BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Teeth Problems Detection

– MIRPR report –

Team members
Hotca Diana, 234/1, email: dianahotca10@gmail.com
Ifrim Gianina, 234/1, email: gcifrim1@gmail.com
Ionita Catalin, 234/1, email: catalin.ionita004@gmail.com

2020-2021

Abstract

The aim of this study is automatic semantic segmentation of teeth problems in one-shot panoramic x-ray image by using deep learning method with U-Net Model and binary image analysis in order to provide diagnostic information for the management of dental disorders, diseases, and conditions. The user can upload a radiography and the algorithm informs him if there is any dental problem by highlighting areas with possible dental problems. The data used to train and test this CNN model consists of 100 dental x-rays collected by doctors.

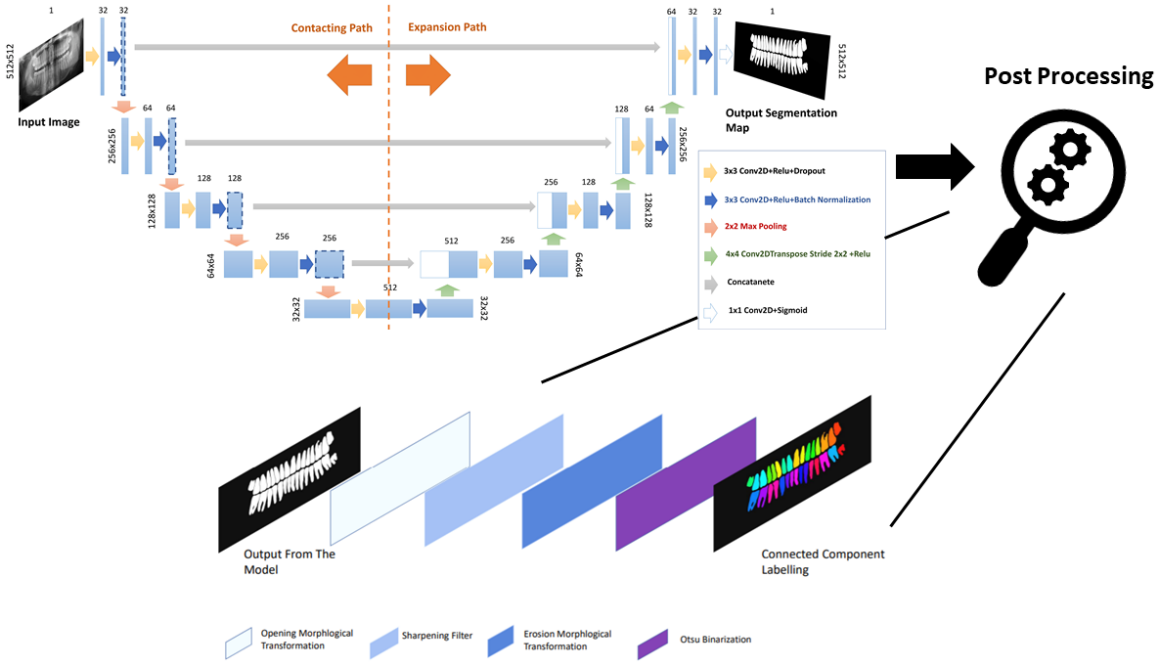


Figure 1: Architecture

Contents

List of Figures

Chapter 1

Introduction

1.1 What? Why? How?

In dentistry, Dental X-ray systems help dentists by showing the basic structure of tooth bones to detect various kinds of dental problems. However, depending only on dentists can sometimes impede treatment since identifying things in X-ray pictures requires human effort, experience, and time, which can lead to delays in the process. In image classification, segmentation, object identification, and machine translation, recent improvements in deep learning have been effective. Deep learning may be used in X-ray systems to detect objects. Radiology and pathology have benefited greatly from the use of deep convolutional neural networks, which are a fast growing new area of medical study. Deep learning techniques for the identification of objects in dental X-ray systems are the focus of this study. As part of the study, Deep Neural Network algorithms were evaluated for their ability to identify dental cavities and a root canal on periapical radiographs. An automated detection method for dental caries and root canals in X-rays is presented utilizing Tensor Flow tool packages' faster regions with convolutional neural network characteristics.

Our goals included:

- Build a model that can identify the tooth official index. This is a classification task, and will be the subject of another article.
- Build a model that can detect restorations, endodontic treatments and implants on a full dental x-ray. This is an object detection task.

The idea was not to make a software that would replace doctors at all, but to build a proof of concept that could add a second opinion and could make reporting easier for doctors.

1.2 Paper structure and original contribution(s)

The research presented in this paper advances the theory, design, and implementation of several particular models.

The main contribution of this report is to present an intelligent algorithm for detecting the teeth problems from dental x-rays.

