A REPORT

ON

LESION LOCALIZATION AND PRIORITIZATION THROUGH COMPUTER VISION TECHNIQUES FOR POINT OF CARE IMAGING DEVICES

BY

RAKUL CHAUHAN **2021AAPS1971H**

&

RISHIKA KALRA **2021A3PS2651P**

AT

CSIR-CEERI (Central Electronics Engineering Research Institute), PILANI

A Practice School-I Station of

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

(June,2023)

A REPORT

ON

LESION LOCALIZATION AND PRIORITIZATION THROUGH COMPUTER VISION TECHNIQUES FOR POINT OF CARE IMAGING DEVICES

BY

RAKUL CHAUHAN	**2021AAPS1971H**	**B.E. ELECTRONICS AND COMMUNICATION**

&

RISHIKA KALRA	**2021A3PS2651P**	**B.E. ELECTRICAL AND ELECTRONICS**

Prepared in partial fulfillment of the

Practice School-I Course Nos.

BITS C221/BITS C231/BITS C241

AT

CSIR-CEERI (Central Electronics Engineering Research Institute), PILANI

A Practice School-I Station of

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

(June,2023)

ACKNOWLEDGEMENTS

I would like to extend my heartfelt thanks and deep appreciation to the individuals who have provided invaluable contributions and unwavering support during the successful completion of the project titled "Lesion Localization and Prioritization Using Computer Vision Techniques for Point of Care Imaging Devices."

I extend my sincere thanks and deep appreciation to Professors Meetha V. Shenoy Ma'am and Abhijit Asati Sir, our esteemed faculty-in-charge at BITS, for their invaluable guidance and mentorship. Their expertise and insightful perspectives have significantly influenced the quality and direction of my work.

Additionally, I would like to express my utmost gratitude to Dr. Satyam Srivastava and Dr. Shalini for their invaluable guidance and supervision throughout this project. Their expertise in the field of AI/ML has been a continuous source of inspiration and motivation. I am deeply honoured to have had the privilege of working under their mentorship, as their profound knowledge has provided me with the necessary direction and resources to succeed.

I would like to extend my heartfelt appreciation to the faculty members and staff at CEERI Pilani for their consistent support and for creating a favourable research environment. Their invaluable assistance in facilitating access to resources and facilities has been highly appreciated and instrumental to the success of our project.

I would like to express my sincere gratitude to my fellow colleagues from BITS Pilani, for their valuable contributions to this project. Their support during brainstorming sessions and their insightful feedback have been immensely helpful and deeply appreciated.

Lastly, I extend my gratitude to the authors, researchers, and scholars whose works have been referenced and cited in this report. Their invaluable research and innovative ideas have served as the foundation for this project's development and implementation.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI(RAJASTHAN)

Practice School Division

Station: CEERI **Centre:** Pilani

Duration: 52 days **Date of Start:** 30th May

Date of Submission: 21st June

Title of the Project: Lesion localization and prioritization through computer vision techniques for point of care imaging devices

2021AAPS1971H RAKUL CHAUHAN B.E. ELECTRONICS AND COMMUNICATION ENGINEERING

2021A3PS2651P RISHIKA KALRA B.E. ELECTRICAL AND ELECTRONICS ENGINEERING

Name and Designation of the expert: DR. SATYAM SRIVASTAVA

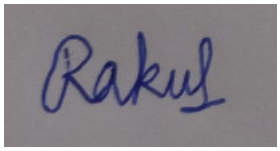
Senior Scientist, Integrated Circuits and Systems Group

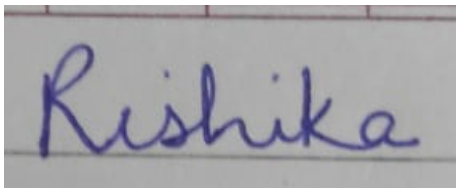
Name of the PS Faculty: DR. MEETHA V SHENOY

Key Words: Artificial Intelligence, Machine Learning, Computer Vision, Cervical cancer, YOLOv5

Project Areas: Lesion localization, Lesion prioritization, Object Identification, Classification

Abstract: Cervical cancer is one of the most spread cancers in the world. Its early detection using imaging techniques and computer vision algorithms can help a lot in saving later difficulties. The project involves working on lesion localization, prioritization and then some already used methods to identify the severe lesions and then finally implementing it on a GUI based real time imaging system. This would definitely help many underprivileged and military personnel and even one hand-held instrument can do much more detections than usual colposcopies and under much lower costs.





Signature(s) of Student(s) Signature of PS Faculty

Date 21/6/2023 Date

TABLE OF CONTENTS

INTRODUCTION

Cervical cancer is one of the most common cancers that women suffer from globally. Cervical cancer is a form of cancer that develops in the cells of the cervix, the lower part of the uterus connecting to the vagina. Its primary cause is persistent infection with certain strains of the human papillomavirus (HPV), a sexually transmitted infection. The progression of cervical cancer is typically slow, beginning with pre-cancerous changes called cervical intraepithelial neoplasia (CIN) or dysplasia. If left untreated, these abnormal cells can evolve into invasive cervical cancer. Common symptoms of cervical cancer include abnormal vaginal bleeding, such as bleeding between periods, after sexual intercourse, or post-menopause. Pelvic pain, discomfort during intercourse, and unusual vaginal discharge may also occur. Regular screening is crucial to detect and prevent cervical cancer. Tests like the Pap test and HPV testing are used to identify any abnormalities in cervical cells and detect high-risk HPV strains. Vaccination against HPV is recommended as a preventive measure.

Our project is based on cervical cancer detection using computer aided technology to ensure that military personnel who can't undergo annual Pap testing and other women in developing countries could undergo the cancer detection process without any hassle. Our project is divided into a series of tasks - the first one being object detection (cervix detection) using YOLOv5 from the images that a real time imaging system would capture. For this, we have to annotate and

prepare the dataset of raw images using Roboflow that we had been given followed by training on YOLOv5. Further on, our project involves classification with the help of a file which specifies each case's identification code and whether the case is positive, negative or suspicious of cancer. We have to label the images accordingly as given in the dataset using Roboflow followed by using YOLOv5 for classification. Further on, we have to localize the lesions followed by prioritizing them on the basis of size, shape and structure. Moreover, we have to classify the cancer, if found, as stage 1,2 and so on. Next, we will start working on replicating the DYSIS Map followed by bettering its algorithm and incorporating it in a real time imaging graphic user interface. The DYSIS map is a visual aid used in colposcopy to support the assessment of cervical abnormalities. It generates a color-coded representation of the cervix by analyzing digital colposcopy images. During a colposcopy exam, the application of acetic acid induces changes in the cervical tissue's appearance. The DYSIS map employs advanced image processing techniques to analyze the intensity and distribution of the resulting acetowhite reaction, where certain areas turn white. The DYSIS map divides the cervix into distinct color zones, each denoting a particular level of suspicion or potential severity of lesions. By providing a visual representation with color-coded zones, the DYSIS map assists colposcopists in focusing on specific areas that may require additional diagnostic procedures or biopsy. It promotes standardized interpretation of colposcopy findings and facilitates effective communication among healthcare professionals.

We started working on the project by reading a lot of research papers so as to understand what kind of algorithms we need to use for it, and also to understand the biological aspect, without which the project's learning would be incomplete and we would not be able to apply those algorithms.
Following this, we got ourselves comfortable with using YOLOv5, which is an open-source project, for detection and classification of various objects. We successfully identified the cervix with a good precision, followed by identification using videos because we had to incorporate this into a real time imaging system.
Simultaneously, we worked on the classification of images into positive, negative or suspicious images. This was followed by classification on videos , followed by color compression of images using k-means clustering to localize the lesions.
This was followed by trying to implement the DYSIS Map . We tried to follow some research papers to implement it . We started off with medical image registration , which is a process of aligning or matching multiple medical images of the same patient, for example in this case, the post-acetic acid image and the pre acetic-acid image of the cervix , to enable comparison, fusion and analysis. It involves coming across a spatial transformation that best fits for both the images. This was followed by opacity analysis of the post acetic image , by subtraction of the two images given, in the g channel of the RGB color space. This was further followed by segmentation of the regions in the cervix on the basis of intensity of white, as greater the intensity of white is, greater the chance of that region being cancerous.The segmentation was done according to a color map such as blue represented weak intensity of white, followed by green, followed by red, followed by yellow which was further followed by white, which detected the highest intensity of white in the acetowhite epithelium.

We are working on the API currently so that we can successfully integrate everything with a pre-made real time imaging graphic user interface.

1. OBJECT (here, cervix) IDENTIFICATION USING YOLOv5:

The first task in this project is to train a model for cervix detection and the software we used for this is Ultralytics YOLOv5, used under the framework of PyTorch and mainly popular for object detection for any custom data.

YOLO is short for You Look Only Once. It is a family of single-stage deep learning based object detectors. It is capable of more than real time object detection with state-of-the-art accuracy.

We followed the custom training codes available online and applied them on a Jupyter notebook, Google Colab. Google Colab (short for Colaboratory) is a hosted Jupyter notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs.

