Configuration Manual

MSc Research Project
Data Analytics 2021-2022

Kajol Daiya
Student ID: x19216831

School of Computing
National College of Ireland

Supervisor: Hicham Rifai

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Kajol Daiya
Student ID:	x19216831
Programme:	MSc Data Analytics
Year:	2021-2022
Module:	Research Project
Lecturer:	Hicham Rafai
Submission Due Date:	31.01.22
Project Title:	Instance Segmentation for Detecting Dental Caries in Panoramic X-rays using Detectron2
Word Count:	**Page Count:** 13

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Kajol Daiya

Date: 31.01.2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Kajol Daiya
Student ID: x19216831

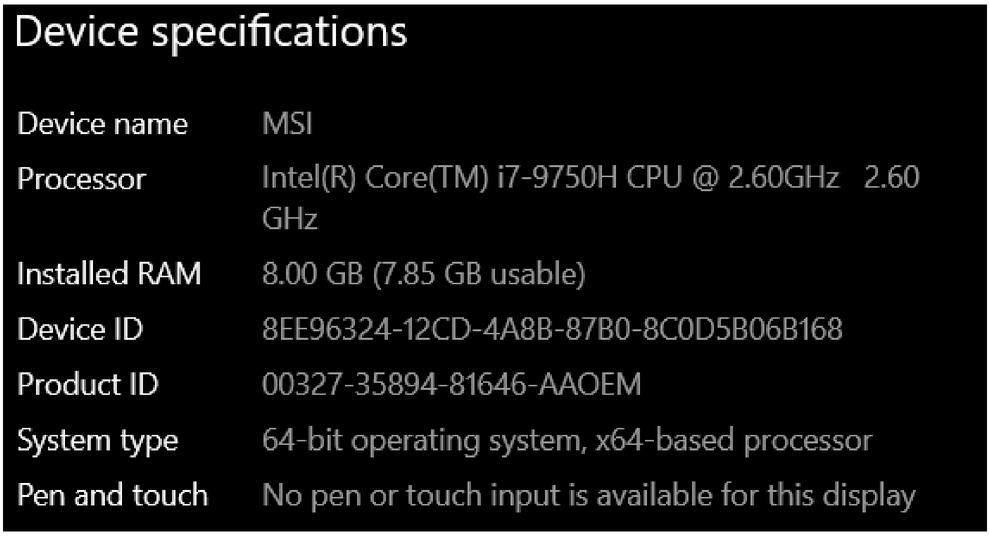
1. Introduction

This document's objective is to outline the stages involved in developing the project. The hardware & system configurations required to reproduce research work are explained in great detail. This section covers the design and implementation strategies needed for effective operations. to built an instance segmentation model to detect dental caries using the Detectron2 model

2. Specifications

Hardware and Software Requirements

- The hardware specifications used to implement the project are shown in figure 1



Device specifications

Device name	MSI
Processor	Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.60 GHz
Installed RAM	8.00 GB (7.85 GB usable)
Device ID	8EE96324-12CD-4A8B-87B0-8C0D5B06B168
Product ID	00327-35894-81646-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Figure 1 Hardware Specification

- **Google Collab** is a cloud-based, open-source platform for developing deep learning algorithms. To sign in to Google Collab, you'll need a Gmail account. Each user is given a minimum of 12.73 GB of RAM, which can be increased to 25 GB, as well as 64 GB of hard disk space. The research is carried out according to the guidelines listed below.

- **LabelMe** is used to annotate the images manually and annotated file is saved inCOCO JSON format to load the data into the Detectron2 model.

- **Cuda Toolkit** is an open-source computing tool that enables users to CUDAgraphics for overall computing.

- **Anaconda3:** For python programming, the platform provides a variety of integrated design frameworks (IDD). The models are created using the libraries indicated below

  - Python 3.6.13 – Libraries
  - numpy 1.19.5
  - tensorflow 1.3.0
  - keras 2.0.8
  - detectron2 arch flags 3.7
  - PyTorch 1.8.0+cu101
  - Pillow 8.3.1
  - torchvision 0.9.0+cu102
  - iopath 0.1.9
  - opencv-python 4.5.3
  - IPython[all]
  - scipy
  - matplotlib
  - scikit-image

3. Data Collection

The data set for this study came from a publicly accessible open-source platform. The dataset contains 116 panoramic x-rays with their relevant masks. The OPG X-ray covers the full region of the patient's mouth. The dental caries type is classed into 5 categories: Dentinal Caries, Proximal Caries, RootPiece, Caries involving pulp, and Secondary Caries, this is annotated manually in this dataset in coordination with 5 dental practitioners to attain accurate annotations.
Link to Dataset: Panoramic Dental X-rays With Segmented Mandibles - Mendeley Data
The database contains 3 directories: Images, Segmentation1 and Segmentation2.
This study takes only the Images folder for Instance Segmentation tasks.

