UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

X-Teeth

Presentado por Ismael Franco Hernando
en Universidad de Burgos — 5 de julio de 2022
Tutores: César Ignacio García Osorio y
José Miguel Ramírez Sanz

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. César Ignacio García Osorio, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos y D. José Miguel Ramírez Sanz, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Exponen:

Que el alumno D. Ismael Franco Hernando, con DNI 71363577M, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado X-Teeth.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección de los que suscriben, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 5 de julio de 2022

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. César Ignacio García Osorio	D. José Miguel Ramírez Sanz

Resumen

La infección de la parte interna del diente, formada por nervios y vasos sanguíneos, requiere de una intervención para poder aliviar el dolor al paciente. Este proceso se conoce como endodoncia, en el cual el odontólogo retira todo el tejido dañado.

Para poder desempeñar de forma correcta la tarea, el odontólogo necesita conocer el tamaño del diente de la forma más exacta posible para no causar daños irreparables. Todo el proceso se realiza de forma manual, donde al tratarse de medidas milimétricas es fácil cometer errores de precisión.

Es por ello, que este proyecto busca de la mano del *deep learning* automatizar la tarea de estimar la longitud del diente a través de una radiografía. Para ello se ha investigado acerca de la creación de un modelo capaz de realizar la segmentación de dientes y nervios en radiografías, y obtener una buena técnica para calcular el tamaño del diente.

Finalmente, indicar que este proyecto ha sido una colaboración con el Dr. Álvaro Zubizarreta Macho[1], odontólogo de la Universidad Alfonso X el Sabio.

Actualmente se puede acceder a la aplicación del proyecto de dos formas diferentes:

- **Google Colab**: `https://bit.ly/3R86dJc`
- **Gamma**: servidor de la Universidad de Burgos que contiene la aplicación.
  - Enlace web: `https://7b9a-193-146-172-150.ngrok.io`
  - Contraseña: `TfgUbu2122`

Descriptores

Endodoncia, *deep learning*, detección, longitud diente, visión artificial, investigación.

[1]Perfil en ResearchGate: `https://www.researchgate.net/profile/Alvaro-Macho`

Abstract

Infection of the inner part of the tooth, formed by nerves and blood vessels, requires an intervention to relieve the patient's pain. This process is known as endodoncy, in which the dentist removes all the damaged tissue.

In order to perform the task correctly, the dentist needs to know the size of the tooth as accurately as possible, so as not to cause irreparable damage. The entire process is carried out manually and it is easy to make a mistake, since millimetric precision is required.

Therefore, this project pursues the objective of estimating the length of the tooth through an X-ray by using deep learning. For this purpose, research has been carried out on the creation of a model capable of performing the segmentation of teeth and nerves in radiographs, and obtaining a good technique to calculate the size of the tooth.

Finally, indicate that this project has been a collaboration with Dr. Álvaro Zubizarreta Macho[2], dentist at the University of Alfonso X el Sabio.

The software developed in this project can currently be accessed in two different ways:

- **Google Colab**: https://bit.ly/3R86dJc
- **Gamma**: server owned by the University of Burgos that contains the application.
  - Web link: https://7b9a-193-146-172-150.ngrok.io
  - Passworrd: `TfgUbu2122`

Keywords

Endodoncy, endodontics, deep learning, detection, tooth length, computer vision, research.

[2]Profile on ResearchGate: https://www.researchgate.net/profile/Alvaro-Macho

Índice general

Índice de figuras

Índice de tablas

Introducción

Una endodoncia [10] se realiza cuando se infecta o se inflama la parte interna del diente, denominada pulpa, y que contiene los vasos sanguíneos y el nervio del diente. Dicho proceso, busca eliminar la pulpa de forma parcial o total, para salvar el diente en su totalidad, y rellenarlo de un material inerte.

La endodoncia es necesaria, ya que al encontrarse la inflamación en el interior de un diente y al no ser un tejido elástico, puede causar un gran dolor.

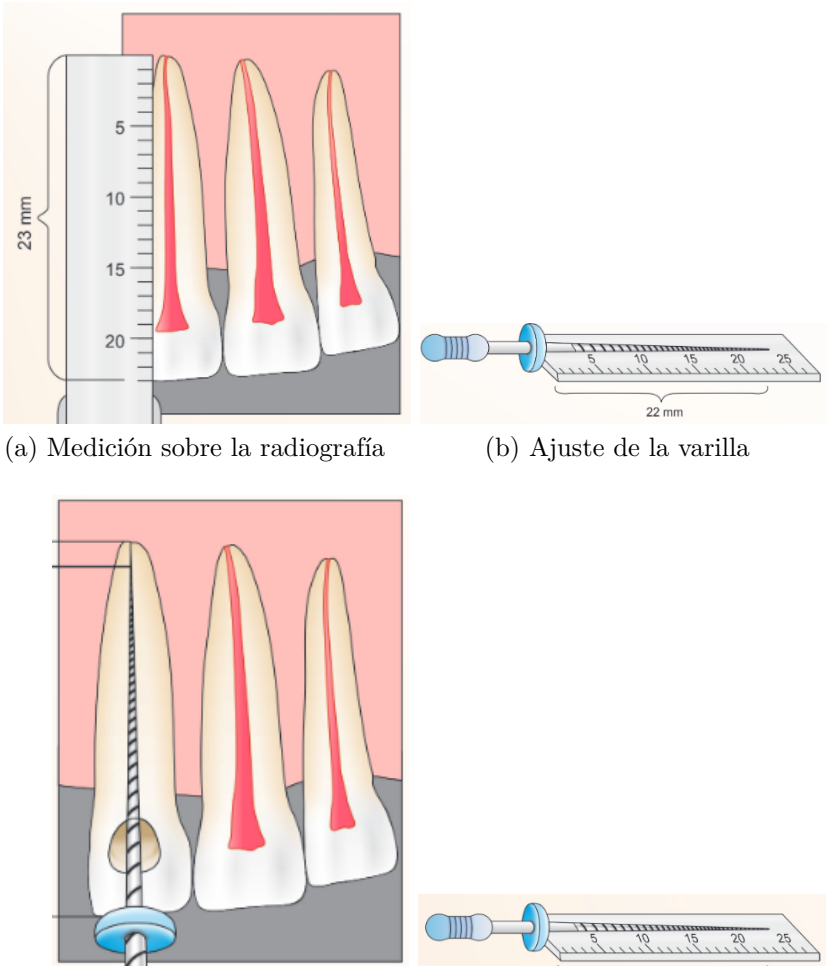
Los cuatro factores más comunes que dan lugar a una posible endodoncia son los siguientes [2]:

- **Caries profundas:** si una caries no se trata de forma correcta y a tiempo, puede acabar llegando a la pulpa del diente, dando lugar a una inflamación de la zona.
- **Traumatismos:** los golpes en los dientes pueden dar lugar a roturas de los tejidos internos del diente, produciendo en la mayoría de los casos una infección de la pulpa.
- **Bruxismo:** se produce con el choque de unos dientes con otros haciendo que se desgaste el diente hasta llegar a la pulpa.
- **Periodontitis:** es una enfermedad que ataca al tejido bucal y que puede llegar al interior del diente en caso de que no se trate a tiempo, produciendo una inflamación de la pulpa.