The second contribution of this report consists of building an intuitive, easy-to-use and user friendly software application. Our aim is to build an algorithm that will help doctors in detection and classification of teeth problems and could add a second opinion and could make reporting easier for doctors.

Chapter 2

Scientific Problem

2.1 Problem definition

The application detects teeth problems represented by more pronounced areas in dental X-rays images with the help of Convolutional Neural Networks.

The input data of the problem is represented by dental X-rays images that in which different teeth problems are represented such as dental caries and root canals.

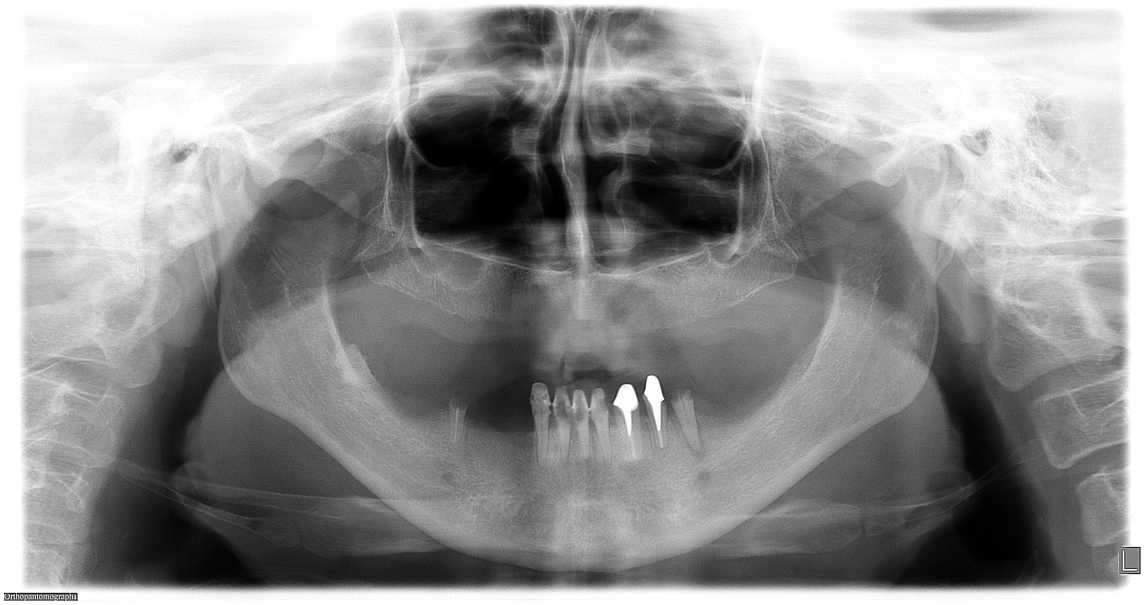


Figure 2.1: Dental X-ray

The output for this model is represented by highlighting areas with possible dental problems.

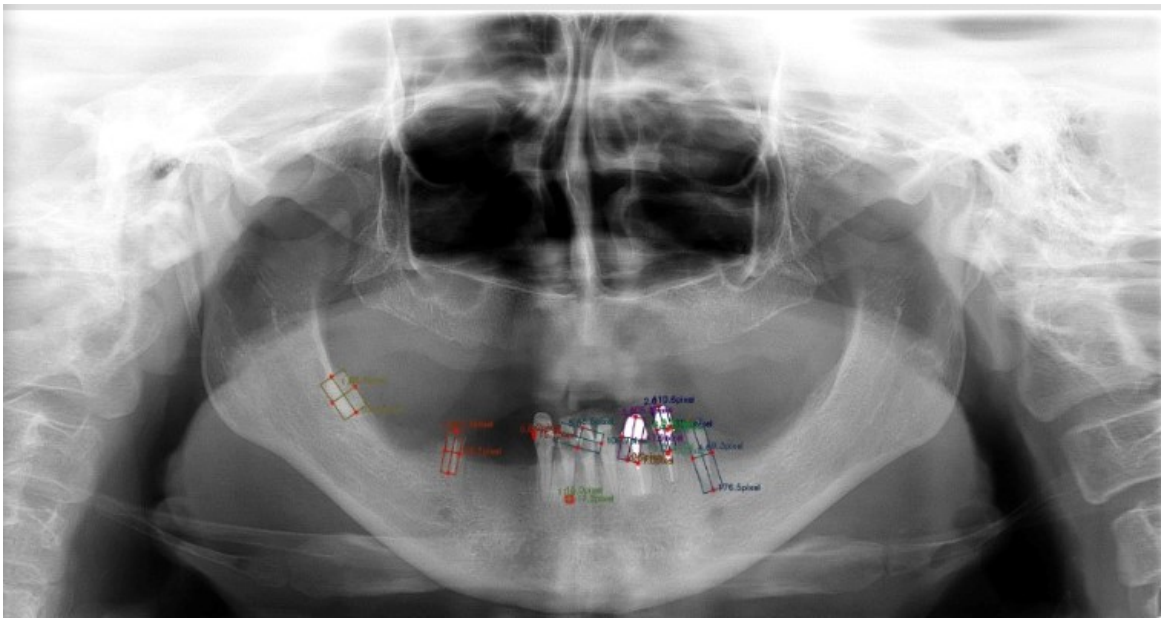


Figure 2.2: Problems Detected

Chapter 3

State of the art/Related work

A good example of detecting dental problems is that of Clement Joudet, made together with Dr Arthur Fourcade, an oral surgeon who made his thesis on CNNs applied to medical imagery. Their work is published here.

