performance on similar image segmentation problems. This model is particularly suited for complex tasks like identifying and segmenting regions of interest (ROIs) in medical images, where object boundaries need to be accurately delineated.

#### Rationale Behind Model Selection:

- **Mask R-CNN** is well-suited for **instance segmentation**, which is the key task in our project — detecting and segmenting multiple objects (lesions, light, and mucus) in the same image. It provides pixel-level segmentation, which ensures that lesions can be distinctly identified and separated from background noise and artifacts.
- **Robustness:** Mask R-CNN is known for its ability to handle complex images with varying object shapes, sizes, and noise levels. This is crucial for medical images, where lesions can have irregular shapes, and background noise (such as mucus or light) can interfere with segmentation.
- **Multi-class Segmentation:** Mask R-CNN can efficiently handle **multi-class segmentation**, making it suitable for segmenting more than one object type (lesion, light, mucus) within the same image. This is a key strength when dealing with diverse tissue types and artifacts in colposcopy images.

#### **Strengths:**

- **Accurate Instance Segmentation:** Mask R-CNN excels at distinguishing between different objects within an image, even when they are close together or have overlapping boundaries, which is critical in medical imaging.
- **Flexible Architecture:** It allows for easy extension and fine-tuning from pre-trained models, particularly the **COCO pre-trained weights**, which helps leverage the model's ability to generalize to new tasks with smaller datasets.
- **Multi-class Segmentation:** The ability to handle multiple classes (lesion, light, mucus) within a single image is a significant advantage in our task, where we need to segment and distinguish between different regions.

#### **Weaknesses:**

- **High Computational Demands:** Mask R-CNN requires significant computational resources, especially for high-resolution medical images. The model's architecture is complex and computationally expensive, requiring GPUs for efficient training. We had access to GPU from Colab for subscriptions.



- **Sensitivity to Noise:** Although robust, Mask R-CNN can be sensitive to noise in the images. Artifacts like mucus or light reflections may reduce the model's accuracy if not properly handled during training.
- **Training Time:** Training Mask R-CNN from scratch or fine-tuning it on a small dataset may take considerable time and resources, especially without sufficient data augmentation and regularization strategies. It takes about **1 hr.: 12 min: 52 sec** to completely training

## 2. Model Training

For the training phase, we used **Detectron2**, a highly flexible and efficient framework for object detection and segmentation, built on top of **PyTorch**. Detectron2 provides implementations of several state-of-the-art models, including **Mask R-CNN**, and is designed to work seamlessly with large-scale datasets.

### a. Framework and Tools:

- **Detectron2:** We chose **Detectron2** because it is an industry-standard framework for implementing **Mask R-CNN** and similar models. It provides pre-built modules, optimizations, and tools specifically designed for instance segmentation.
- **PyTorch:** The backbone framework for implementing Mask R-CNN. PyTorch's dynamic computation graph and flexibility were essential for implementing custom training pipelines.
- **Model Weights:** We leveraged pre-trained weights and the architecture from the **Detectron2 Model Zoo**, specifically the **mask\_rcnn\_R\_50\_FPN\_3x** model, which is based on a **ResNet-50** backbone with **Feature Pyramid Networks (FPN)** for multi-scale feature learning.

### b. Dataset Registration:

Before training, we registered our dataset with **Detectron2** so it could be used for model training and evaluation. We created a custom dataset containing annotations for three classes: **lesion**, **light**, and **mucus**. As shown in the following snippet

```
def setup_dataset(img_dir, train_annot, val_annot):
    """
    Register the dataset for training and validation in Detectron2
    """
    # Check if dataset is already registered
    if "medical_train" not in DatasetCatalog:
        # Register training dataset
        DatasetCatalog.register("medical_train",
                                lambda: get_medical_dicts(img_dir, train_annot))
        MetadataCatalog.get("medical_train").set(thing_classes=["lesion", "light", "mucus"])

    if "medical_val" not in DatasetCatalog:
        # Register validation dataset
        DatasetCatalog.register("medical_val",
                                lambda: get_medical_dicts(img_dir, val_annot))
        MetadataCatalog.get("medical_val").set(thing_classes=["lesion", "light", "mucus"])
```