4. Data Pre-processing

The image labeling is performed on LabelMe software after the images are resized. Images are resized in a go with natsorted function in python. In figure 1, the lines of code have successfully resized the images into (255,255)

```
In [3]: ## Exploring Data set
        img_path = 'C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images/1.png'

        img = Image.open(img_path)
        print('{}'.format(img.format))
        print('size:{}'.format(img.size))
        print('image mode:{}'.format(img.mode))
        img.show()

        PNG
        size:(3100, 1300)
        image mode:L

In [4]: #empty lists
        image_list = []
        resized_images = []

In [5]: #append images to list
        for filename in natsorted(glob.glob('C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images/*.png')):
            print(filename)
            opg_image = Image.open(filename)
            image_list.append(opg_image)

        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\7.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\8.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\9.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\10.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\11.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\12.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\13.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\14.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\15.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\16.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\17.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\18.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\19.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\20.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\21.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\22.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\23.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\24.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\25.png
        C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/Images\26.png

In [9]: #append resized images to list
        for image in image_list:
            image.show()
            image = image.resize((255,255))
            resized_images.append(image)

In [10]: #save resized images to new folder

         for (i, new) in enumerate(resized_images):
             new.save('{}{}{}'.format('C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/ResizedImages',i+1,'.png'))
```

Figure 1 Images Resized

Figure 2 Interface of the LabelMe Software

Dental X-ray images tend to be quite noisy coming from various types of noise sources. To de-noise these x-ray images, denoising filters are used to highlight useful details in the x-ray and increase its image quality. Image thresholding and equalization are some of the tools that we have for image processing so image thresholding for segmentation tasks becomes a bit easier. With histogram equalization, we can stretch the histogram to span the entire range. Histogram Equalization considers the global contrast of the image, not just the local contrast. The result of

Histogram equalization and Contrast Limiting Adaptive Histogram Equalizer was taken out. CLAHE does histogram equalization in small patches and it works very well and does contrast limiting

CLAHE and EQUALIZATION

```
In [12]: img_path = 'C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/ResizedImages/ResizedImages1.png'

         img = Image.open(img_path)
         print('{}'.format(img.format))
         print('size:{}'.format(img.size))
         print('image mode:{}'.format(img.mode))
         img.show()

         PNG
         size:(255, 255)
         image mode:L
```

```
In [2]: import cv2
        import numpy as np
        from matplotlib import pyplot as plt

        C:\Users\kdaiy\anaconda3\lib\site-packages\numpy\_distributor_init.py:30: UserWarning: loaded more than 1 DLL from .libs:
        C:\Users\kdaiy\anaconda3\lib\site-packages\numpy\.libs\libopenblas.NOIJJG62EMASZI6NYURL6JBKM4EVBGM7.gfortran-win_amd64.dll
        C:\Users\kdaiy\anaconda3\lib\site-packages\numpy\.libs\libopenblas.WCDJNK7YVMPZQ2ME2ZZHJJRJ3JIKNDB7.gfortran-win_amd64.dll
          warnings.warn("loaded more than 1 DLL from .libs:"
```

```
In [30]: xray = cv2.imread("C:/Users/kdaiy/Downloads/Teeth_SegCaps/SegCaps/ResizedImages/ResizedImages114.png",0)
         eq_img = cv2.equalizeHist(xray)

         plt.hist(eq_img.flat, bins=100, range=(0,255))

         clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
         cl_img = clahe.apply(xray)

         #cv2.imshow('Equalised Image', eq_img)
         #cv2.imshow('CLAHE image', cl_img)
         #cv2.waitKey(0)
         #cv2.distroyAllWindows()
         img_titles =["ORIGINAL","CLAHE", "EQUALISED"]
         opg_xray = [xray, cl_img, eq_img]
         for i in range(3):
             plt.subplot(1,2,i+1),plt.imshow(opg_xray[i],'gray')
             plt.title(img_titles[i])
             plt.xticks([]),plt.yticks([])
         plt.show()
```