Los odontólogos, antes de realizar la endodoncia, necesitan conocer la longitud del diente implicado. Para dicho proceso, los odontólogos llevan a

cabo una radiografía de la zona involucrada y posteriormente miden sobre ella la longitud del diente (Figura 1.1a).

A continuación, introducen una varilla en el diente con un tope de longitud menor o igual a la obtenida de la radiografía (Figura 1.1b). De nuevo, se hace una radiografía del diente con la varilla y se calcula lo que falta al instrumento para llegar al final del diente (Figura 1.1c). Finalmente, se suman la longitud de la varilla introducida y la parte que faltaba hasta llegar al final del diente (Figura 1.1d).



(a) Medición sobre la radiografía (b) Ajuste de la varilla

(c) Radiografía con la varilla (d) Estimación final del tamaño del diente

Figura 1.1: Ejemplo Proceso Medición Diente [6]

Todo este proceso es manual. Por tanto, la probabilidad de cometer errores en el proceso de cálculo de medidas es bastante alta. Esta es una de

las causas que hacen que el uso del *deep learning* en la rama de la odontología permita obtener una mayor precisión, además de ser una herramienta de apoyo que facilita el trabajo de los odontólogos. Algunas de las principales tareas que puede realizar son [5]:

- Detección y segmentación de dientes y nervios.
- Detección de enfermedades.
- Fabricación de prótesis.
- Detección de implantes junto con la predicción de éxito del mismo.

Este proyecto busca automatizar el proceso de medición de la longitud del diente, de forma que a través de una radiografía se pueda dar un valor preciso de su longitud, evitando así cualquier error humano durante la medición y facilitando al odontólogo todo el proceso de medición del diente.

Para ello, el proyecto trabajará con técnicas del *deep learning* para ser capaz de segmentar dientes y nervios en las radiografías, y gracias a esta segmentación poder calcular la longitud del diente deseado.

1.1. Estructura

Este trabajo está formado por dos documentos principales: la Memoria y el Anexo. Cuya estructura es la siguiente:

Memoria

La memoria está formada por los siguientes puntos:

1. **Introducción:** puesta en contexto del problema al que se va enfrentar el proyecto. También se incluyen los documentos adjuntos junto con la estructura de cada uno de ellos.
2. **Objetivos del Proyecto:** descripción de los objetivos generales y técnicos.
3. **Conceptos Teóricos:** explicación de los diferentes conceptos teóricos que necesitan ser comprendidos para entender todo el proyecto.

4. **Técnicas y Herramientas:** contiene una breve descripción de todas las herramientas usadas y metodología seguida a lo largo de todo el proyecto.

5. **Aspectos Relevantes del Desarrollo del Proyecto:** exposición de aquellos aspectos más importantes surgidos durante el desarrollo.

6. **Trabajos Relacionados:** selección de algunos trabajos que tienen una cierta conexión con la endodoncia y el *deep learning*.

7. **Conclusiones y Líneas de Trabajo Futuras:** contiene las conclusiones finales junto con las posibles mejoras o actualizaciones que podría tener la aplicación en el futuro.

Anexo

El anexo está formado por cinco apéndices:

1. **Plan de Proyecto:** contiene el desarrollo que se ha llevado mediante las destinas tareas que se marcaban en cada *sprint*. Además, contiene el estudio de viabilidad del proyecto, a través del análisis de la viabilidad económica y legal.

2. **Especificación de Requisitos:** contiene los objetivos generales, los requisitos funcionales y los distintos casos de uso.

3. **Especificación de Diseño:** se centra en los diseños de datos, procedimental y arquitectónico.

4. **Documentación Técnica de Programación:** contiene la estructura de directorios, manual del programador, ejecución del proyecto y las pruebas del sistema.

5. **Documentación de Usuario:** describe los requisitos de usuario, la instalación y el manual de usuario.

Objetivos del proyecto

Este apartado contiene los objetivos del proyecto. Se pueden distinguir dos tipos: generales y técnicos. Dichos objetivos buscan cumplirse con el desarrollo del propio proyecto.

2.1. Objetivos Generales

Los objetivos generales del proyecto son:

- Investigar y aplicar nuevas funcionalidades del *deep learning* en la rama de la odontología.
- Desarrollar una aplicación web, sencilla de usar y con un fácil acceso para todo el mundo.
- Ayudar en el proceso de la endodoncia, de forma que la obtención de la longitud, actualmente hecha de forma manual, se pueda automatizar con la aplicación.
- Intentar conseguir que el error de la aplicación en la estimación de la longitud del diente no sea superior a 0.5 milímetros.

2.2. Objetivos Técnicos

Los objetivos técnicos del proyecto son:

- Crear una aplicación web capaz de calcular la longitud de un diente a través de una radiografía.

- Permitir el uso de la aplicación de dos modos distintos:
  - Acceder directamente al *notebook* en *Google Colab*.
  - Usar *ngrok* para acceder a la dirección web que contiene el *notebook* con la aplicación en Gamma.
- Usar el lenguaje de programación *Python* para realizar las pruebas pertinentes y la aplicación final, junto con las distintas librerías que existen en *Python*.
- Conseguir que los tiempos de espera en la aplicación sean los más breves posibles.
- Utilizar la plataforma de *GitHub*, de forma que se vea reflejado el trabajo incremental que se lleva a cabo en cada *sprint*.

2.3. Objetivos Personales

Los objetivos marcados a nivel personal son:

- Aprender sobre el proceso de la endodoncia, para poder investigar y desarrollar una aplicación que permita ayudar a los odontólogos en su trabajo.
- Aplicar los conocimientos adquiridos a lo largo del grado universitario.
- Mejorar los conocimientos de *Python* y aplicar aquellos ya adquiridos.
- Aprender sobre el *deep learning* para conocer sus posibilidades y poder aplicarlas a lo largo del proyecto.

Conceptos teóricos

3.1. Machine Learning y Deep Learning

La base del proyecto se debe al uso del *deep learning*, una rama del *machine learning*.

Machine Learning

El *machine learning*, o en español, aprendizaje automático, busca como principal objetivo desarrollar técnicas o algoritmos que permitan a un computador aprender, a partir de unos datos iniciales.

El proceso es similar al utilizado por animales y humanos para aprender en base a la experiencia, de forma que cuantos más datos se usen para entrenar un algoritmo, mejores resultados se obtengan.

El uso del *machine learning* se debe de emplear en problemas donde nos encontremos con muchos casos diferentes y con multitud de posibilidades y no se encuentre una forma de normalizar de forma matemática todas las posibles opciones.

De forma simple, el esquema [7] que sigue cualquier método de aprendizaje automático se puede ver en la Figura 3.1.

Deep Learning

El *deep learning*, en español, aprendizaje profundo, se inspira en el funcionamiento del sistema nervioso humano, donde existen multitud de redes neuronales especializadas en tareas distintas.