The dataset they used consists of more than 500 dental panoramic x-rays, using the VoTT open-source software. The labels can be exported under various formats, including the Tensorflow PASCAL VOC. Dr Fourcade used various datasets of dental x-rays he built during his career. The annotated dataset is not currently open sourced, but will hopefully be soon. They used Tensorflow Object Detection API because of its simplicity and plug-and-play approach.

They explained every step of their work here and made the code public.

Dataset features:

- Every x-ray has been anonymized for privacy protection reasons, there is no way for me to trace back to any individual.
- More than 500 panoramic x-rays annotated.
- Average size of x-ray: 2900 * 1400 px
- Classes: 933 endodontic, 2331 restorations and 145 implant occurrences. The number of implant occurrences is a bit low and makes it quite hard for the model to detect implants in an evaluation dataset. This can be countered with data augmentation techniques.

Chapter 4

Investigated approach and Methodology

The first step we take is to download the dataset that includes 100 dental X-rays images and 100 images that have the teeth problems segmented.

Then, we split the data into training data(80% of the data) and testing data (20%) and also augment training data so that the algorithm detects dental problems more clearly.

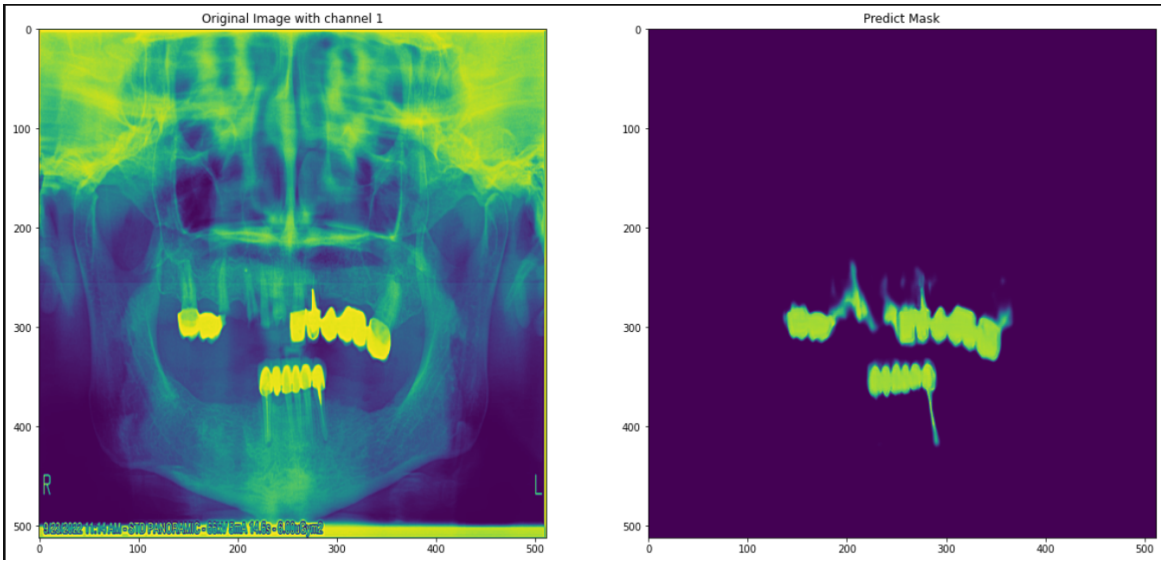
After all that, we train the algorithm by using deep learning method with U-Net Model and binary image analysis in order to detect dental problems.

```
from model import *
model=UNET(input_shape=(512,512,1),last_activation='sigmoid')
model.summary()
```