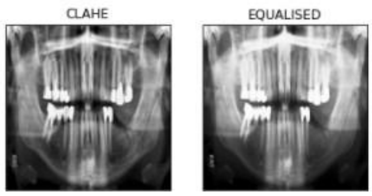


Figure 3 Image Preprocessing

Thresholding

In [32]:
```
ret, thresh1 = cv2.threshold(cl_img, 190,150, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(cl_img, 190,255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(cl_img,190,255,cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(cl_img,190,255,cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(cl_img,190,255,cv2.THRESH_TOZERO_INV)
#cv2.imshow('Original', xray)
#cv2.imshow('Binary Threshold 1',cl_img)
#cv2.waitKey(0)
titles = ['Original Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
xray_images = [xray, thresh1, thresh2, thresh3, thresh4, thresh5]
for i in range(6):
    plt.subplot(2,3,i+1),plt.imshow(xray_images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

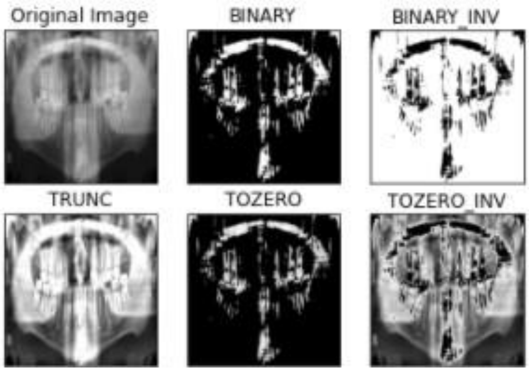


Figure 4 Thresholding Technique

Data Transformation:

The dataset in this study contains only 116 images which are split into a train (70%), test(20%), validation(10%). The training dataset contains 93 relatively small images, also these images are distributed unevenly among 5 classes of dental caries.

Split into train test validate

```
In [36]: pip install split-folders

Collecting split-folders
  Downloading split_folders-0.4.3-py3-none-any.whl (7.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.4.3
Note: you may need to restart the kernel to use updated packages.

In [27]: import splitfolders

In [28]: pip install split-folders tqdm

Requirement already satisfied: split-folders in c:\users\kdaiy\anaconda3\lib\site-packages (0.4.3)
Requirement already satisfied: tqdm in c:\users\kdaiy\anaconda3\lib\site-packages (4.62.2)
Requirement already satisfied: colorama in c:\users\kdaiy\anaconda3\lib\site-packages (from tqdm) (0.4.4)
Note: you may need to restart the kernel to use updated packages.

In [29]: input_folder = 'opg_xrays/'

In [30]: splitfolders.ratio(input_folder, output="opg_xrays2",
                            seed=42, ratio=(.7, .2, .1),
                            group_prefix=None) # default values

Copying files: 0 files [15:00, ? files/s]

Copying files: 15 files [00:00, 148.54 files/s]

Copying files: 45 files [00:00, 237.27 files/s]

Copying files: 69 files [00:00, 237.96 files/s]

Copying files: 93 files [00:00, 233.47 files/s]

Copying files: 123 files [00:00, 256.75 files/s]

Copying files: 149 files [00:00, 248.14 files/s]

Copying files: 178 files [00:00, 261.24 files/s]
```

5. Detectron2 Model

Google Colab was used to train the Detector2 model

```
#Mount drive to fetch data
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

#Installing Libraries
!pip install pyyaml==5.1
!pip install torch==1.8.0+cu101 torchvision==0.9.0+cu101 -f https://download.pytorch.org/whl/torch_stable.html

Collecting pyyaml==5.1
  Downloading PyYAML-5.1.tar.gz (274 kB)
     |████████████████████████████████| 274 kB 5.2 MB/s
Building wheels for collected packages: pyyaml
  Building wheel for pyyaml (setup.py) ... done
  Created wheel for pyyaml: filename=PyYAML-5.1-cp37-cp37m-linux_x86_64.whl size=44092 sha256=31c606e19fe7979d416737e4dd8bdbfbeacb82e8b773d92e7291e53301c71298
  Stored in directory: /root/.cache/pip/wheels/77/f5/10/d00a2bd30928b972790053b5de0c703ca87324f3fead0f2fd9
Successfully built pyyaml
Installing collected packages: pyyaml
  Attempting uninstall: pyyaml
    Found existing installation: PyYAML 3.13
    Uninstalling PyYAML-3.13:
      Successfully uninstalled PyYAML-3.13
Successfully installed pyyaml-5.1
Looking in links: https://download.pytorch.org/whl/torch_stable.html
Collecting torch==1.8.0+cu101
  Downloading https://download.pytorch.org/whl/cu101/torch-1.8.0%2Bcu101-cp37-cp37m-linux_x86_64.whl (763.5 MB)
     |████████████████████████████████| 763.5 MB 15 kB/s
Collecting torchvision==0.9.0+cu101
  Downloading https://download.pytorch.org/whl/cu101/torchvision-0.9.0%2Bcu101-cp37-cp37m-linux_x86_64.whl (17.3 MB)
     |████████████████████████████████| 17.3 MB 841 kB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from torch==1.8.0+cu101) (1.19.5)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from torch==1.8.0+cu101) (3.10.0.2)
Requirement already satisfied: pillow>=4.1.1 in /usr/local/lib/python3.7/dist-packages (from torchvision==0.9.0+cu101) (7.1.2)
Installing collected packages: torch, torchvision
  Attempting uninstall: torch
    Found existing installation: torch 1.10.0+cu111
    Uninstalling torch-1.10.0+cu111:
      Successfully uninstalled torch-1.10.0+cu111
  Attempting uninstall: torchvision
    Found existing installation: torchvision 0.11.1+cu111
    Uninstalling torchvision-0.11.1+cu111:
      Successfully uninstalled torchvision-0.11.1+cu111
```

Figure 3 Mounting Google Drive