The training included some basic codes for importing libraries and defining some functions for later stages. After that, the training consisted of the following parts:

a) Dataset Preparation:

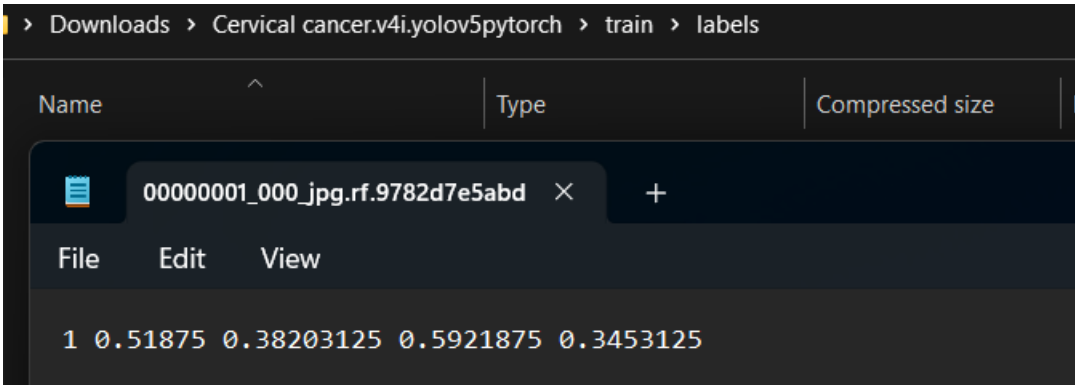
The dataset preparation was done using Roboflow, a software for uploading images or videos, annotating them, generating the dataset and then splitting it into train, valid and test parts.

For our project, we used 50 cervix images, 25 cat images and 25 dog images. This is usually not the best way to have data because the data should be similar for good training of the model, but it was okay for general object detection.

While saving and continuing, we were asked to decide in what ratio we would like to split the images into train, valid and test parts. The standard division is 70% train, 20% valid and 10% test, which we followed.

After this, we need to annotate the images. The annotation required bounding boxes around the main identification areas and assigning classes as cat, cervix, or dog.
Then, the annotated images are sent for generation, where it asks for some preferences in generating the version. After generating, we can download the dataset or just get a code snippet of it which has the API key unique to a version. The snippet code was copied and put in the custom training code.

The code created a directory of our dataset in different folders of test, train and valid, each having two parts, images and labels and a YAML file containing the paths to these. The labels are actually coordinates of the bounding boxes and are prefixed by 0,1 or 2 referring to cat, cervix or dog here. For example,



b) Training on YOLOv5:

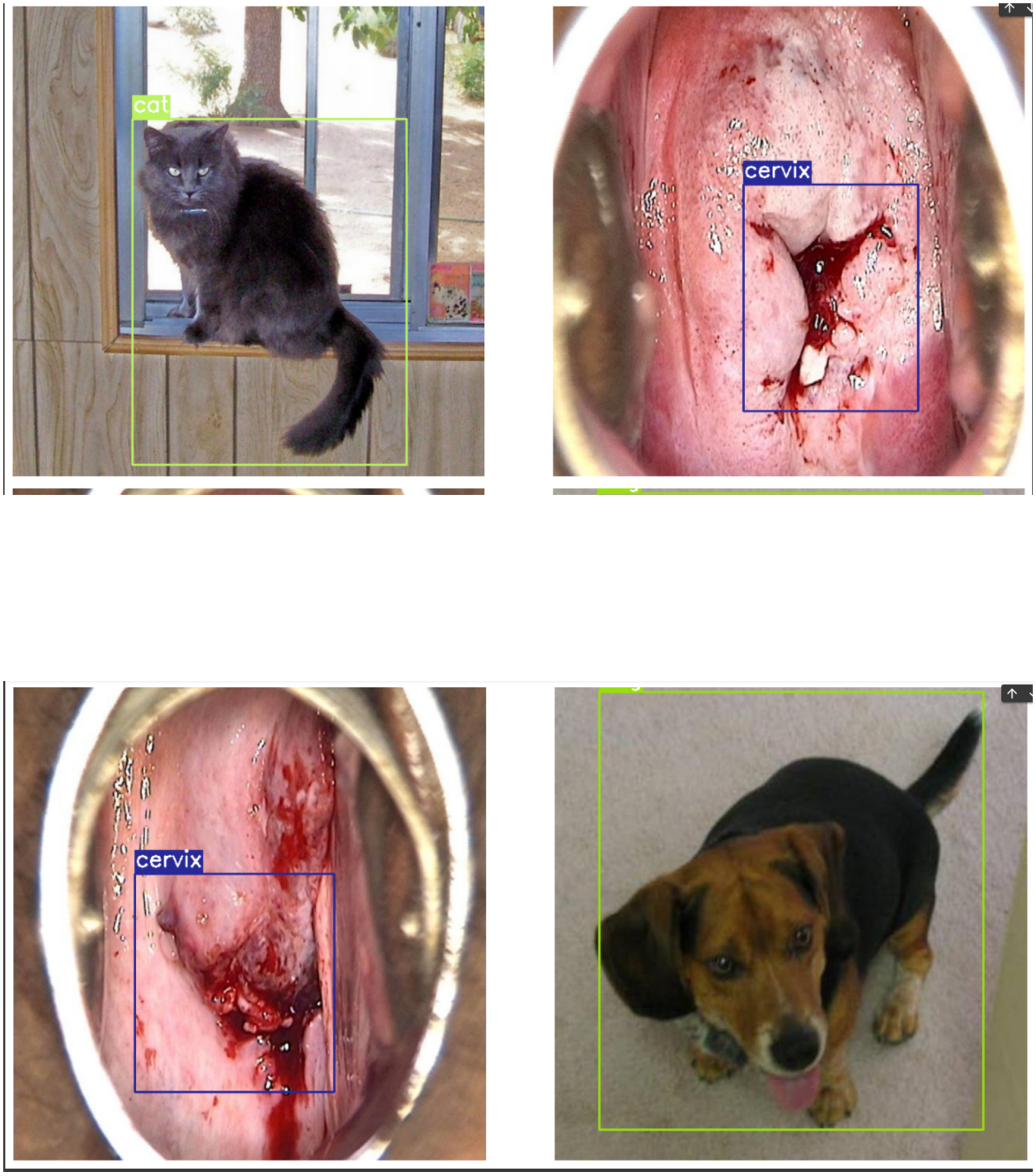
Firstly, we defined various functions like:

1. Converting bounding boxes in YOLO format to x and y coordinates

2. Adjusting the width and heights

3. Plotting images with the bounding boxes

4. Visualizing a few training images (here, we took 4 samples):



After this, we cloned the YOLOv5 repository and started training the model.

The training resulted in the following model summary:

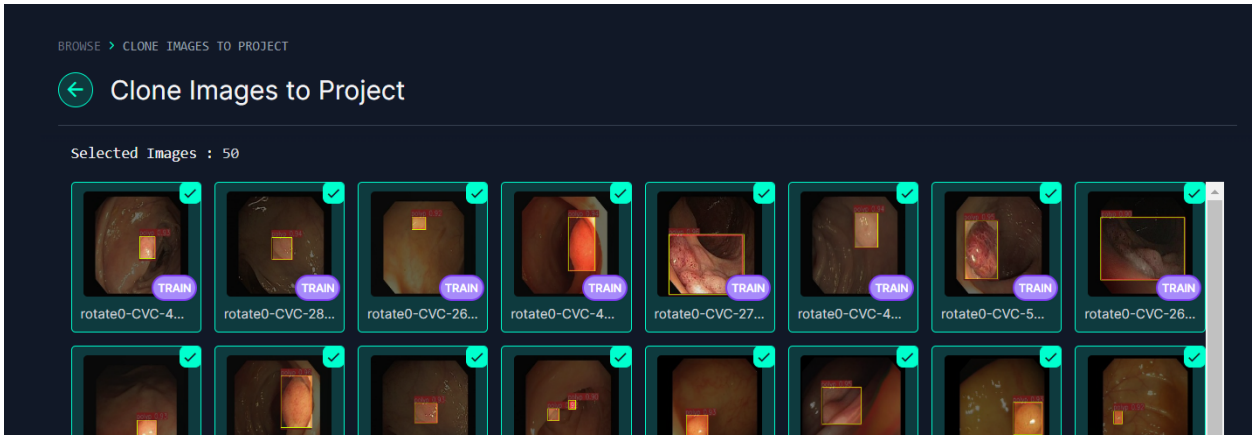
```
150 epochs completed in 1.735 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.3MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.3MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7018216 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 1/1 [00:03<00:00,  3.30s/it]
                   all         20         22      0.798      0.767      0.781      0.418
                   cat         20          8       0.94       0.75      0.785      0.427
                cervix         20         11      0.548      0.552      0.562      0.175
                   dog         20          3      0.906          1      0.995      0.653
Results saved to runs/train/exp
```

The model was trained with a mAP (mean average precision) of 78.1% for 0.5 IoU (Intersection over Union) and 41.8% for 0.5:0.95 IoU.

c) Validating and inferencing images and videos:

We are working on this part where we validate and infer data from other datasets and check whether the model can detect and automatically form bounding boxes in videos also.

A better dataset: The dataset which we were using had cats and dogs other than cervix images. This isn't a very dataset as it should contain objects similar to the one we are detecting. As a result, we searched for body organs or cancer related images on Kaggle and other platforms. Finally found oral cancer(abnormal and normal) and skin cancer(polyp) images on Roboflow universe which were already annotated and just needed to be cloned.