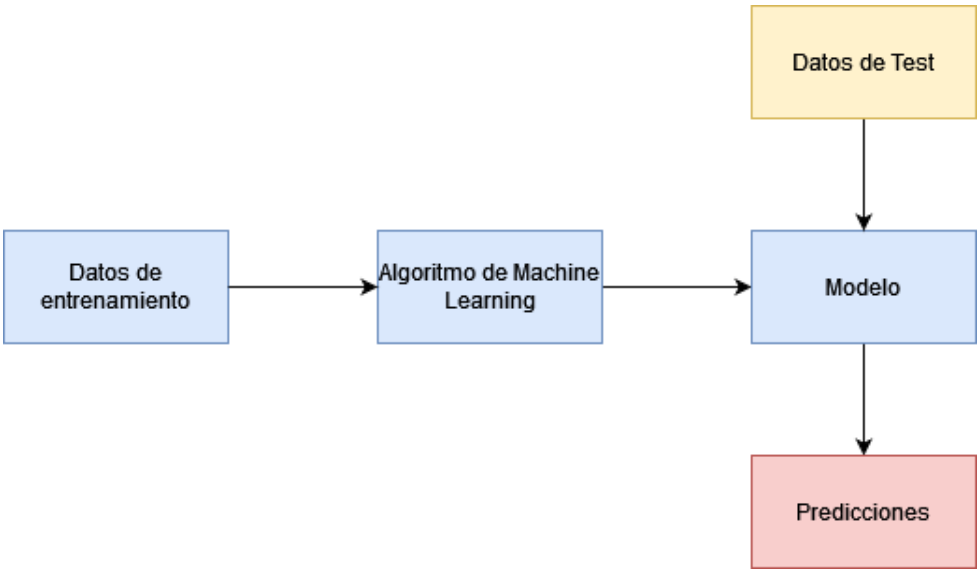


Figura 3.1: Esquema Algoritmos Machine Learning (Realización Propia).

Debido a ello, el *deep learning* crea arquitecturas neuronales de forma de que haya zonas encargadas en detectar ciertas características especificas en los datos. Esto ha supuesto grandes mejoras frente a otros tipos de *machine learning* basados en redes artificiales monolíticas.

3.2. Visión Artificial

La visión artificial [11], o en inglés, *computer vision*, tiene como objetivo identificar los elementos de las imágenes con el fin de extraer información útil con la que pueda trabajar un ordenador. Busca recrear el proceso que hacen las personas al observar un objeto y saber identificarlo de los demás, pero en este caso, dicho proceso lo hará de forma automática un computador.

Segmentación

La segmentación [8] es una técnica empleada en la visión artificial, la cual consiste en identificar y dividir una imagen en los diferentes objetos que la componen, de modo que, se puedan extraer aquellos con mayor interés.

3.3. Redes Neuronales Artificiales

Las redes neuronales artificiales son un conjunto de neuronas artificiales conectadas entre sí que permiten transmitir señales entre ellas. El principal objetivo de estas redes es resolver problemas como lo hacen los tejidos neuronales biológicos.

En una red neuronal artificial, dos neuronas están conectadas entre sí por un enlace. Dichos enlaces suelen tener un peso asociado, de forma que otras neuronas puedan activarse según los diferentes pesos de las señales que reciben.

La principal ventaja de estas redes es que aprenden solas y no es necesaria su programación paso por paso, ya que la propia red busca minimizar una función de pérdida de la red al completo, consiguiendo que sea capaz de encontrar el mejor camino para cada caso.

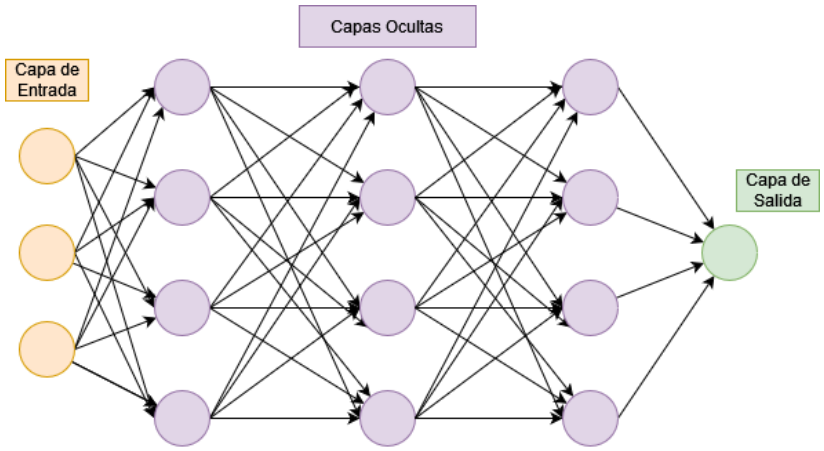


Figura 3.2: Ejemplo Red Neuronal Artificial (Realización Propia)

Existen dos tipos de redes neuronales artificiales:

- **Red Neuronal Prealimentada:** son un tipo de red neuronal que no permite bucles, es decir, la información siempre va de izquierda a derecha, como se aprecia en la Figura 3.2, de modo que la información siempre va de la capa de entrada a la de salida pasando por las capas ocultas, y nunca pueden volver hacia atrás.
- **Red Neuronal Recurrente:** son un tipo de red que permite bucles, de forma que si se puede volver hacia atrás en las capas. Esto permite a la red tener memoria de los datos recibidos por la neurona anterior.

Redes Neuronales Convolucionales

Las redes neuronales convolucionales [1], o CNN (del inglés *Convolutional Neural Network*), son el tipo más extendido de redes en computadores para poder darles la capacidad de reconocimiento de los elementos de imágenes y vídeos.

El funcionamiento de las mismas se debe a dos operaciones básicas: convolución y *pooling*.

Convolución

Esta operación utiliza una matriz llamada *kernel* que recorre cada uno de los píxeles de la imagen y obtiene un valor calculado (producto escalar) por el píxel que esta procesando junto con todos los que rodean al mismo, y de esta forma se va completando dicha matriz.

Cabe destacar que la nueva matriz obtenida tiene menor tamaño que la imagen original, como se puede ver en la Figura 3.3. También, este proceso se puede ir repitiendo varias veces obteniendo una matriz *kernel* de dimensiones menores.

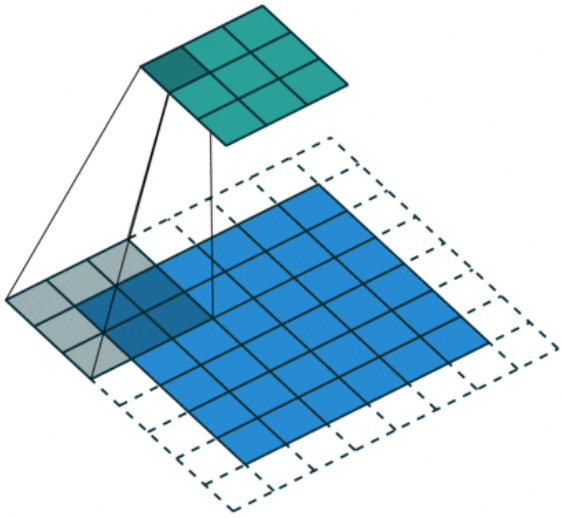


Figura 3.3: Operación de Convolución [3]

Pooling

La operación *pooling* se encarga principalmente de obtener dimensiones menores e ir obteniendo las características principales en cada caso.

Esta operación es muy sencilla. En primer lugar, se divide la imagen en una cuadrícula de un tamaño específico cada una. A continuación, por cada cuadrícula se devuelve un valor (generalmente el valor máximo o el valor medio).

La Figura 3.4 muestra un ejemplo de la operación *pooling* con una cuadrícula de 2x2 y devolviendo el tamaño máximo de cada celda.