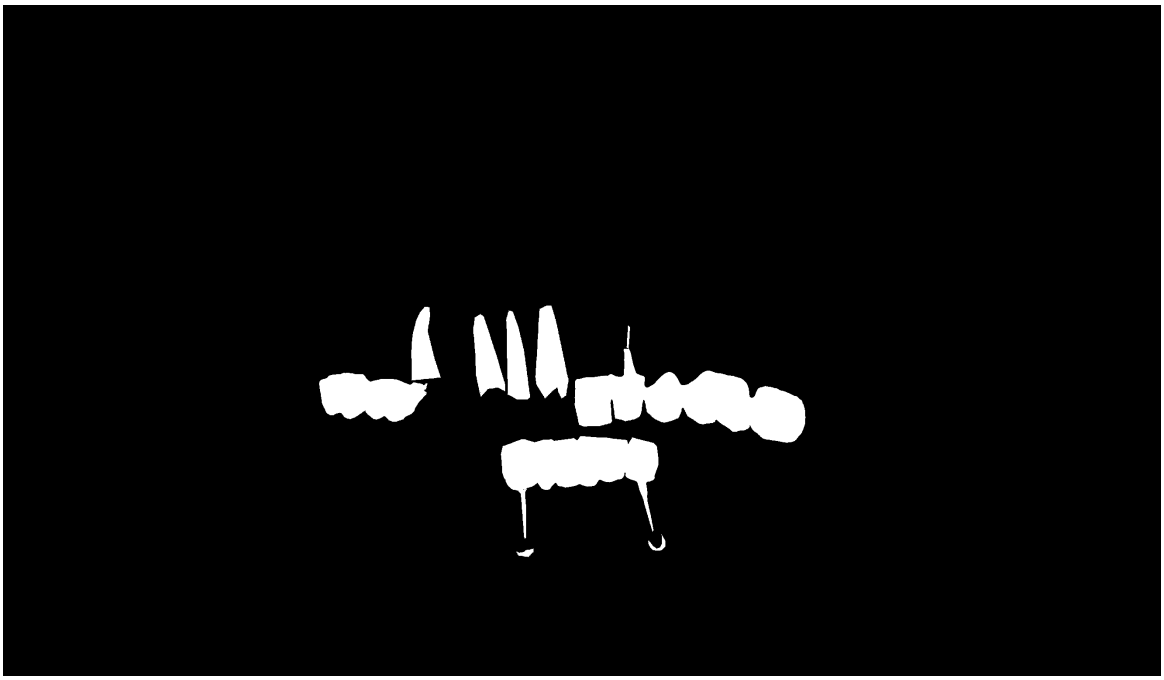
```
model.compile(optimizer ='adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

model.fit(x_train,y_train,batch_size=8,epochs=100,verbose=1)
```

Then, we make predictions:

```
predict_img=model.predict(x_test)
##model.save(path)
predict=predict_img[0,:,:,0]
```



The original mask was:



Then, we make a CCA Analysis to detect dental problems based on prediction:

```
cca_result,teeth_count=CCA_Analysis(img,predicted,3,2)
plt.imsave("/content/detected.png",cca_result)
```

And .... it's done! The algorithm detected teeth problems from an image.

Connecting to the Web App

For this part we used **Anvil** - a free Python-based drag-and-drop web app builder.

```
anvil.server.connect("X4D6NLG34YNOA6TM7Q2J4EWN-T3IPCXFQWW4PSGYC")
```

```
import anvil.media
import base64
import io
from PIL import Image
import tensorflow
from tensorflow.keras.utils import load_img


@anvil.server.callable
def detect_problems(file):
  img = Image.open(io.BytesIO(file.get_bytes()))
  img.save("/content/uploaded.png")
  img=cv2.imread("/content/uploaded.png")

  ########################################
  # predictie
  img2 = Image.open("/content/uploaded.png")
  img2 = img2.resize((512,512), Image.ANTIALIAS)
  img2 = convert_one_channel(np.asarray(img2))
  # img2 = np.expand_dims(img2, axis=0)
  img2 = np.reshape(img2,(1,512,512,1))
  img2 = np.float32(img2/255)

  predict_img=model.predict(img2)
  predict=predict_img[0,:,:,0]
  plt.imsave("/content/predict.png",predict)
  ########################################

  predicted=cv2.imread("/content/predict.png")
  predicted = cv2.resize(predicted, (img.shape[1],img.shape[0]), interpolation=cv2.INTER_LANCZOS4)

  cca_result,teeth_count=CCA_Analysis(img,predicted,3,2)
  plt.imsave("/content/detected.png",cca_result)

  with open("/content/detected.png", "rb") as image_file:
     encoded_string = base64.b64encode(image_file.read())
  image_binary = base64.b64decode(encoded_string)
  my_media = anvil.BlobMedia(content_type="img/png", content=image_binary, name="detected")

  return my_media
```

Form1 Design Code

```
1  from ._anvil_designer import Form1Template
2  from anvil import *
3  import anvil.server
4
5  class Form1(Form1Template):
6
7    def __init__(self, **properties):
8      # Set Form properties and Data Bindings.
9      self.init_components(**properties)
10
11   def file_loader_1_change(self, file, **event_args):
12       self.image_1.source = file
13
14
15   def button_1_click(self, **event_args):
16       self.image_1.source =  anvil.server.call('detect_problems',self.image_1.source)
```