So our new dataset had 154 cervix images, 70 polyp and 70 oral cancer images. This made it a 294 images dataset. We ran it for 75 epochs and got the following results.

```
75 epochs completed in 2.279 hours.
Optimizer stripped from runs/train/results_1/weights/last.pt, 14.5MB
Optimizer stripped from runs/train/results_1/weights/best.pt, 14.5MB

Validating runs/train/results_1/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7018216 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 1/1 [00:09<00:00,  9.59s/it]
                   all         20         21      0.959      0.267      0.299      0.132
              Abnormal         20          2          1          0     0.0085      0.001
                cervix         20         15      0.878        0.8      0.875      0.392
                 polyp         20          4          1          0     0.0146    0.00261
Results saved to runs/train/results_1
```

This had very good precision values but the mAP values were poor. As a result, we had to increase the no. of epochs as it might increase the accuracy. While running the Colab notebook on 300 epochs, it took a huge amount of time and got disconnected before completion.

So, next we started working on VSCode as it works offline. The same code when run on VSCode , gave multiple errors, especially the index out of range errors, which couldn't be resolved easily.

Got to know about the GPU runtime feature in Colab, it increased the training speed a lot and 300 epochs could now be done in under half an hour if there is no disruption. Again, the accuracy wasn't very good, so changed the dataset by including 50 images each of polyp and oral cancer making a total of 374 images.

```
400 epochs completed in 0.689 hours.
Optimizer stripped from runs/train/results_1/weights/last.pt, 14.5MB
Optimizer stripped from runs/train/results_1/weights/best.pt, 14.5MB

Validating runs/train/results_1/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7020913 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 2/2 [00:01<00:00,  1.67it/s]
                   all         52         52      0.876      0.631      0.714      0.449
              Abnormal         52          6      0.782      0.167      0.227      0.142
                cervix         52         23      0.877      0.619      0.805       0.33
                normal         52          4       0.85          1      0.995       0.65
                 polyp         52         19      0.993      0.737      0.827      0.673
Results saved to runs/train/results_1
```

As this accuracy was pretty good (except for the abnormal class), we started working on the inference part.

This part gave several errors due to wrong file path or drive zipping and unzipping issues. After a while , it worked and it ran correctly both on images and videos. However, this was the case when there was no confidence threshold applied. So, the model was even detecting a video of a dog as an abnormal class. So, next we went through the README file to understand where the threshold parameters are defined in the script and applied a threshold of 0.5. Now, the model was unable to detect all the videos properly. We increased the number of epochs and even used a new pre-annotated dataset but it hardly improved.

Here's an example of a inference video of treating the cervical region :

https://drive.google.com/file/d/1wQCHEKlWGfJjwXQu3QeKmAB7e1YDnOP9/view?usp=sharing

We kept working on this with changes in datasets and labels but this is the best we got.

2. CLASSIFICATION OF CERVIX IMAGES AS POSITIVE, NEGATIVE OR CANCEROUS USING YOLOv5:

The next task was to train a model using YOLOv5 for classification of cervix images as positive, negative or cancerous. We were given an excel sheet which contained the data about images and their characteristics, i.e., positive, negative or cancerous. So, we had to use this data to train a YOLOv5 model for classification.

Again, the first few steps include setting up the environment and installing the dependencies. After that, the following steps were done:

a) Dataset Preparation on Roboflow:

In this case, we used single label classification instead of bounding boxes for our project in Roboflow. Then we uploaded the images, which were in 186 folders, each having 2 or 3 images, thus 417 images in total. The first 92 folders had negative ones, 93 to 166 had positive and 167 to 186 were cancerous.

There is this feature in Roboflow that if all the images having a certain label (e.g., negative) are put under a folder named with the same label, then if we upload that folder, all the files inside that folder are automatically labelled as negative. So, we tried to do this as we had quite a big dataset and labelling each one of them would have been a tedious task. But we faced problems in this. Firstly, the folders we formed already had folders inside them, so we had to extract all files. We did this, but even after doing it multiple times, there was some issue with the

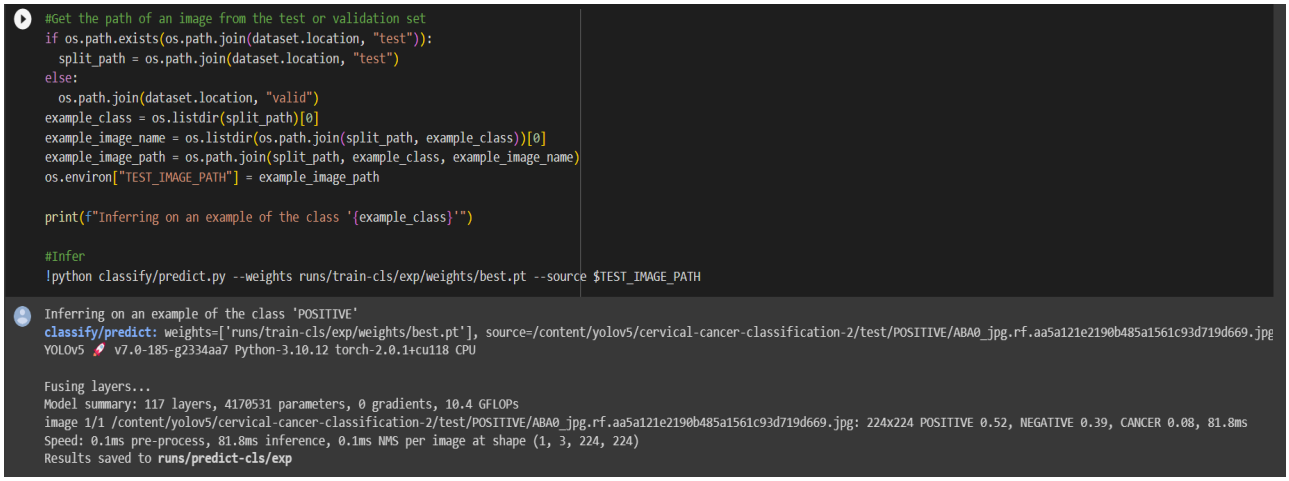
Roboflow assignment and so we could not get automatic labels. As a result, we uploaded all the images and labelled them one by one by matching with that excel sheet.

After annotation is done, we generated the dataset and copied the code snippet formed and pasted it in the Colab notebook.

b) Training YOLOv5:

Firstly, we downloaded some pre-trained weights so that the training process is accelerated and the results quality is good.

Then, we started with the training and got the following results.

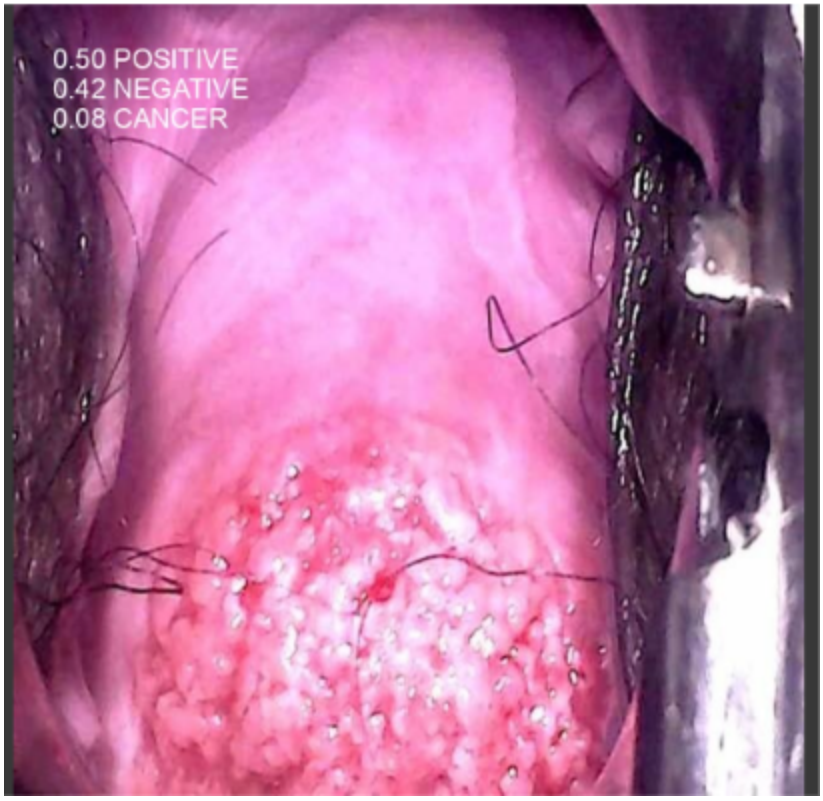


```
#Get the path of an image from the test or validation set
if os.path.exists(os.path.join(dataset.location, "test")):
  split_path = os.path.join(dataset.location, "test")
else:
  os.path.join(dataset.location, "valid")
example_class = os.listdir(split_path)[0]
example_image_name = os.listdir(os.path.join(split_path, example_class))[0]
example_image_path = os.path.join(split_path, example_class, example_image_name)
os.environ["TEST_IMAGE_PATH"] = example_image_path

print(f"Inferring on an example of the class '{example_class}'")

#Infer
!python classify/predict.py --weights runs/train-cls/exp/weights/best.pt --source $TEST_IMAGE_PATH
```

```
Inferring on an example of the class 'POSITIVE'
classify/predict: weights=['runs/train-cls/exp/weights/best.pt'], source=/content/yolov5/cervical-cancer-classification-2/test/POSITIVE/ABA0_jpg.rf.aa5a121e2190b485a1561c93d719d669.jpg
YOLOv5 🚀 v7.0-185-g2334aa7 Python-3.10.12 torch-2.0.1+cu118 CPU

Fusing layers...
Model summary: 117 layers, 4170531 parameters, 0 gradients, 10.4 GFLOPs
image 1/1 /content/yolov5/cervical-cancer-classification-2/test/POSITIVE/ABA0_jpg.rf.aa5a121e2190b485a1561c93d719d669.jpg: 224x224 POSITIVE 0.52, NEGATIVE 0.39, CANCER 0.08, 81.8ms
Speed: 0.1ms pre-process, 81.8ms inference, 0.1ms NMS per image at shape (1, 3, 224, 224)
Results saved to runs/predict-cls/exp
```

While inferring an example of the class ‘POSITIVE’ , it showed 52% positive, 39% negative and 8% suspicious of cancer.