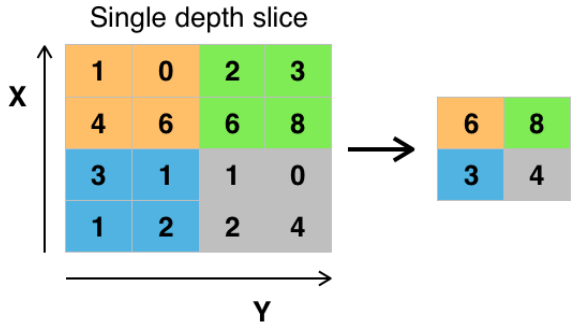


Figura 3.4: Operación de *Pooling* [4]

R-CNN

Las CNN son unas redes muy costosas de entrenar en aquellos casos donde las imágenes cuentan con muchas regiones diferentes. Es por ello, que surgen las R-CNN, del inglés *Regions with CNN features*.

La principal diferencia entre las R-CNN y las CNN, es que las primeras tienen la capacidad de detectar varios objetos en una imagen, mientras que el segundo tipo tan solo puede clasificar las imágenes pero sin saber exactamente donde se encuentra el elemento detectado.

Esto se debe a que en las redes R-CNN existe un algoritmo que divide en pequeñas regiones la imagen e intenta identificar en cada una de las regiones los objetos que sabe detectar. De este modo siempre sabe en que región o regiones se encuentra cada elemento.

Faster R-CNN

Las *Faster R-CNN* suponen una mejora de velocidad en el rendimiento y mejoras en la detección de objetos con respecto a su antecesora R-CNN. Los tiempos que tardan en realizar la predicción estas redes son de alrededor de 0.2 segundos, mientras que las R-CNN tiene un tiempo de entre 40-50 segundos, por lo que se ve claramente la mejora en cuanto a tiempo.

La red *Faster R-CNN* contiene un RPN (*Region Proposal Network*) que permite crear un conjunto de regiones. Esta regiones se crean a través de una capa nueva, denominada ROI (*Region Of Interest*), que permite extraer vectores de regiones y delimitar zona en las imágenes que podrían contener objetos. Todo esto permite obtener mejoras muy considerables en la detección.

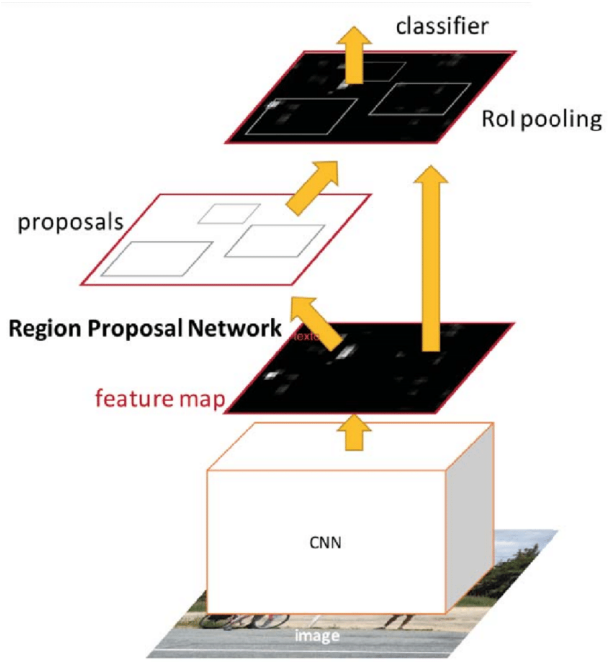


Figura 3.5: Funcionamiento Faster R-CNN [9]

Mask R-CNN

Las redes *Mask R-CNN* son una continuación de las redes *Faster R-CNN*. La principal mejora que trae es que este tipo de redes son capaces de obtener la máscara del objeto detectado, mientras que sus antecesoras tan solo podían identificar los elementos a través de una figura dibujada en la imagen.

Este es el tipo de red neuronal utilizada a lo largo del proyecto, ya que las detecciones por parte del modelo de *Detectron2* funcionan de este modo.

Detectron2 es una librería de *Python*, desarrollada por *Facebook*, basada en *PyTorch* para la creación de modelos y la segmentación de elementos.

3.4. COCO (Common Objects in COntext)

Para poder trabajar con *Detectron2* es necesaria tener la segmentación de las diferentes imágenes en formato COCO. Dicho formato, se basa en tener la segmentación en un JSON con diferentes atributos para poder reconstruir la parte segmentada sin la necesidad de almacenarla de forma original (máscara binaria), y poder trabajar con ella de forma más cómoda.

Existen varios tipos de anotaciones, pero en este proyecto se ha utilizado la formada por los siguientes atributos:

- `file_name:` contiene el nombre de la imagen a la que referencia dicha segmentación.
- `height:` contiene el valor correspondiente a la altura de la imagen.
- `width:` contiene el valor correspondiente a la anchura de la imagen.
- `annotations:` es un diccionario que contiene los siguientes atributos:
  - `bbox:` contiene los puntos máximos y mínimos en el eje horizontal y vertical, de modo que se pueda generar la caja que contiene en su interior la zona segmentada de dicha imagen.
  - `bbox_mode:` el formato que tiene el atributo anterior, es decir, cada valor se indica a que eje corresponde y cuál es el valor mínimo y cuál el máximo.
  - `segmentation:` lista que contiene todos los puntos que dan lugar mediante su unión a la zona segmentada del diente.
  - `category_id:` identificador de la categoría a la que se corresponde dicha segmentación.

3.5. IoU (Intersection Over Union)

La técnica IoU se utiliza principalmente para comprobar como de buena ha sido la segmentación producida por un modelo con respecto a la segmentación real.

En definitiva, mide cuanto están solapadas ambas segmentaciones, tomando valores comprendidos entre el 0 y el 1, siendo este último el mejor valor, indicando que ambas segmentaciones están totalmente solapadas, y 0 que no se solapan, y que por ello la predicción no ha sido muy buena.

A nivel matemático, la formula para obtener el valor de IoU entre dos segmentaciones A y B sería:

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Técnicas y herramientas

En este punto de la memoria se habla de las diferentes herramientas y técnicas utilizadas a lo largo del proyecto.

4.1. Metodología

A la hora de llevar a cabo el proyecto se ha seguido la metodología ágil del *Scrum*. Dicha metodología busca realizar tareas incrementales para cada reunión, denominado *sprint*. Los *sprints* tenían una duración de una semana, aunque en algunas situaciones especiales se han alargado o disminuido la duración de los mismos.

Para cada reunión se revisaban las tareas indicadas a llevar a cabo para ese *sprint*, además, se hablaba de las siguientes tareas a realizar.

4.2. Gestión del Proyecto

GitHub

GitHub[3] es un servicio web que permite almacenar el repositorio de un proyecto, e ir actualizándolo con las nuevas versiones, de forma que se queda registrado el progreso incremental que se ha ido llevando a cabo.

[3]GitHub: https://github.com/

Visual Studio Code

Con el fin de poder subir los avances del proyecto a *GitHub*, se ha usado *Visual Studio Code* que es un editor de código el cual permite sincronizarse a un repositorio propio de *GitHub* e ir subiendo las distintas versiones.

En mi caso he utilizado su versión *online*[4], ya que permite utilizar la aplicación sin la necesidad de instalar nada.