Chapter 5

Application (Study case)

5.1 App's description and the main functionalities

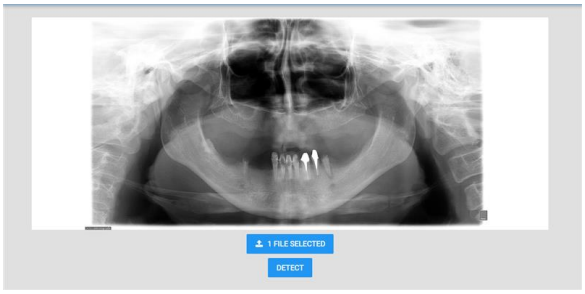


Figure 5.1: Before detection

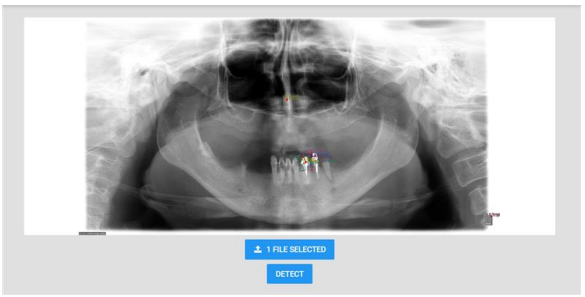


Figure 5.2: After detection

- **Upload Image**

  By clicking the "Upload Image" Button, the user can select an X-Ray image from his computer which will be uploaded and further analyzed.

- **Detect**

  By clicking the "Detect" Button, the application will start analyzing the image and will detect possible dental problems which will be highlighted on the previously uploaded X-Ray.

5.2 Implementation

For the implementation we used Python combined with .ipynb notebook for the back-end side. For the front-end side we used Anvil web app builder. Anvil is a free Python-based drag-and-drop web app builder.

5.3 Numerical validation

As you can see in the images below, the loss is continuously decreasing, and the accuracy increases at the beginning, and then it drops to a super high value. (algorithm ran with `batch_size`=8 and epochs=100)

```
model.compile(optimizer ='adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

model.fit(x_train,y_train,batch_size=8,epochs=100,verbose=1)

Epoch 1/100
50/50 [==============================] - 56s 1s/step - loss: 5.0401e-04 - accuracy: 0.9841
Epoch 2/100
50/50 [==============================] - 54s 1s/step - loss: 5.0231e-04 - accuracy: 0.9841
Epoch 3/100
50/50 [==============================] - 54s 1s/step - loss: 4.9804e-04 - accuracy: 0.9841
Epoch 4/100
50/50 [==============================] - 54s 1s/step - loss: 4.9197e-04 - accuracy: 0.9841
Epoch 5/100
50/50 [==============================] - 54s 1s/step - loss: 4.9151e-04 - accuracy: 0.9841
Epoch 6/100
50/50 [==============================] - 54s 1s/step - loss: 4.8972e-04 - accuracy: 0.9841
Epoch 7/100
50/50 [==============================] - 54s 1s/step - loss: 4.8912e-04 - accuracy: 0.9841
Epoch 8/100
50/50 [==============================] - 54s 1s/step - loss: 4.8581e-04 - accuracy: 0.9841
Epoch 9/100
50/50 [==============================] - 54s 1s/step - loss: 4.8788e-04 - accuracy: 0.9841
Epoch 10/100
50/50 [==============================] - 54s 1s/step - loss: 4.8502e-04 - accuracy: 0.9841
Epoch 11/100
50/50 [==============================] - 54s 1s/step - loss: 4.7906e-04 - accuracy: 0.9841
Epoch 12/100
50/50 [==============================] - 54s 1s/step - loss: 4.7899e-04 - accuracy: 0.9841
Epoch 13/100
50/50 [==============================] - 54s 1s/step - loss: 4.7652e-04 - accuracy: 0.9841
```

Figure 5.3: Problems Detected

```
50/50 [==============================] - 54s 1s/step - loss: 4.3001e-04 - accuracy: 0.9841
Epoch 95/100
50/50 [==============================] - 54s 1s/step - loss: 4.2972e-04 - accuracy: 0.9841
Epoch 96/100
50/50 [==============================] - 54s 1s/step - loss: 4.2944e-04 - accuracy: 0.9841
Epoch 97/100
50/50 [==============================] - 54s 1s/step - loss: 4.2922e-04 - accuracy: 0.9841
Epoch 98/100
50/50 [==============================] - 54s 1s/step - loss: 4.2939e-04 - accuracy: 0.9841
Epoch 99/100
50/50 [==============================] - 54s 1s/step - loss: 4.2894e-04 - accuracy: 0.9841
Epoch 100/100
50/50 [==============================] - 54s 1s/step - loss: 4.2903e-04 - accuracy: 0.9841
```