Here’s an example of testing on an image:



c) Inferencing on videos:

This part required writing the same predict.py code line as above but with the source path changed to ‘video_path’ –view-img.

```
!python classify/predict.py --weights runs/train-cls/exp/weights/best.pt --source '/content/yolov5/movi0 (2).mp4' --view-img

WARNING ⚠ 'ultralytics.yolo.v8' is deprecated since '8.0.136' and will be removed in '8.1.0'. Please use 'ultralytics.models.yolo' instead.
WARNING ⚠ 'ultralytics.yolo.utils' is deprecated since '8.0.136' and will be removed in '8.1.0'. Please use 'ultralytics.utils' instead.
Note this warning may be related to loading older models. You can update your model to current structure with:
    import torch
    ckpt = torch.load("model.pt")  # applies to both official and custom models
    torch.save(ckpt, "updated-model.pt")

classify/predict: weights=['runs/train-cls/exp/weights/best.pt'], source=/content/yolov5/movi0 (2).mp4, data=data/coco128.yaml, imgsz=[224, 224],
YOLOv5 🚀 v7.0-193-g485da42 Python-3.10.6 torch-2.0.1+cu118 CPU

Fusing layers...
Model summary: 117 layers, 4170531 parameters, 0 gradients, 10.4 GFLOPs
qt.qpa.xcb: could not connect to display
qt.qpa.plugin: Could not load the Qt platform plugin "xcb" in "/usr/local/lib/python3.10/dist-packages/cv2/qt/plugins" even though it was found.
This application failed to start because no Qt platform plugin could be initialized. Reinstalling the application may fix this problem.

Available platform plugins are: xcb.
```

We faced this error multiple times and could not be resolved. (Note that this is not because of the space in the video path given as any path given in single quotes doesn’t matter).

As a result , we found another way to classify the images. We got a different dataset which had the labels of type 1,2 and 3 referring to the stages CIN 1,2 and 3 respectively. We ran the dataset on the object detection model and got the following result.

https://drive.google.com/file/d/1HCPl_VRngWcslm8fWVOnykapDIjgK8pj/view?usp=sharing

Here in this video, 0 refers to type 1 , 1 to type 2 and 2 to type 3 of cervix images.

3. LESION LOCALIZATION:

The next main task is to localize the lesion present in the cervix. A lesion is an area of abnormal tissue, which may be benign (not cancer) or malignant(cancer).

By lesion localization, we mean the identification of the exact location of abnormal or potentially cancerous tissues. There are few methods to do that, including:

a) Colposcopy: It involves visual inspection of the cervix using a colposcope, a magnifying instrument with a light source.

b) Biopsy: If an abnormal area is identified during colposcopy, it can be taken for biopsy where a pathologist performs a precise diagnosis of it.

c) Imaging techniques: Sometimes, if the lesions are larger in size, ultrasonic or magnetic resonance imaging (MRI) can also be used to locate the lesions.

4. LESION PRIORITIZATION:

The next task is to prioritize the lesions found in the order of their severeness. This can be done on the basis of size, structure, colors, etc. Other than that, we can also prioritize on the basis of the following parameters:

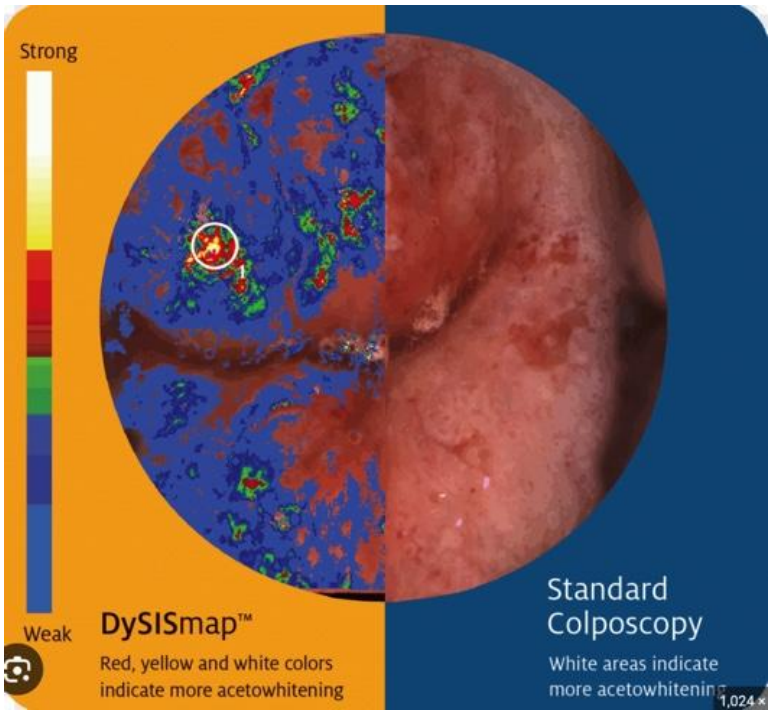
a) Cervical Intraepithelial Neoplasia (CIN) grades: These grades help in estimating the severity of the lesion based on the cellular changes observed. CIN 1 is considered mild, CIN2 is moderate and CIN3 is severe. CIN2 and CIN3 are considered to have more chances to convert into cancer later on.

b) High-Risk Human Papillomavirus (HPV) Types: This virus is considered the primary cause for cervical cancer and thus if the lesion is found to be having high levels of HPV, then it can be prioritized.

c) Patient's history and biodata: Factors like age, reproductive plans, previous surgeries (if any), etc. help in understanding the priority of the lesions.

These two tasks are quite important and we will be doing these with the help of DYSIS Map implementation.

5. DYSIS MAP IMPLEMENTATION:



Cervical cancer is one of the most common cancers among women globally, and it is the primary cause of most cancer-related deaths in developing countries. However, when detected early and treated effectively, cervical cancer can be almost entirely cured. By utilizing visual inspection exams, doctors can identify specific morphologic features in cervical precursor lesions and invasive cancer. With the aid of digital imaging technologies, physicians can enhance their diagnostic process by incorporating Computer-Aided Diagnosis (CAD) systems.

During colposcopy, when acetic acid is applied, the epithelium that turns white is referred to as acetowhite epithelium. This observation plays a significant role in detecting cancer and precancerous areas. Colposcopists assess the potential severity of lesions by examining the color and density of the acetowhite reaction. Abnormal acetowhite epithelium can range from a subtle or intense white to a dense, grayish-white shade. Studies have shown that the degree of whiteness is directly related to the severity of Cervical Intraepithelial Neoplasia (CIN). Moreover, the opacity or translucency of the acetowhite reaction varies across the spectrum of CIN: normal metaplasia and low-grade lesions typically exhibit a faint white or slightly translucent appearance, while high-grade lesions display a more opaque acetowhite reaction. However, automatically extracting acetowhite regions from cervical images is quite challenging as it involves dealing with issues such as specular reflection, diverse illumination conditions, and notably, substantial variations within patients.

A computerized image analysis system for uterine cervical images is capable of examining and extracting diagnostic characteristics from these images. It can provide physicians with suggested clinical diagnoses, aiding in the diagnostic process. This system can also be integrated with a medical screening device, enabling non-medical personnel to conduct cervical cancer screening.

This technology holds promise for various applications, including screening for female military personnel stationed in areas where regular Pap testing is not feasible. Additionally, it has the potential to greatly benefit underserved women in developing nations by facilitating cervical cancer screening.

Yang et al. and his research team from Texas Tech University devised a sophisticated method that combines K-means clustering and deterministic annealing techniques to accurately detect acetowhite regions. It aimed to overcome challenges posed by illumination effects and significant intra-patient variation. Gordon and his colleagues from Tel-Aviv University developed an unsupervised segmentation algorithm that employs a Gaussian mixture model to identify three tissue types in cervical imagery.

In their latest study, Gordon and his co-workers focused on extracting the acetowhite region by isolating the cluster with the highest mean intensity among the smooth regions. They acknowledged that incorrect detection of acetowhite lesions can occur due to illumination variations and that lesions in shaded areas of the image may go undetected. It is important to note that the mentioned research works were conducted using CervigramTM images collected by the National Cancer Institute (NCI).

The concept of using visual mapping inS colposcopy examinations can be traced back to the early 2000s. Researchers and medical professionals recognized the need for a standardized and objective approach to assess cervical abnormalities and distinguish between different grades of lesions.

The DYSIS map was introduced by DySIS Medical Ltd., a medical technology company which specializes in women's health. The initial version of the DYSIS map was based on the analysis of digital colposcopy images using proprietary algorithms. This early version provided a visual representation of the cervix and highlighted the areas which were concerning. The usage of advanced imaging technologies such as cross-polarization and advanced image analysis algorithms, have increased the accuracy and reliability of the DYSIS map.

There has been research and clinical studies to prove the effectiveness of the DYSIS map in improving colposcopy evaluations and lesion detection. These studies have focused on the potential of the DYSIS map in assisting colposcopists and improving the diagnostic process for cervical abnormalities and proving to be a boon to women in developing countries.