4.3. Herramientas

Anaconda

Anaconda[5] es una distribución libre de *Python* empleada principalmente en ciencia de datos y *deep learning*. Además, permite crear varios entornos con distintos paquetes y versiones, donde la instalación de los mismos se realiza de una forma muy sencilla.

Google Colab

Google Colab[6] es un editor de *Python* en el navegador web, de manera que se pueda escribir y ejecutar el código de los diversos *notebook* del usuario.

Se ha usado varias veces a lo largo del proyecto, sobre todo al principio, cuando no se tenía instalado *Detectron2*, debido a la facilidad de instalar paquetes en el propio *Colab*.

Además, la aplicación final se puede utilizar en *Google Colab*, debido a la facilidad que tiene para que otros usuarios puedan usar tus *notebook*.

Todo ello es mediante la suscripción gratuita, la cual incluye unos 12 GB de RAM y unos 80 GB de disco. Además, todas las ejecuciones son en los servidores de *Google*.

Jupyter Notebook

Jupyter Notebook[7] es un entorno web que permite generar diversos *notebook* donde poder ejecutar tu propio código.

[4] Visual Studio Code Online: https://vscode.dev/
[5] Anaconda: https://www.anaconda.com/
[6] Google Colab: https://colab.research.google.com/
[7] Jupyter Notebook: https://jupyter.org/

Cada *notebook* está formado por celdas, las cuales pueden tener diverso código y mostrar cada una de ellas el resultado de su ejecución. Uno de sus principales beneficios es que permite pasar los *notebooks* a otras personas y que puedan ver los resultados de la ejecución.

Ngrok

Ngrok[8] es una herramienta que nos permite lanzar un servidor local en un dominio de Internet, de forma que se pueda acceder al mismo fuera de la misma LAN.

Su uso ha sido para poder lanzar la aplicación de *Jupyter Notebook* en Gamma a un dominio de Internet que permita a los usuarios conectarse y poder usarla.

Tmux

Tmux[9] es un multiplexor de terminal, es decir, permite al usuario tener varias sesiones abiertas en la terminal de manera independiente, de modo que ninguna de ellas finalizará hasta que el usuario lo desee.

La aplicación principal de esta herramienta ha sido la de mantener siempre activo el proceso de *Jupyter Notebook* y *Ngrok*, para que la aplicación tenga un dominio web siempre activo.

PuTTY

PuTTY[10] es un cliente SSH con licencia gratuita que permite conectarse a servidores, iniciando sesiones en ellos, de forma remota.

En el proyecto se ha utilizado para poder conectarse a Gamma desde un ordenador con *Windows 10*, ya que este sistema operativo necesita de una aplicación externa para usar el cliente SSH.

4.4. Bibliotecas de Python

Para todo el proyecto se ha usado el lenguaje de programación *Python*. Las bibliotecas utilizadas a lo largo de todo el proyecto han sido:

[8] Ngrok: https://ngrok.com/
[9] Tmux: https://github.com/tmux/tmux
[10] PuTTY: https://www.putty.org/

NumPy

NumPy[11] permite crear vectores y matrices de grandes dimensiones con los que se puede usar un gran número de operaciones matemáticas.

IO

La biblioteca *IO* permite leer y escribir en los archivos del sistema. Además, permite controlar los permisos de solo lectura, escritura o ambas.

OS

La biblioteca *OS* permite utilizar las distintas funcionalidades del sistema operativo. En este proyecto se usa principalmente para dirigirse a las distintas rutas donde se encuentran los archivos.

OpenCV

OpenCV[12], también conocido como CV2, es una biblioteca centrada en la visión artificial en la parte de reconocimiento facial y de objetos. Será esta última el uso principal de dicha biblioteca.

Matplotlib

Matplotlib[13] se emplea principalmente para mostrar visualizaciones estáticas, animadas e interactivas en *Python*. Sus principales usos son los de crear gráficos en dos dimensiones o la de dibujar elementos sobre imágenes, siendo este último el uso principal de la biblioteca en el proyecto.

Jupyter Widgets

Jupyter Widgets permite añadir cierta interacción a los *notebooks*. Se pueden añadir diferentes widgets de todo tipo por todas las celdas, consiguiendo que el *notebook* sea mucho más atractivo y dinámico.

[11] NumPy: https://numpy.org/
[12] OpenCV: https://opencv.org/
[13] Matplotlib: https://matplotlib.org/

PIL

PIL[14] se dedica principalmente a la edición de imágenes con *Python*. Además, soporta los formatos más utilizados en multimedia.

Detectron2

Detectron2[15] es una biblioteca gratuita desarrollada por *Facebook AI* y que se centra en la detección de objetos. Es capaz de etiquetar varios objetos dentro de una sola imagen, además de permitir a los usuarios crear sus propios modelos.

Para la parte de detección de elementos a través de un modelo utiliza redes neuronales del tipo *Mask R-CNN*, las cuales ya se han explicado en el apartado 3.3.

SciPy

SciPy[16] es una biblioteca con herramientas y algoritmos matemáticos. Se basa principalmente en usar *NumPy*.

Gdown

Gdown[17] permite descargar archivos pesados de *Google Drive* al entorno de trabajo deseado.

JSON

JSON permite de una forma sencilla leer archivos con extensión JSON en *Python*. También, permite guardar archivos JSON a través de diccionarios.

Shutil

Shutil se encarga de poder copiar, mover y eliminar todo tipo de archivos del sistema en el que se esté utilizando.

[14] PIL: https://pypi.org/
[15] Detectron2: https://github.com/facebookresearch/detectron2
[16] SciPy: https://scipy.org/
[17] Gdown: https://github.com/wkentaro/gdown

ZIPfile

ZIPfile es una biblioteca que permite trabajar con archivos ZIP. Las posibles herramientas son crear, leer, o añadir un elemento a un ZIP.

PyTorch

PyTorch[18] es un paquete creado con el fin de realizar cálculos numéricos. Además, permite el uso de la GPU del sistema con el fin de acelerar los tiempos de cómputo.

Scikit-Image

Scikit-Image[19] contiene un gran número de algoritmos dedicados principalmente al procesamiento de imágenes y a la visión artificial.

4.5. Documentación

LaTeX

LaTeX[20] es un sistema de composición de textos centrado en la construcción de textos con alta calidad tipográfica.

Overleaf

Overleaf[21] es un editor colaborativo de LaTeX centrado en la escritura y edición de documentos. Todo ello de forma *online* y de forma gratuita.

Tables Generator

Tables Generator[22] es una web que permite, entre otras cosas, crear tablas y posteriormente exportarlas directamente a código en LaTeX, cosa que facilita de gran manera la creación de las mismas.

[18] PyTorch: https://pytorch.org/
[19] Scikit-Image: https://scikit-image.org/
[20] LaTeX: https://www.latex-project.org/
[21] Overleaf: https://www.overleaf.com/
[22] Tables Generator: https://www.tablesgenerator.com/#

Draw.io

Draw.io[23] es una web que permite a sus usuarios crear todo tipo de diagramas. Usado principalmente para poder crear diagramas UML o breves esquemas.

[23]Draw.io: https://app.diagrams.net/

Aspectos relevantes del desarrollo del proyecto

En este punto se recoge de manera detallada y descriptiva el proceso seguido durante todo el proyecto. El orden de los puntos es cronológico, donde se habla acerca de las decisiones tomadas, los problemas surgidos y las posibles alternativas.