```
        annos = img_anns["shapes"]
        objs = []
        for anno in annos:
            px = [a[0] for a in anno['points']] #x coord
            py = [a[1] for a in anno['points']] #y coord
            poly = [(x,y) for x, y in zip(px,py)] #poly for segmentation
            poly = [p for x in poly for p in x]

            obj = {
                "bbox": [np.min(px), np.min(py), np.max(px), np.max(py)],
                "bbox_mode": BoxMode.XYXY_ABS,
                "segmentation": [poly],
                "category_id": classes.index(anno['label']),
                "iscrowd": 0
            }
            objs.append(obj)
        record["annotations"] = objs
        dataset_dicts.append(record)
    return dataset_dicts
```

Figure 4 Creating directory for image data and json file

```
classes = ['Dentinal Caries', 'Caries involving pulp', 'Rootpiece', 'Proximal Caries', 'Secondary Caries','Healthy Dentition']
data_path = '/content/drive/MyDrive/Kajol/'
for d in ["train", "test"]:
    DatasetCatalog.register(
        "category_" + d,
        lambda d=d: get_data_dicts(data_path+d, classes)
    )
    MetadataCatalog.get("category_" + d).set(thing_classes=classes)
microcontrollermetadata_metadata = MetadataCatalog.get("category_train")
```

Figure 5 Classes are defined

Model Training

Pre trained model mask rcnn R 101 is selected as the base model by selecting hyperparameters as seen in the Figure 6.

```
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_101_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("category_train",)
cfg.DATASETS.TEST = ("category_test",)
cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_101_FPN_3x.yaml")
cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.0025
cfg.SOLVER.MAX_ITER = 3500
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 6

cfg.TEST.EVAL_PERIOD = 1000
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
cfg.MODEL.RETINANET.NUM_CLASSES = 6
cfg.MODEL.ROI_KEYPOINT_HEAD.NUM_KEYPOINTS = 4
cfg.TEST.KEYPOINT_OKS_SIGMAS = np.ones((4,1), dtype=float).tolist()
```

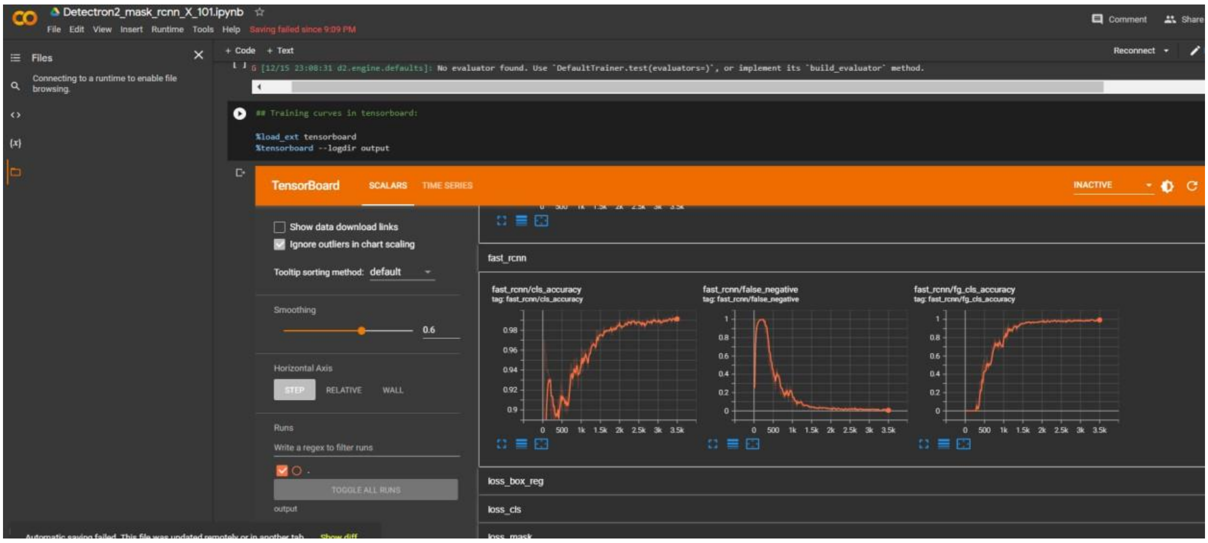


Figure 6 TensorFlow Output after training the model