The DYSIS map holds a lot of promise for further advancements in the field of colposcopy. It has the potential to revolutionize cervical examinations by providing more objective and standardized information to guide healthcare professionals in the diagnosis and management of cervical abnormalities. The DYSIS map makes use of advanced image processing algorithms to assess the acetowhite reaction, which is the color change observed in the cervical epithelium after the application of acetic acid. The DYSIS map helps colposcopists identify and differentiate between normal, low-grade, and high-grade lesions by quantifying the intensity and the distribution of the acetowhite epithelium.

The map is divided into color zones, each representing a different level of suspicion. Typically, green or blue zones indicate normal or low-grade areas, while yellow, orange, or red zones signify regions of potential concern or high-grade lesions. This color-coded representation enables healthcare professionals to focus on specific areas that require closer examination or biopsy.

We started working on the DYSIS map by following some research papers .
It mainly consisted of the following two parts.

a) **MEDICAL IMAGE REGISTRATION**: The first step was to do medical image registration , which is a critical process which involves aligning or matching multiple images of the same patient or the anatomy for comparison and analysis . The goal is to find such a spatial transformation that best aligns the images.

   Simple ITK is a software library designed to facilitate medical image analysis and processing . We used the optical flow technique to estimate the apparent motion of the two images given , the pre-acetic acid image and the post-acetic acid image. We loaded the pre-acetic acid image as the moving image while we loaded the post-acetic acid image as the reference image. We loaded them using OpenCVs ‘cv2.imread’ function , followed by converting them to grayscale using the ‘cv2.cvtColor’ to simplify the optical flow calculation. Optical flow, which captures the displacement vectors for each pixel between the reference and the moving images, is calculated using the Farneback method , a well known optical flow algorithm, using the ‘cv2.calcOpticalFlowFarneback’.

   A new image called ‘warped_image’ , is created using the same dimensions as the reference image . It serves as a canvas for the registered image. The calculated flow field is used to warp the moving image and then align it with the reference image. Each pixel in the reference image is mapped to its corresponding location in the moving image based on the displacement vectors from the flow field. The pixel value from the moving image is then assigned to the corresponding pixel in the ‘warped_image’. Finally, the resulting registered image, ‘warped_image’ , is saved to the specified output path using the ‘cv2.imwrite’ as can be seen in the code below .

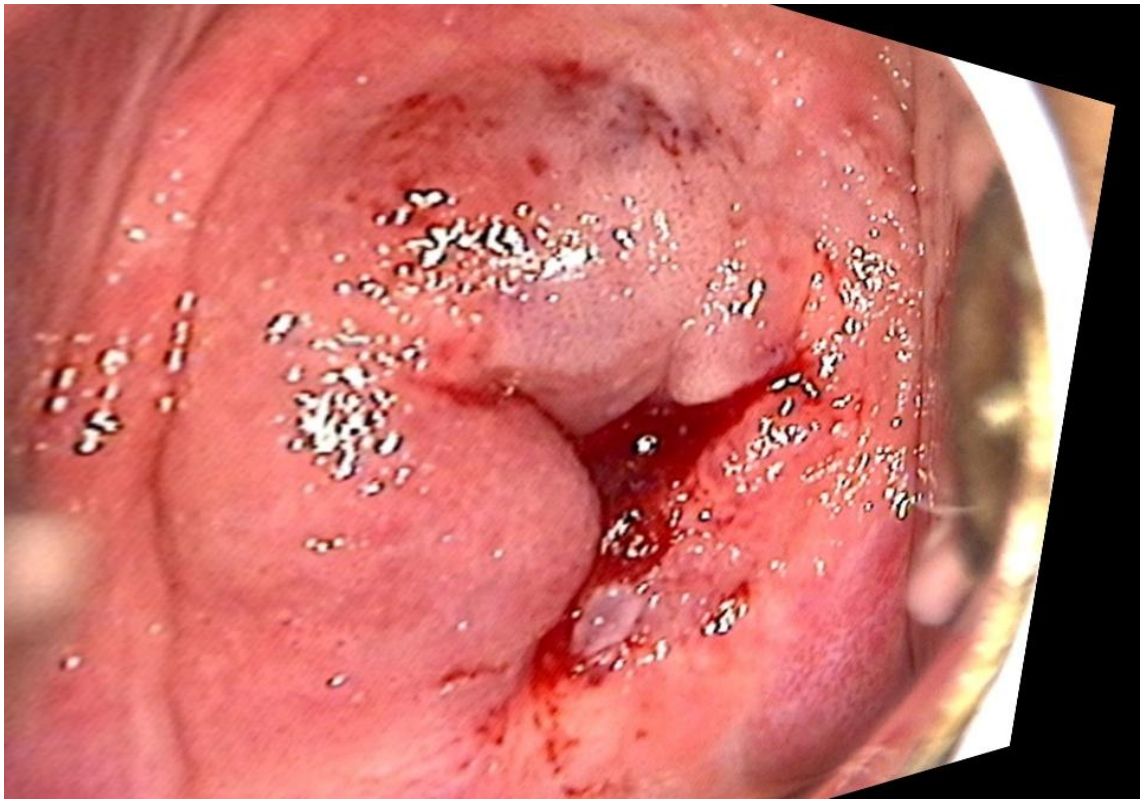
```python
def register_images(ref_image_path, moving_image_path, output_path):
    # Load the fixed and moving images
    ref_image = cv2.imread(ref_image_path)  # Load as color
    moving_image = cv2.imread(moving_image_path)  # Load as color

    # Convert the images to grayscale for optical flow calculation
    ref_image_gray = cv2.cvtColor(ref_image, cv2.COLOR_BGR2GRAY)
    moving_image_gray = cv2.cvtColor(moving_image, cv2.COLOR_BGR2GRAY)

    # Calculate the optical flow using Farneback method
    flow = cv2.calcOpticalFlowFarneback(ref_image_gray, moving_image_gray, None, 0.5, 3, 15, 3, 5, 1.2, 0)

    # Warp the moving image using the flow field
    rows, cols, _ = ref_image.shape
    warped_image = np.zeros_like(ref_image)
    for i in range(rows):
        for j in range(cols):
            dx, dy = flow[i, j]

            # Avoid out-of-bounds indices
            warped_i = min(max(int(i+dy), 0), rows-1)
            warped_j = min(max(int(j+dx), 0), cols-1)

            warped_image[warped_i, warped_j] = moving_image[i, j]

    # Save the registered image to the output path
    cv2.imwrite(output_path, warped_image)
```

We also tried detecting the key points and computing the descriptors using the ORB feature detector for both images. We created a matcher using the Brute Force matching algorithm with the Hamming distance metric. We matched the descriptors of the moving image with the descriptors of the reference image. The matches are sorted on the basis of their Hamming distance. We selected the top 90% of matches (best matches) . We displayed the 100 best images. We extracted the corresponding keypoint coordinates for the selected image. We found the homography matrix using the RANSAC algorithm, which estimates the geometric transformation between the keypoint matches. We further warped the moving image using the obtained homography matrix to align it with the reference image. We saved the registered image to the specific output path.