Como aspecto importante, el objetivo inicial del proyecto era obtener la longitud del nervio, pero durante el transcurso del mismo, el odontólogo que da pie a este trabajo nos informó que lo importante era la longitud del diente, y no del nervio.

5.1. Obtención de imágenes para entrenar

Para empezar con el proyecto se contaba con tan solo 10 radiografías de la zona dental que se interesaba estudiar, además de los dos JSON correspondiente a cada radiografía que contenían los puntos del diente y del nervio segmentado.

El número de radiografías con las que se contaban era bajo, por lo que se decidió crear copias de las mismas, pero con diferentes modificaciones (proceso conocido como *data augmentation*) para poder trabajar con un número de radiografías de las que se podría esperar buenos resultados. El tutor José Miguel había diseñado un *notebook* encargado de realizar dicha función, de forma que, por cada radiografía original, se obtenían 11 (10 modificaciones y 1 original). Mi labor en dicho *notebook* fue únicamente el de llevar a cabo una modificación para conseguir que las nuevas radiografías tuvieran el mismo tamaño que la radiografía original.

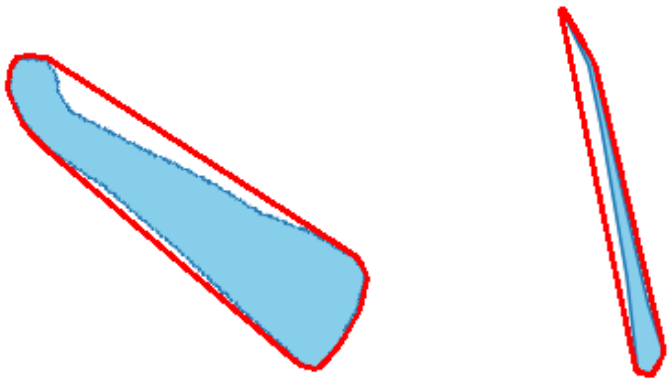
Obtención de puntos de las imágenes modificadas

Ahora que se contaban con las diferentes variaciones de las radiografías y de las diferentes máscaras del nervio y del diente, era turno de conseguir los puntos que delimitaban el borde de ambos elementos y almacenarlos en un JSON para poder trabajar con ellos.

Para ello se realizaron un total de 3 pruebas diferentes hasta que se obtuvo una forma de detectar los bordes de los dientes y nervios de forma precisa.

Convex Hull

La primera aproximación para obtener los puntos de las máscaras fue utilizar la técnica de *convex hull*, o en español, envolvente convexa, pero que debido la propia expresión de la misma los resultados no fueron buenos, ya que las partes curvas no las detectaba del todo bien, como se aprecia en las Figuras 5.1a y 5.1b.



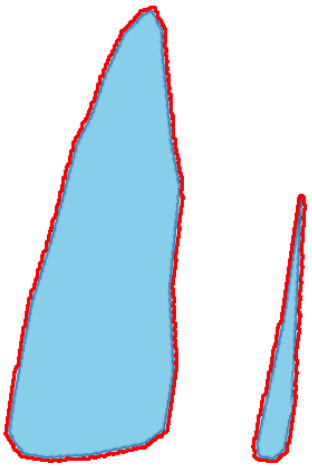
(a) Diente (b) Nervio

Figura 5.1: Ejemplo Resultados Convex Hull

OpenCV

Tras los resultados, no muy buenos, obtenidos por el *convex hull*, se estuvieron analizando diferentes alternativas. Una de ellas fue el uso de la biblioteca *OpenCV* la cual tiene funciones implementadas para detectar bordes en imágenes y devolver todos los puntos que lo forman.

Los resultados que se obtuvieron fueron bastante buenos, aunque los bordes no eran de todo uniformes, como se puede apreciar en las Figuras 5.2a y 5.2b.



(a) Diente (b) Nervio

Figura 5.2: Ejemplo Resultados Bordes OpenCV

Sckit-Image + OpenCV

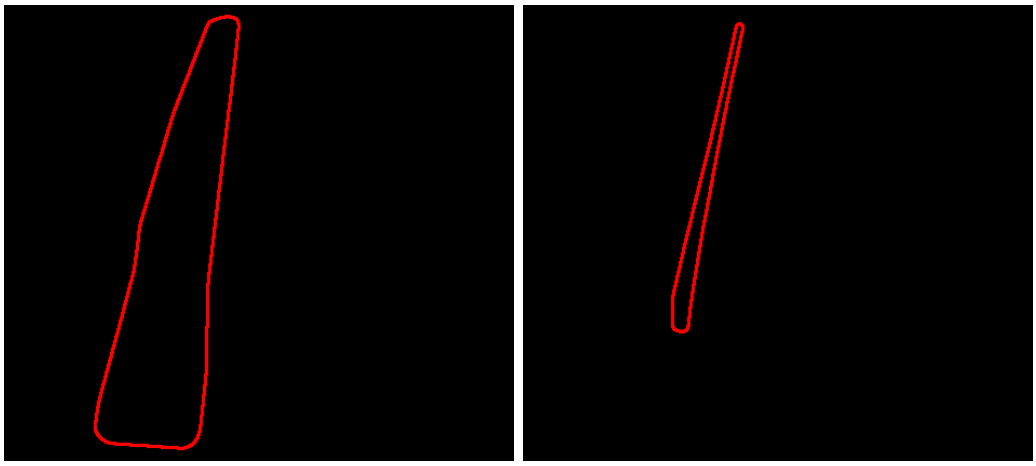
Pese a tener unos resultados bastante aceptables, se siguió investigando si existía alguna otra alternativa que mejorara los resultados anteriores.

Tras analizar varias opciones, se decidió usar la librería *Sckit-Image* para obtener una máscara de cada diente y nervio con la silueta del borde, donde los resultados eran bastante buenos.

Scikit-Image no devuelve los puntos correspondientes al borde, sino una máscara binaria con dicha silueta. Es por ello, que para obtener los puntos correspondientes se volvió a utilizar la librería *OpenCV*, donde ahora los bordes obtenidos sí que eran uniformes, como se puede ver en las Figuras 5.3a y 5.3b

Problemas con los elementos en los bordes

Para concluir con este punto, hay que hablar de un problema que había en algunas imágenes, donde el diente o el nervio estaba pegado al borde y la detección de los mismos no era correcta. La solución que se implementó fue la de añadir un borde blanco de 3 píxeles de ancho a todas las imágenes



(a) Diente (b) Nervio

Figura 5.3: Ejemplo Resultados Bordes Sckit-Image + OpenCV

para evitar que los elementos estuvieran tocando el borde y la detección de los mismos fuera la correcta.

5.2. Instalación Detectron2

Ahora, que ya se contaba con un gran número de radiografías, gracias al *data augmentation*, junto con sus JSON correspondientes a los puntos del nervio y del diente, era turno de instalar *Detectron2* [12].

Dicha tarea fue muy costosa, ya que no hay ninguna versión oficial para *Windows*, aunque haya gente que sí lo ha conseguido instalar.

Viendo que había gente que había tenido éxito, decidí crear un entorno en *Anaconda* e intentar instalarlo. Dicha tarea fue imposible, puesto que constantemente salían errores nuevos hasta que llegué a un punto en el que no podía hacer nada para solucionarlo.