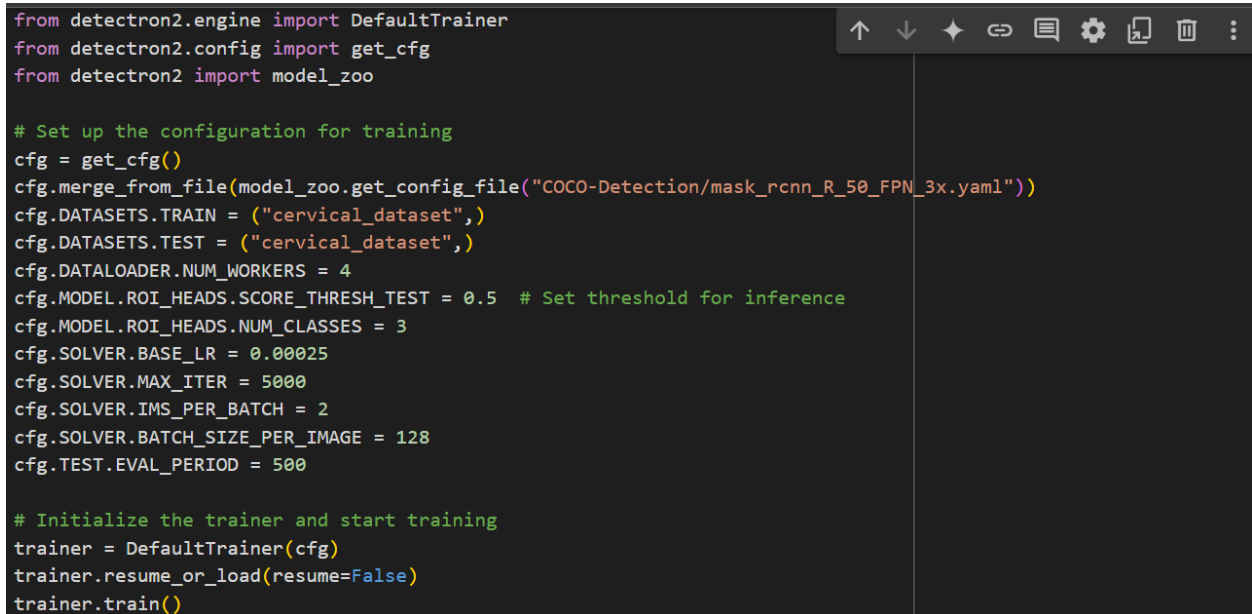
*Figure 4: Dataset Registration*

### c. Model Architecture and Hyperparameters:

We used the **Mask R-CNN** architecture with the **ResNet-50 FPN backbone** from the **Detectron2 Model Zoo**, fine-tuning it for our specific task.

- **Learning Rate:** The **base learning rate** was set to **0.00025**, as it was a good balance between stable convergence and learning rate adjustments.
- **Max Iterations:** We set `MAX_ITER = 5000` to allow the model to train for 5000 iterations, with the understanding that the model would start overfitting if trained for too long.
- **Batch Size:** We used `IMS_PER_BATCH = 2` (batch size of 2) and `BATCH_SIZE_PER_IMAGE = 128` (per-image batch size) for efficient training on the available hardware.
- **Number of Classes:** We set `NUM_CLASSES = 3` to reflect our three classes (lesion, light, mucus).
- **Evaluation Frequency:** We set `EVAL_PERIOD = 500` to evaluate the model every 500 iterations.