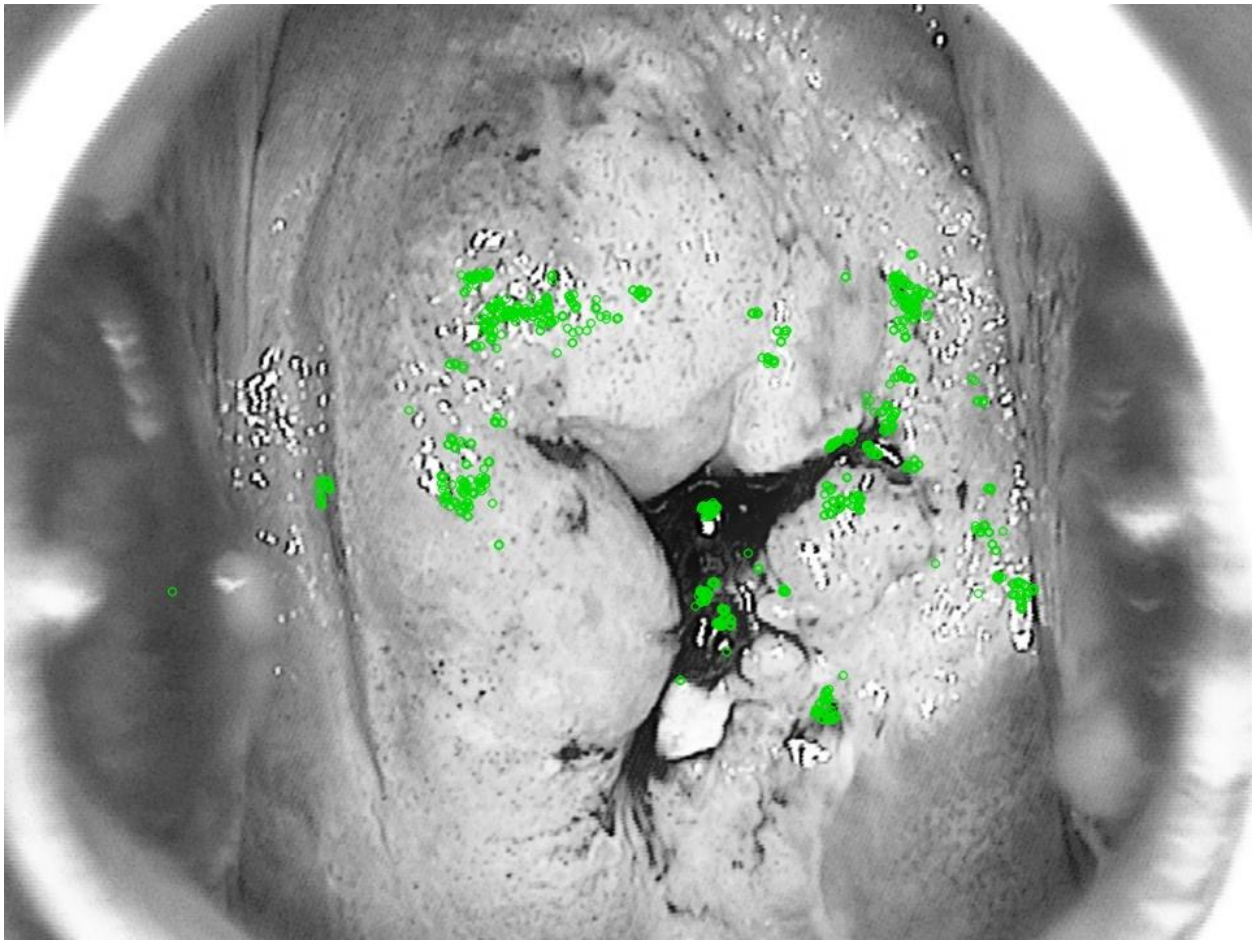
```python
def register(ref_image_path, moving_image_path, output_path):
    # Load the fixed and moving images
    ref_image = cv2.imread(ref_image_path)  # Load as color
    moving_image = cv2.imread(moving_image_path)  # Load as color

    # Convert the images to grayscale for optical flow calculation
    img1 = cv2.cvtColor(ref_image, cv2.COLOR_BGR2GRAY)
    img2 = cv2.cvtColor(moving_image, cv2.COLOR_BGR2GRAY)

    # Warp the moving image using the flow field
    height,width = img1.shape

    orb_detector = cv2.ORB_create(800)

    # Extract key points and descriptors for both images
    kp1, des1 = orb_detector.detectAndCompute(img2, None)
    kp2, des2 = orb_detector.detectAndCompute(img1, None)

    # Display keypoints for reference image in green color
    imgKp_Ref = cv2.drawKeypoints(img1, kp1, 0, (0,222,0), None)
    cv2.imwrite('/content/keypoints.jpg', imgKp_Ref)

    matcher = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

    # Match the two sets of descriptors.
    matches = matcher.match(des1, des2)

    # Sort matches on the basis of their Hamming distance.
    matches = sorted(matches, key=lambda x: x.distance)

    # Take the top 90 % matches forward.
    matches = matches[:int(len(matches) * 0.9)]
    no_of_matches = len(matches)
    print(len(matches))

    # Display only 100 best matches {good[:100}
    imgMatch = cv2.drawMatches(img2, kp2, img1, kp1, matches[:100], None, flags = 2)
    cv2.imwrite('/content/matches.jpg', imgMatch)
```

```
# Define empty matrices of shape no_of_matches * 2.
p1 = np.zeros((no_of_matches, 2))
p2 = np.zeros((no_of_matches, 2))

for i in range(len(matches)):
  p1[i, :] = kp1[matches[i].queryIdx].pt
  p2[i, :] = kp2[matches[i].trainIdx].pt

# Find the homography matrix.
homography, mask = cv2.findHomography(p1, p2, cv2.RANSAC)

# Use this matrix to transform the
# colored image wrt the reference image.
warped_img = cv2.warpPerspective(moving_image,
                homography, (width, height))

# Save the registered image to the output path
cv2.imwrite(output_path, warped_img)
```

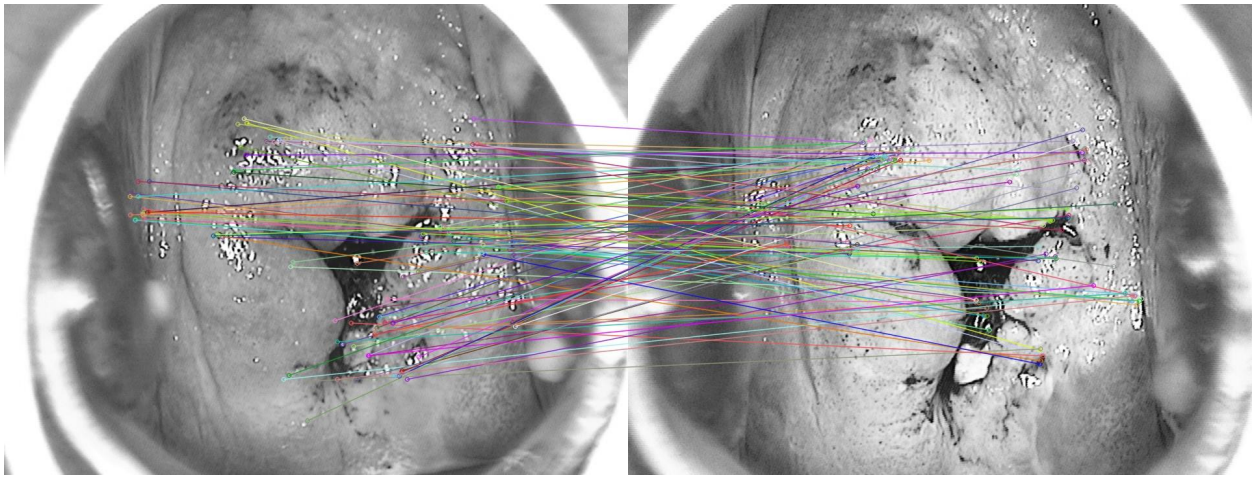
The results for the same were as follows:



aligned.jpg



keypoints.jpg



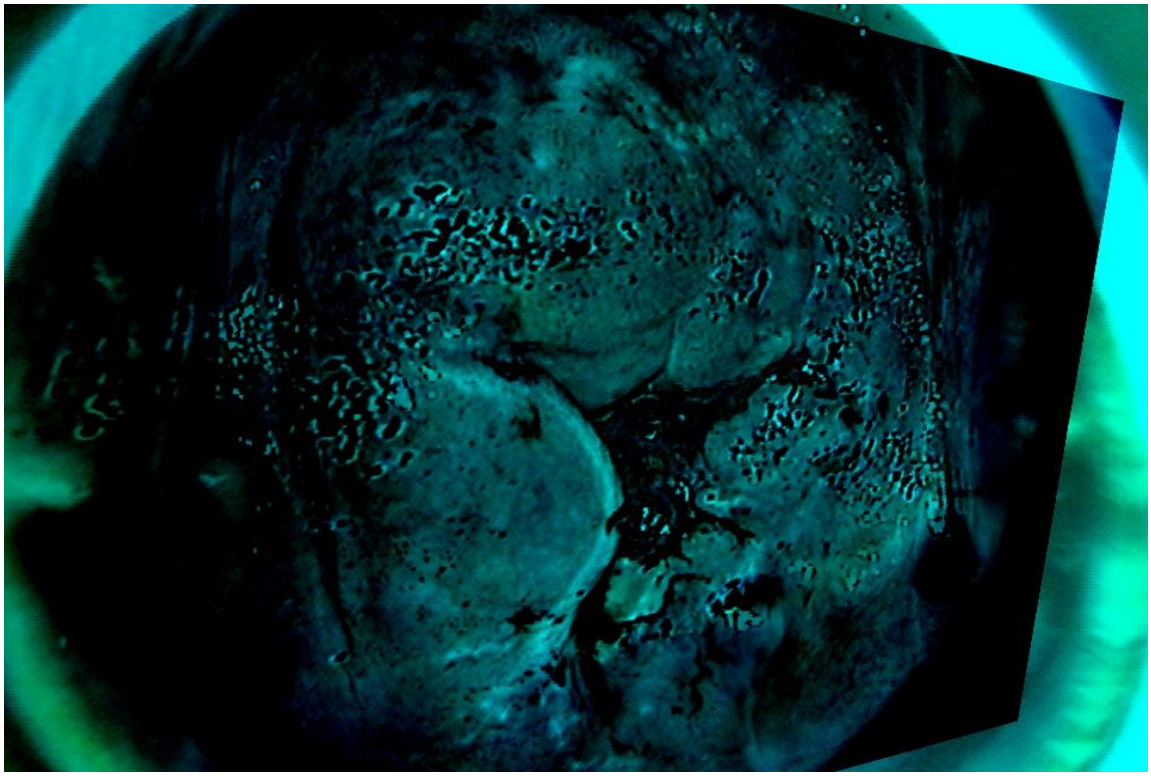
matches.jpg

b) **OPACITY ANALYSIS**: This was followed by opacity analysis by subtracting the post-acetic acid image from the pre-acetic acid image in the g channel of RGB space .

We started off with loading the input images , followed by splitting them into their individual color channels (blue, green and red). We subtracted the corresponding blue channels of the two images to calculate the difference in the blue channel , followed by subtraction in the corresponding green channels of the two images to calculate the difference in the green channel.
This was followed by creating a new image by combining the difference in the blue channel and the difference in the green channel. Following this , we set the red channel of the new image to all zeroes. The resulting image was saved to the specified output path. This code performed a simple opacity analysis by comparing the blue and green channels between the two images. By subtracting the corresponding channels , it highlights the difference in these channels. This analysis is useful in detecting changes in color and transparency between the two images.

```
import cv2
import numpy as np

def opacity_analysis(image1_path, image2_path, output_path):
    # Load the images
    image1 = cv2.imread(image1_path)
    image2 = cv2.imread(image2_path)

    # Split the images into channels
    b1, g1, r1 = cv2.split(image1)
    b2, g2, r2 = cv2.split(image2)

    # Subtract the blue and green channels
    bg_diff = cv2.subtract(b1, b2)
    gg_diff = cv2.subtract(g1, g2)

    # Create a new image with blue and green channels
    result = cv2.merge((bg_diff, gg_diff, np.zeros_like(r1)))

    # Save the resulting image
    cv2.imwrite(output_path, result)


# Usage example
image1_path = '/content/AEB1.jpg'
image2_path = '/content/aligned.jpg'
output_path = '/content/outputopacity.jpg'
opacity_analysis(image1_path, image2_path, output_path)
```

The results of the same were as follows:



outputopacity.jpg

The green color as shown in the output opacity image indicates the intensity and the affected area of the acetowhite epithelium ,as was required by the DYSIS map.