Tras los intentos fallidos, se utilizó temporalmente *Google Colab* donde era sencillo de instalar y por el momento nos permitía seguir adelante. Pese a poder instalarse, los tiempos de ejecución eran demasiado elevados, por lo que se buscó como solución usar la máquina de la Universidad de Burgos, Gamma.

Finalmente, tras tener acceso a Gamma se consiguió instalar *Detectron2* y se pudo empezar a trabajar en dicha máquina con unos tiempos de ejecución bastante bajos y muchos mejores que en *Google Colab*.

5.3. Construcción del modelo

Ahora que ya se tenía instalado *Detectron2* era momento de trabajar con el mismo. Debido a que *Detectron2* es una biblioteca muy extensa y con muchas posibilidades. Se utilizó su *notebook*, que tienen de ejemplo en *Google Colab*[24], junto con una web[25] que tenía todos las funciones subdividas según sus funcionalidades, de forma que, según se necesitaba información de alguna cosa se recurría a dicha web, ya que tenía la documentación al completo.

Para obtener el modelo final se barajaron dos opciones:

- Un modelo para detectar a la vez diente y nervio: consistía en un único modelo que, para cada radiografía, detectara a la vez el diente y el nervio de la misma (será la opción utilizada para llevar a cabo la aplicación).

- Un modelo para detectar el diente, y posteriormente otro modelo para detectar el nervio: se basaba en tener un modelo para detectar el diente de la radiografía, recortar dicha parte que contenía el diente, y finalmente tener otro modelo para detectar el nervio (esta opción se utilizó únicamente en las pruebas, ya que para calcular la longitud del diente era más sencillo poder acceder a las dos predicciones a la vez).

Los pasos seguidos para obtener el modelo en ambas opciones han sido:

Distribución de las radiografías

El primer paso fue dividir las diferentes radiografías en tres bloques diferentes: entrenamiento, validación y test. Para distribuirlas se hizo de forma aleatoria para no influir de ninguna manera, tanto positiva como negativamente, al modelo.

Los porcentajes de cada bloque fueron: 60 % entrenamiento, 20 % test y otro 20 % validación.

[24] Ejemplo uso Detectron2: https://colab.research.google.com/drive/16jcaJoc6bCFAQ96jDe2HwtXj7BMD_-m5

[25] Web documentación Detectron2: https://detectron2.readthedocs.io/en/latest/index.html

Conversión JSON a COCO

Para poder trabajar con imágenes segmentadas, *Detectron2* necesita tener los datos en forma COCO, como ya se ha visto en la sección anterior 3.4.

Una vez ya se tenían los datos de forma correcta, se podían añadir cada dato a su correspondiente *DataSet* (entrenamiento, test y validación).

Entrenamiento

Para realizar el entrenamiento, y con ello, obtener el modelo, se siguió la configuración recomendada en el *notebook* de *Detectron2* mencionado anteriormente.

Además, se configuró el número de clases a predecir según las dos estrategias de modelo a conseguir. Para la opción de un único modelo tenía dos clases a predecir (diente y nervio), mientras que para la estrategia de dos modelos, cada uno de ellos tenía que predecir únicamente una clase (un modelo para el diente y otro modelo para el nervio).

Predicción

Una vez se obtuvo el modelo, era turno de realizar las predicciones. Para ello se usó el *dataset* correspondiente a las imágenes de test.

Adicionalmente, para analizar los resultados de las predicciones se utilizó IoU para conocer el porcentaje de similitud entre las predicciones y la segmentación real. Los valores mínimos, máximos y medios de los dientes y nervios obtenidos fueron los mostrados a continuación:

```
DIENTES:
La precisión media obtenida es de: 0.9284.
La precisión mínima obtenida es de: 0.8487.
La precisión máxima obtenida es de: 0.9793.

NERVIOS:
La precisión media obtenida es de: 0.8203.
La precisión mínima obtenida es de: 0.5532.
La precisión máxima obtenida es de: 0.9744.
```

5.4. Técnicas para medir la longitud del diente

La forma de medir la longitud del diente a través de las predicciones obtenidas por parte de *Detectron2* fue un proceso costoso ya que cuando se implementaba una técnica, se le mostraba al odontólogo y nos comentaba que no era muy adecuada.

Es por ello, que se han implementado un total de cuatro técnicas diferentes para medir el tamaño del diente.

Técnica 1 - Longitud máxima del diente

La primera aproximación fue obtener la distancia máxima entre la fila superior del diente y la inferior. Aparentemente, no da malos resultados como se puede apreciar en la Figura 5.4, pero en aquellos dientes con cierta curvatura o inclinados la distancia no se calculaba de forma correcta.

Figura 5.4: Ejemplo Resultado Técnica 1

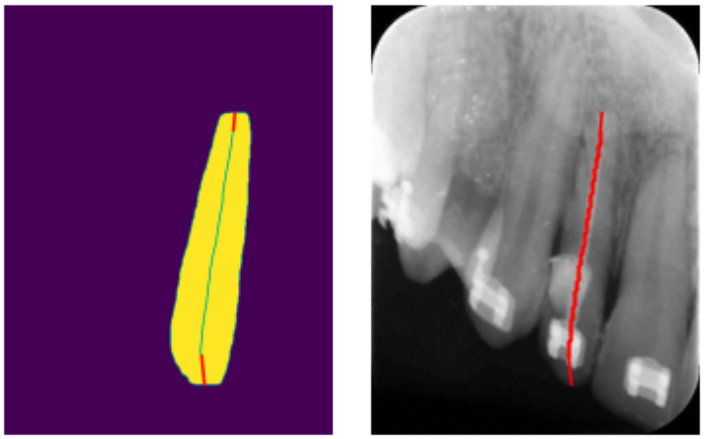
Técnica 2 - Skeleton del diente

La siguiente técnica se basa en utilizar una función de la biblioteca *Scikit-Image*, llamada *skeleton* que devuelve una máscara binaria con los puntos centrales de un objeto.

Por tanto, lo que hacía era calcular los puntos centrales del diente y posteriormente alargarlo hasta la fila superior e inferior del diente, Figura 5.5a.

Tras analizar los resultados obtenidos por dicha función, había casos en los que los puntos hacían una pequeña bifurcación al ensancharse el diente, por lo que se empleó una función muy similar, llamada *thin* que eliminaba aquellos extremos que añadía *skeleton*.

El problema de esa técnica es que alargar hasta la fila inicial y final del diente no era una buena idea, ya que había casos en los que los dientes, al tener una inclinación, el inicio y el fin del mismo, no se correspondía con dichas filas, como se puede apreciar en la Figura 5.5b.



(a) Ejemplo Thin Diente (b) Resultado Final

Figura 5.5: Ejemplo Resultado Técnica 2

Técnica 3 - Thinning del nervio

La técnica 3 busca mejorar los problemas de la técnica anterior, de forma que en vez de aplicar el *thinning* al diente se aplica al nervio, de forma que se consiguen unos puntos más precisos al encontrarse dicho nervio en la zona media del diente.

El operador morfológico *thinning* funciona de igual forma que el *skeleton*, solo que sobre los puntos obtenidos, por cada iteración, se borran los correspondientes a los bordes de forma que se finaliza el proceso una vez se pierde la conectividad.