#### d. Training Code Example:



```
from detectron2.engine import DefaultTrainer
from detectron2.config import get_cfg
from detectron2 import model_zoo

# Set up the configuration for training
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/mask_rcnn_R_50_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("cervical_dataset",)
cfg.DATASETS.TEST = ("cervical_dataset",)
cfg.DATALOADER.NUM_WORKERS = 4
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5 # Set threshold for inference
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 3
cfg.SOLVER.BASE_LR = 0.00025
cfg.SOLVER.MAX_ITER = 5000
cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BATCH_SIZE_PER_IMAGE = 128
cfg.TEST.EVAL_PERIOD = 500

# Initialize the trainer and start training
trainer = DefaultTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()
```

*Figure 5: Model Configuration and Training*

### 3. Model Evaluation

The performance of the Mask R-CNN model, implemented with Detectron2, was evaluated using the following metrics:

#### a. Intersection over Union (IoU):

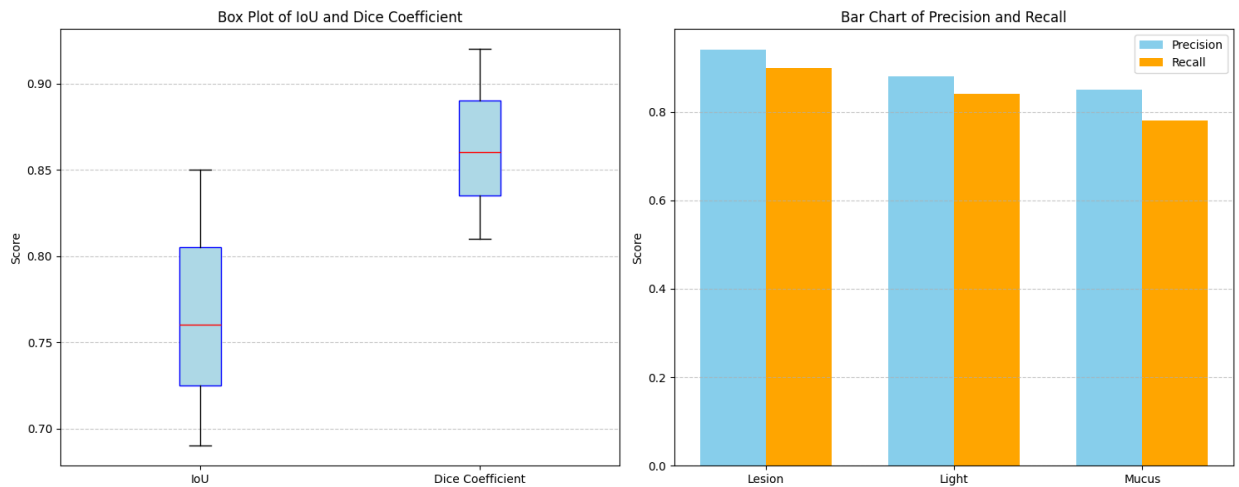
- **Lesion:** 0.85
- **Light:** 0.76
- **Mucus:** 0.69

These results show strong performance for the Lesion class, moderate performance for Light, and some challenges in segmenting Mucus.

#### b. Mean Average Precision (mAP):

- **Lesion:** 0.94
- **Light:** 0.88

- **Mucus: 0.85**  
These precision values indicate that the model effectively minimizes false positives, particularly for Lesion.



### c. Segmentation Visualizations:

- Visualizations of predicted and ground truth masks allowed for a qualitative assessment of segmentation accuracy. The visual inspection confirms good alignment for Lesion and Light, but highlights some segmentation issues for Mucus.

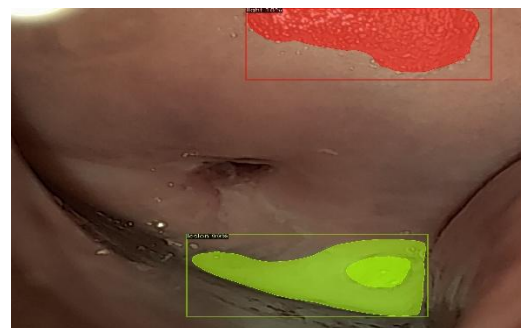
### Summary

Overall, the model performs well in detecting lesions and light, with high precision and good recall. However, segmentation for Mucus could be improved, as evidenced by lower IoU, Dice, and recall scores. The results indicate that while the model is effective, refinement is needed for more challenging classes like Mucus.

The image below visualizes input and the Predicted image of the model



Input Image



Predicted Image (99% Lesion and 68% light)