Following this, we worked on the DYSIS map , which involved converting the image to grayscale, which was followed by defining different pixel ranges and corresponding grouped values for the grayscale image. Created a copy of the grayscale image and assigned grouped values based on the pixel ranges. Defined color mappings for different grayscale values. Created an empty color image with the same shape as the grayscale image. Iterated over each pixel of the grouped image and assigned color based on the grayscale value using the defined color mappings .

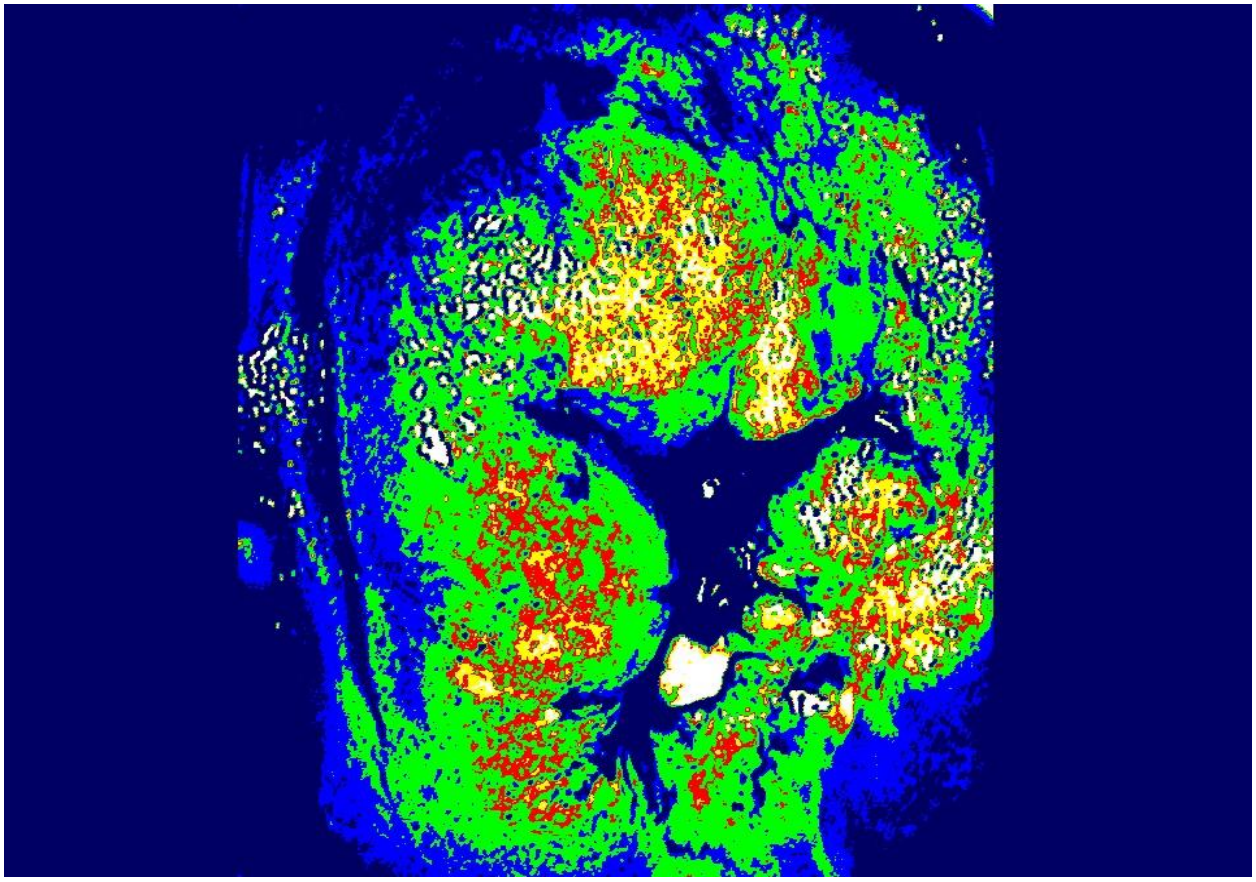
Saved the resulting color image as can be seen in the code given below.

```
from google.colab.patches import cv2_imshow
import cv2
import numpy as np
import matplotlib.pyplot as plt

def acetowhite_opacity_analysis(image):

# Convert the image to grayscale
```

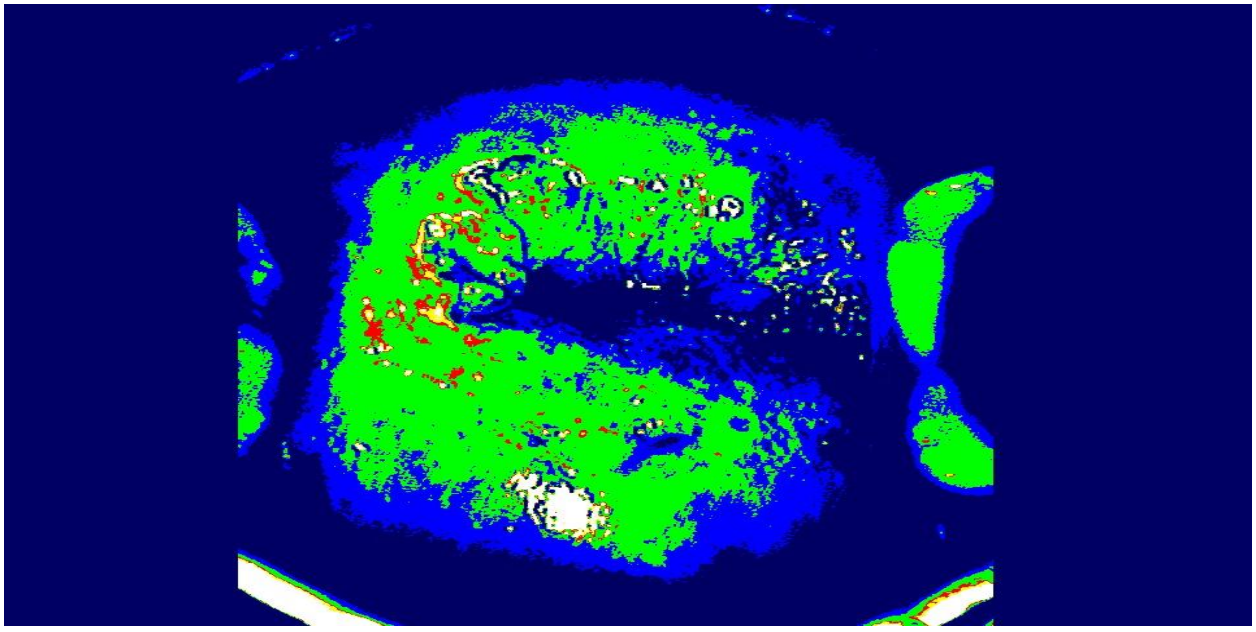
```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
print(gray.shape)
pixel_ranges = [(0, 150), (150, 180), (180, 200),(200,210),(210, 225),
(225, 255)]
grouped_values = [0, 80,130,170, 210, 255]
grouped_image = np.copy(gray)

for i in range(len(pixel_ranges)):
lower = pixel_ranges[i][0]
upper = pixel_ranges[i][1]
grouped_value = grouped_values[i]
mask = cv2.inRange(gray, lower, upper)
grouped_image[np.where(mask>0)] = grouped_value

color_mappings = {
0: [100, 0, 0], # Mapping for grayscale value 0 - Black
80: [255, 0, 0], # Mapping for grayscale value 1 - Blue
130: [0, 255, 0] , # Mapping for grayscale value 2 - Green
170: [0, 0, 255], # Mapping for grayscale value 3 - Yellow
210: [0, 255, 255], # Mapping for grayscale value 4 - Red
255: [255, 255, 255] # Mapping for grayscale value 4 - White
}

color_image = np.zeros((gray.shape[0], gray.shape[1], 3), dtype=np.uint8)

# Apply color coding based on grayscale values
for i in range(grouped_image.shape[0]):
for j in range(gray.shape[1]):
if j<180 or j>760:
color_image[i, j] = [100, 0, 0]
continue
grayscale_value = grouped_image[i, j]
color = color_mappings.get(grayscale_value, [0, 0, 0])
color_image[i, j] = color
return color_image

# Specify the path to your cervical image
image_path = '/content/AEB1.jpg'
image = cv2.imread(image_path)
# Perform acetowhite opacity analysis
```

```
color_image = acetowhite_opacity_analysis(image)
cv2.imwrite('/content/dysis.jpg',color_image)
```