```
[ ]  for d in random.sample(test_dataset_dicts, 1):
         img = cv2.imread(d["file_name"])

         outputs = predictor(img)
         v = Visualizer(img,
                        metadata=microcontrollermetadata_metadata,
                        scale=0.8,
                        instance_mode=ColorMode.IMAGE_BW   #removes colors of unsegmented pixels
         )
         v = v.draw_instance_predictions(outputs["instances"].to("cpu"))
         plt.figure(figsize = (14,10))
         plt.imshow(cv2.cvtColor(v.get_image()[:, :, ::-1], cv2.COLOR_BGR2RGB))
         plt.show()
```

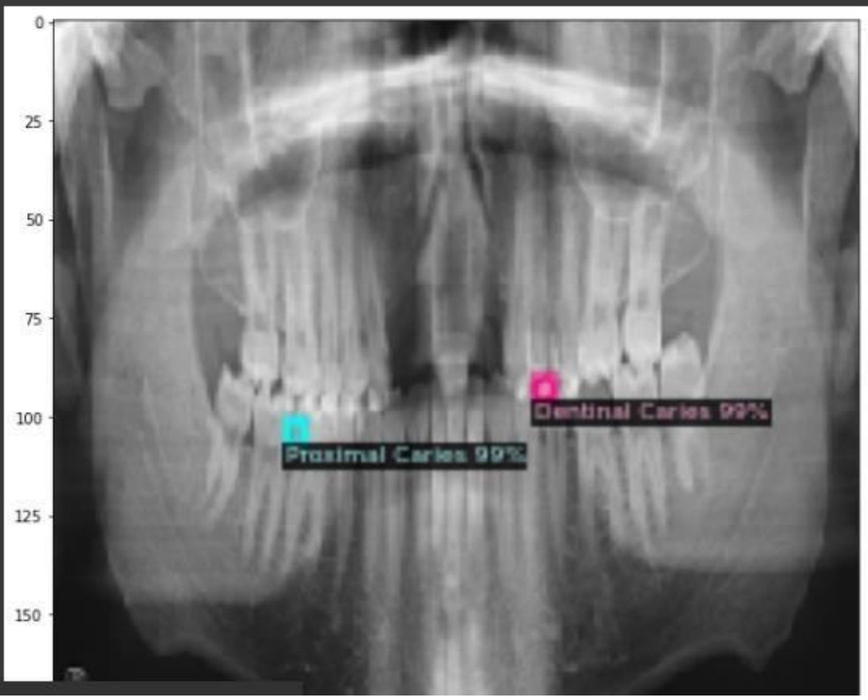


Figure 7 Instance Segmentation Results are visualized with Visualizer class

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.535
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.977
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.510
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.535
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.488
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.593
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.593
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.593
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
[01/10 03:19:32 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|   AP   |  AP50  |  AP75  |  APs   |  APm  |  APl  |
|:------:|:------:|:------:|:------:|:-----:|:-----:|
| 53.512 | 97.707 | 51.005 | 53.512 |  nan  |  nan  |
[01/10 03:19:32 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
[01/10 03:19:32 d2.evaluation.coco_evaluation]: Per-category segm AP:
| category         | AP     | category              | AP     | category          | AP     |
|:-----------------|:-------|:----------------------|:-------|:------------------|:-------|
| Dentinal Caries  | 51.785 | Caries involving pulp | 43.117 | Rootpiece         | 54.281 |
| Proximal Caries  | 47.771 | Secondary Caries      | 39.695 | Healthy Dentition | 84.422 |
```

Figure 8 Model Evaluation

Figure 9 Average Precision and Recall results

6. References

COCO Consortium (2016) COCO - Common Objects in Context. Available at: https://cocodataset.org/#detection-eval (Accessed: 13 August 2020).

Detectron2: A PyTorch-based modular object detection library (2019). Available at: https://ai.facebook.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library-/ (Accessed: 13 August 2020).

Installation — detectron2 0.2.1 documentation (2019). Available at: https://detectron2.readthedocs.io/tutorials/install.html (Accessed: 13 August 2020).