Figure 5.4: Problems Detected

5.4 Data

For the **first step** of the app (detecting teeth from dental x-rays) we used a dataset downloaded from here. There are 116 X-ray images with teeth and 116 images with the same images but the teeth are segmented:

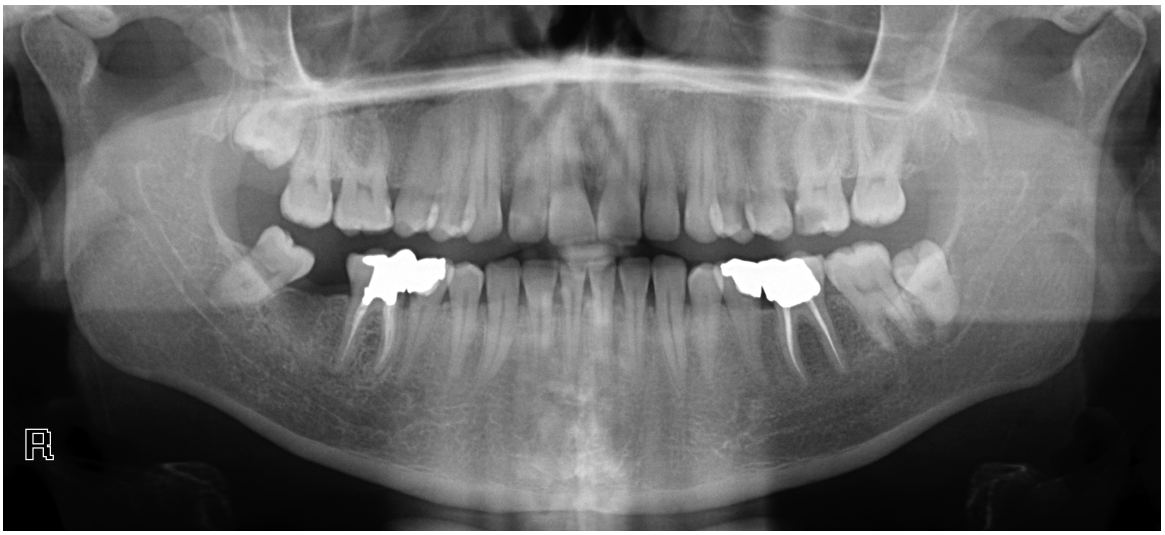


Figure 5.5: X-Ray Image

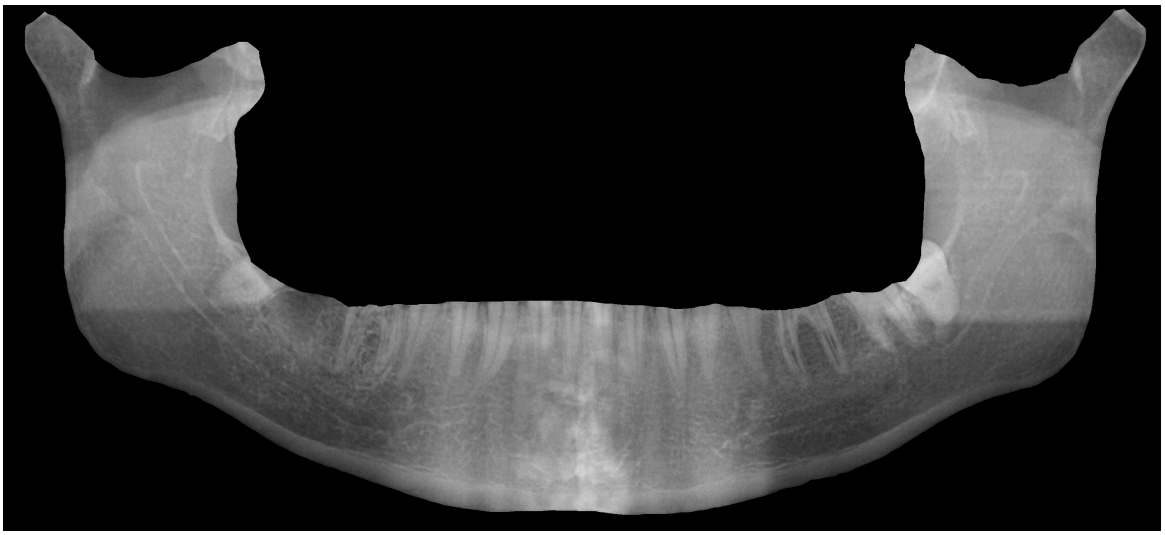


Figure 5.6: Segmented Teeth Image