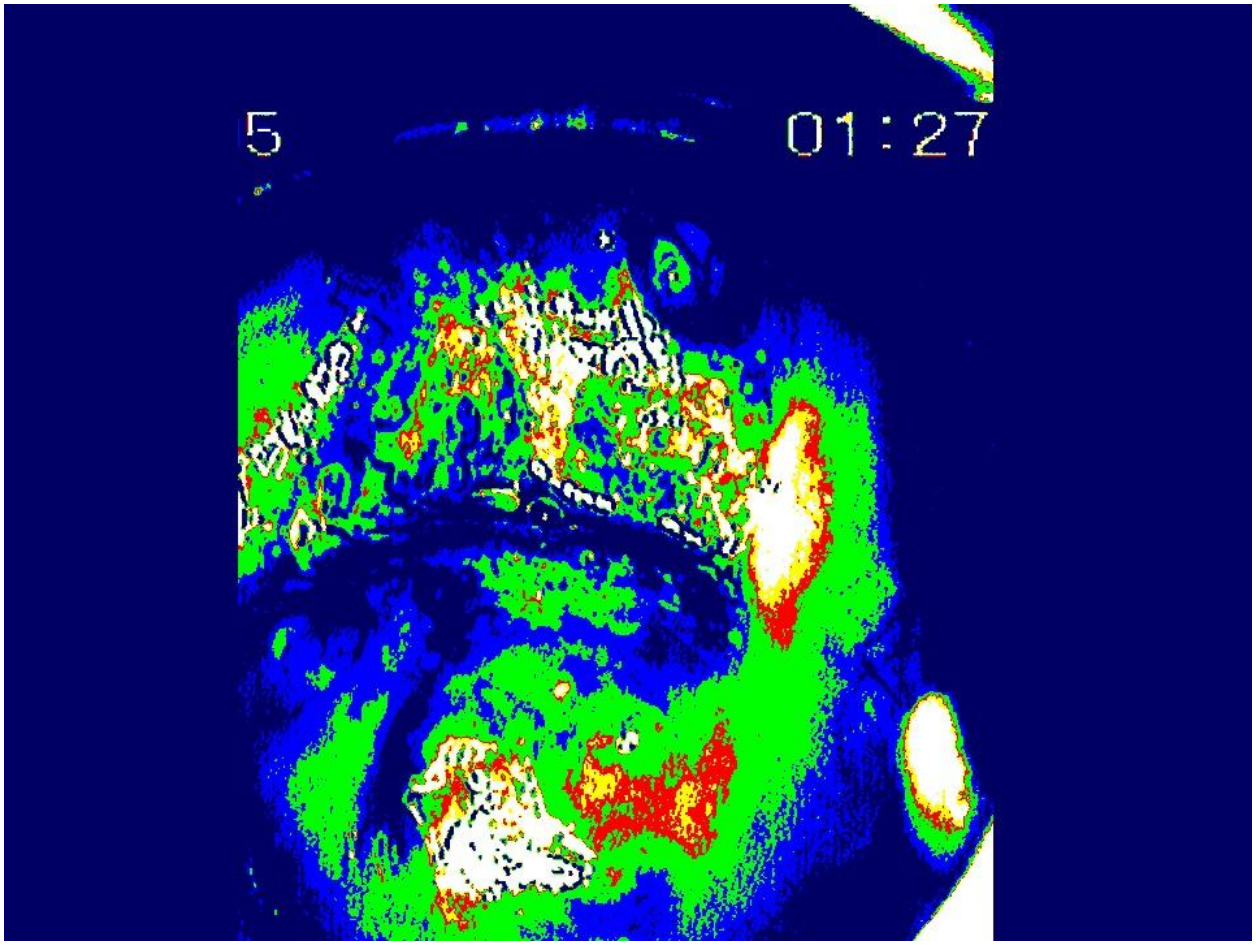
The results of the same were as follows for positive , negative and suspicious of cancer images:



SUSPICIOUS IMAGE



NEGATIVE IMAGE



POSITIVE IMAGE

As can be understood from the images , the DYSIS map is working correctly . According to the color coding order followed by the DYSIS map , the higher the intensity of white , the colors are mapped as follows : blue(weak), green,red, yellow, white (strong).
As can be seen in the negative image , we see very less amount of yellows and whites , which are an indication of cancer , in the DYSIS map generated. For the positive image , we see a lot of yellows and whites while for the suspicious image , we see all sorts of colors .

c) VIDEO INFERENCING:

Next we applied this algorithm to videos.
Here's the code we used for it.

```
import cv2
import numpy as np

def acetowhite_opacity_analysis(video_path):
    # Open the video file
    video = cv2.VideoCapture(video_path)

    # Get video properties
    frame_width = int(video.get(cv2.CAP_PROP_FRAME_WIDTH))
    frame_height = int(video.get(cv2.CAP_PROP_FRAME_HEIGHT))
    frame_count = int(video.get(cv2.CAP_PROP_FRAME_COUNT))
    fps = video.get(cv2.CAP_PROP_FPS)

    # Calculate the aspect ratio of the input video
    aspect_ratio = frame_width / frame_height

    # Set the output video's frame width and height based on the aspect ratio
    output_width = 400  # Adjust this value as needed
    output_height = int(output_width / aspect_ratio)

    # Create an output video writer
    output_path = '/content/drive/MyDrive/output.mp4'
    output_video = cv2.VideoWriter(output_path, cv2.VideoWriter_fourcc(*'XVID'), fps, (output_width, output_height))

    # Process each frame of the video
    for frame_index in range(frame_count):
        # Read the current frame
        ret, frame = video.read()
        if not ret:
```

```
        if not ret:
            break

        # Convert the frame to grayscale
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Perform acetowhite opacity analysis on the grayscale frame
        pixel_ranges = [(0, 150), (150, 180), (180, 215), (215, 225), (225, 235), (235, 255)]
        grouped_values = [0, 80, 130, 170, 210, 255]
        grouped_image = np.copy(gray)

        for i in range(len(pixel_ranges)):
            lower = pixel_ranges[i][0]
            upper = pixel_ranges[i][1]
            grouped_value = grouped_values[i]
            mask = cv2.inRange(gray, lower, upper)
            grouped_image[np.where(mask > 0)] = grouped_value

        color_mappings = {
            0: [100, 0, 0],    # Mapping for grayscale value 0 - Black
            80: [255, 0, 0],   # Mapping for grayscale value 1 - Blue
            130: [0, 255, 0],  # Mapping for grayscale value 2 - Green
            170: [0, 0, 255],  # Mapping for grayscale value 3 - Red
            210: [0, 255, 255],  # Mapping for grayscale value 4- Yellow
            255: [255, 255, 255]  # Mapping for grayscale value 4 - White
        }

        color_image = np.zeros((gray.shape[0], gray.shape[1], 3), dtype=np.uint8)
```

```
        # Apply color coding based on grayscale values
        for i in range(grouped_image.shape[0]):
            for j in range(gray.shape[1]):
                if j < 180 or j > 760:
                    color_image[i, j] = [100, 0, 0]
                    continue
                grayscale_value = grouped_image[i, j]
                color = color_mappings.get(grayscale_value, [0, 0, 0])
                color_image[i, j] = color

        #Resize the frame to match the output video's frame size
        #resized_frame = cv2.resize(color_image, (output_width, output_height))

        #Write the resized frame to the output video
        #output_video.write(resized_frame)

    # Release the video file and the output video writer
    video.release()
    output_video.release()

    print("Processing complete. Output video saved at:", output_path)

# Specify the path to your input video
video_path = '/content/drive/MyDrive/inference_data/inference_videos/movi0 (1).mp4'
# Perform acetowhite opacity analysis on the video
acetowhite_opacity_analysis(video_path)
```

Here's the link to the final DYSIS mapping on a video: (Half of the video is cut due to some frame resizing issues which we are working on)

https://drive.google.com/file/d/15X_-J09cdbiSGyoKxGXQ68WHlMapm7Z9/view?usp=sharing

CONCLUSION

I would like to conclude by telling you about the progress that we have achieved in the project so far and how we would continue further. With the help of our mentors, Dr. Shalini and Dr. Satyam Srivastava, we were able to learn how to use YOLOv5 for basic object identification in both images and videos, and we were able to successfully classify the images as positive, negative and suspicious of cancer cases, as shown in the snippets we shared earlier in the main text.

We worked on localizing the lesions with the help of K-means clustering and color compression. Next,we replicated the DYSIS map by implementing the medical image registration and opacity analysis.

Other than that, we worked on inferencing all these tasks on videos as that's where it would be finally used. We have inferenced on videos but there are some issues still there which we are working on.

After this, an API integration on GUI would complete the project, but as of now, this much is done.

We are really glad to have got this opportunity to work on such a sophisticated research project and learned a lot through this.

REFERENCES

1. https://learnopencv.com/custom-object-detection-training-using-yolov5/

2. https://github.com/ultralytics/yolov5

3. https://blog.roboflow.com/train-yolov5-classification-custom-data/#:~:text=YOLOv5%20is%20one%20of%20the,classification%20on%20a%20custom%20dataset.

RESEARCH PAPERS:

4. file:///C:/Users/hp/Downloads/Increased_detection_of_precancerous_cervical_lesio.pdf

5. file:///C:/Users/hp/Downloads/Automated_image_analysis_of_uterine_cervical_image.pdf

6. file:///C:/Users/hp/Downloads/Computerized_image_analysis_for_acetic_acid_induce.pdf

7. file:///C:/Users/hp/Downloads/TSP_CMC_327941%20(1).pdf

GLOSSARY

1. HPV: human papillomavirus

2. CIN: cervical intraepithelial neoplasia

3. DYSIS: Digital Intelligence Systems

4. YOLOv5: You Only Look Once

5. mPA: Mean Average Precision

6. IoU: Intersection over Union