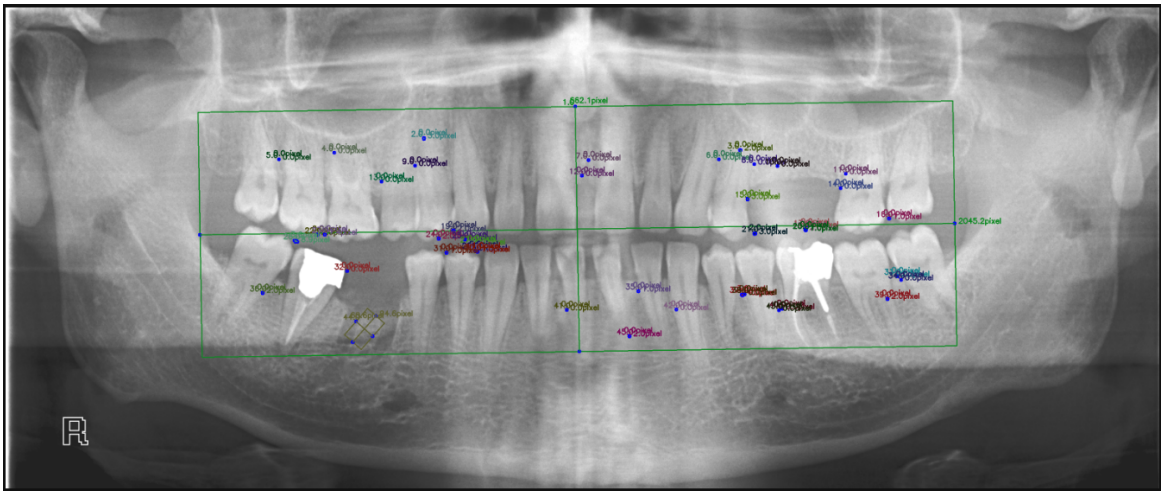
For the **next step** of the app (detecting teeth problems from dental x-rays) we used a dataset that has 100 dental x-rays and 100 images who have segmented dental problems

Figure 5.7: Segmented Teeth Problems Image

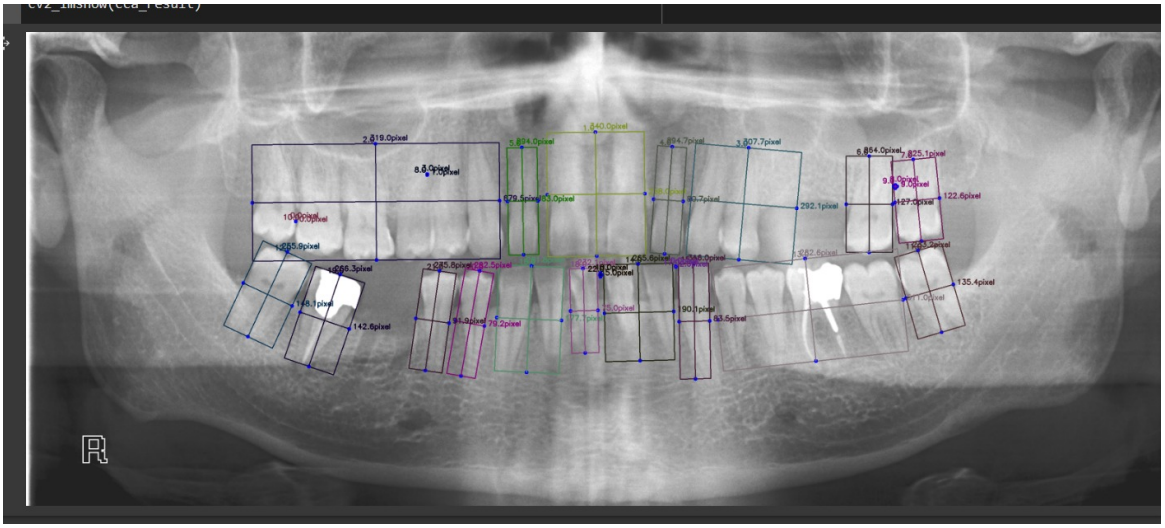
5.5 Results

In the **first version** of the app, where we detected teeth, we obtained the following results:

- For `batch_size`=8 and epochs=10

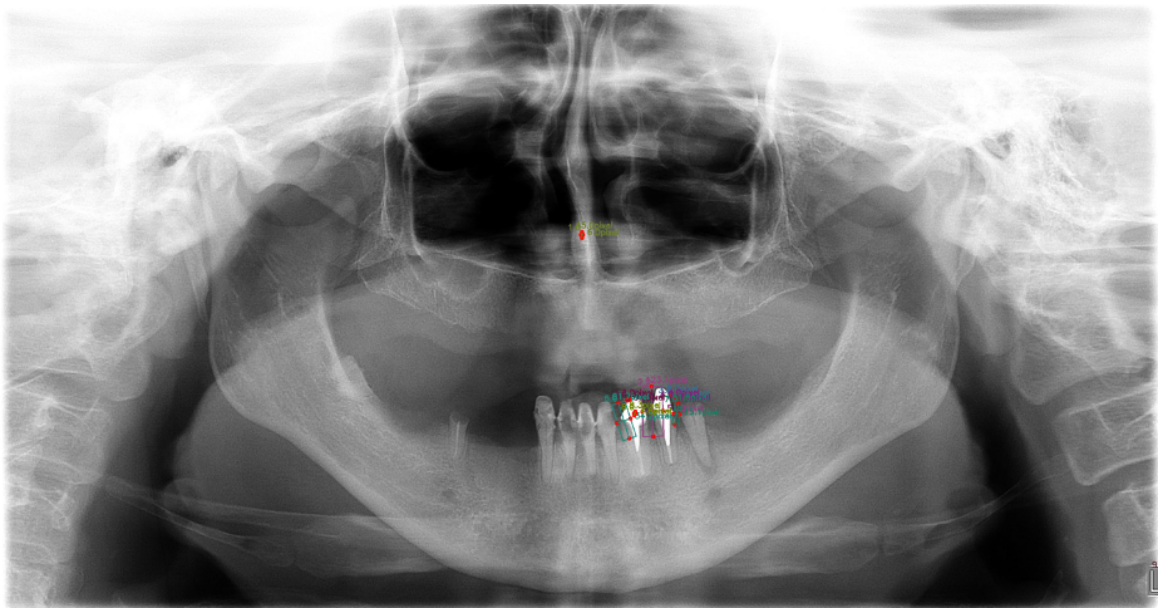


- For `batch_size`=8 and epochs=50

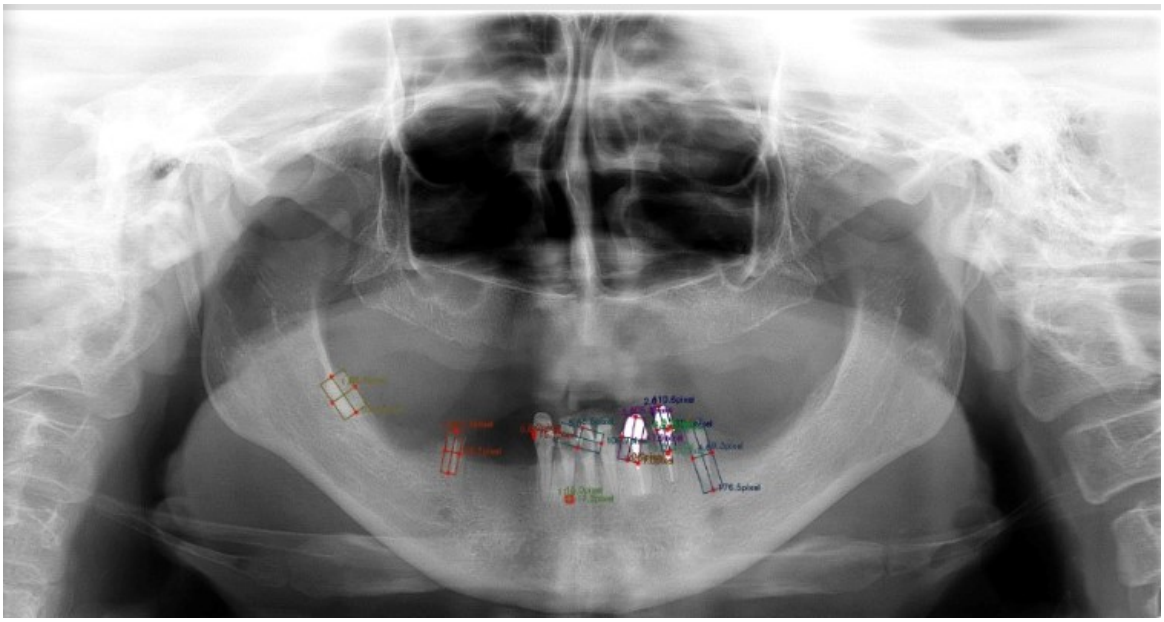


In the **second version** of the app, where we detected dental problems, we obtained the following results:

- For `batch_size`=8 and epochs=10



- For `batch_size`=8 and epochs=100



As the images show in both cases, along with the increase in the number of epochs, the performance of the algorithm also increased, detecting teeth/teeth problems better.

Chapter 6

Conclusion and future work

The applications detects most of the teeth problems, but still cannot classify dental problems.

In the future we would like to be able to classify dental problems and also to optimize the algorithm to run in the shortest possible time but also efficiently at the same time.

Bibliography

[1] Clement Joudet *Deep learning object detection on dental x-rays*

[2] SerdarHelli *Segmentation-of-Teeth-in-Panoramic-X-ray-Image-Using-U-Net*

[3] *Anvil*