Por otro lado, para alargar los puntos obtenidos hasta el diente, ahora se realiza cogiendo el primer y el último punto, se calcula la recta que pasa por ambos puntos y se alargar hasta el borde del diente, Figura 5.6a. De

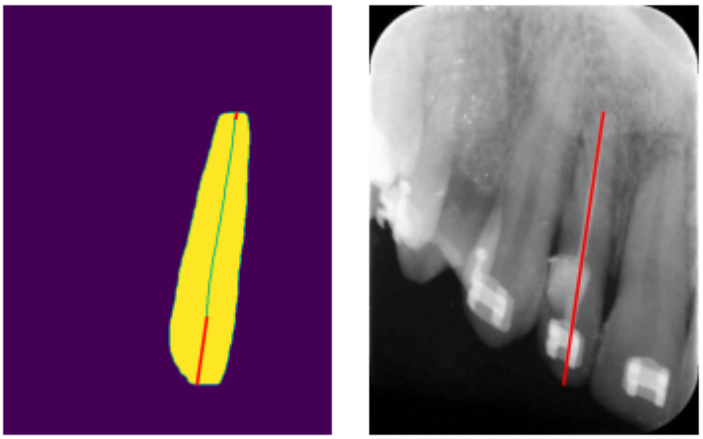
esta manera se calcula la longitud del diente teniendo en cuenta su parte central independientemente de la inclinación del mismo.

La ecuación de la recta que pasa por dos puntos, $A(x_a, y_a)$ y $B(x_b, y_b)$ es:

$$\frac{x - x_a}{x_b - x_a} = \frac{y - y_a}{y_b - y_a}$$

Podemos hallar la pendiente de la recta anterior, m, evaluando:

$$m = \frac{y_b - y_a}{x_b - x_a}$$



(a) Thin Alargado hasta el Diente

(b) Resultado Final

Figura 5.6: Ejemplo Resultado Técnica 3

Técnica 4 - Perpendiculares en el thin del nervio

Tras mostrar los resultados obtenidos por parte de la técnica anterior al odontólogo, nos indicó que las mediciones eran correctas, menos en aquellos casos donde el nervio tenga una cierta curvatura, ya que no se tendría en cuenta.

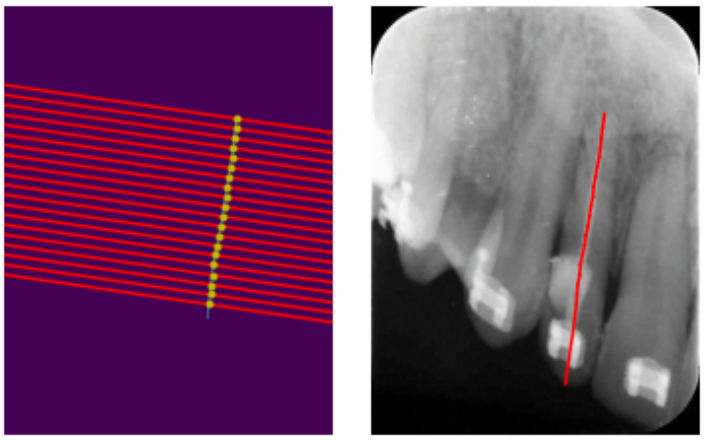
Por tanto, esta técnica busca solucionar ese problema. Para ello, se realizarán perpendiculares a la recta obtenida por la técnica 3 en el *thin* obtenido del nervio, de forma que se obtendrán los puntos de intersección, Figura 5.7a, y se irán sumando las distancias entre los puntos de corte.

Nuestro objetivo ahora es obtener rectas perpendiculares a las calculadas en el apartado anterior. Como sabemos el punto por el que han de pasar estas perpendiculares, basta calcular la pendiente, donde basta calcular:

$$m_{\perp} = \frac{-1}{m}$$

Donde m es la pendiente de la recta obtenida en la técnica 3.

Finalmente, a la suma total de los puntos de intersección se le sumará la longitud del final del *thin* del nervio al punto final del diente (obtenido en la técnica 3), y de la misma manera, se sumará la distancia del punto inicial del diente (también obtenido en la técnica anterior) al punto inicial del *thin*.



(a) Puntos de Intersección (b) Resultado Final

Figura 5.7: Ejemplo Resultado Técnica 4

Validación de los resultados

Para poder validar la calidad de los resultados obtenidos se le solicitó al odontólogo que nos pasara la longitud de cada uno de los dientes de las radiografías que nos había proporcionado, junto con las longitudes, en milímetros, del ancho y del largo de cada radiografía para poder convertir la longitud de píxeles a milímetros.

Pero a partir de aquí surgieron varios problemas. El primero de ellos fue que el odontólogo nos pasó la longitud real del diente y no la de la radiografía, que es siempre superior a la real, por lo que siempre los resultados obtenidos eran mayores.

El segundo problema era que, pese a saber lo que debería de medir en milímetros cada diente, los tamaños de las radiografías eran diferentes. Es decir, las radiografías que nos pasó el odontólogo no seguían las proporciones que deberían de tener en realidad, y pese a preguntar el porqué ocurría esto, el odontólogo nunca encontró una respuesta.

Por esta razón, los resultados obtenidos son siempre con unos errores relativos muy altos e inviables en la endodoncia (ya que el odontólogo quería que el error máximo fuera de 0.5 milímetros, puesto que si fuera superior se podrían causar daños irreparables en el paciente).

De modo que, nos hemos centrado más en comprobar que las segmentaciones de los dientes y nervios son buenas, junto con la técnica para calcular la longitud del diente y no tanto en el valor obtenido, ya que son valores que se alejan de la realidad y siempre van a dar resultados incorrectos.

5.5. Desarrollo de la aplicación final del proyecto

El último paso era desarrollar la aplicación final, puesto que ya que se contaba con el modelo encargado de realizar las predicciones y con una buena técnica para calcular la longitud del diente.

Para desarrollar la aplicación, inicialmente se barajó el instalar alguna biblioteca que permitiera lanzar una GUI desde *Jupyter Notebook*. Se instalaron *PySimpleGUI* y *EasyGUI*, ya que tras analizarlas eran interfaces sencillas y fáciles de implementar.

Pero al tener alojada la aplicación en Gamma, nos encontramos con un problema , y es que en un servidor no se pueden lanzar GUI al no tener un *display* asignado, como se aprecia a continuación:

```
TclError: no display name and no $DISPLAY enviroment
variable
```

Viendo que no se podían usar ninguna GUI, se optarón por dos opciones distintas:

- *Google Colab*: hacer que la aplicación final fuese un *notebook* en dicha plataforma, donde se trabajaría con el modelo obtenido en Gamma.

- *Ngrok*: emplear dicha herramienta para dar acceso a la aplicación de forma remota desde Internet y sin necesidad de estar conectado a Eudoram.

Pese a barajar ambas opciones, se llevaron a cabo las dos, ya que serían dos formas distintas de poder transmitir la aplicación del proyecto.

Google Colab

Para poder dar acceso en un *notebook* a otro usuario es muy simple, ya que permite obtener un enlace a través del cual cualquier persona puede acceder y hacer funcionar la aplicación.

Ngrok

Para poder utilizar *Ngrok* era necesario instalar dicha herramienta en Gamma y crearse una cuenta en su web oficial[26] para poder obtener el toquen, Figura 5.8, que permite dar de alta la aplicación, Figura 5.9.

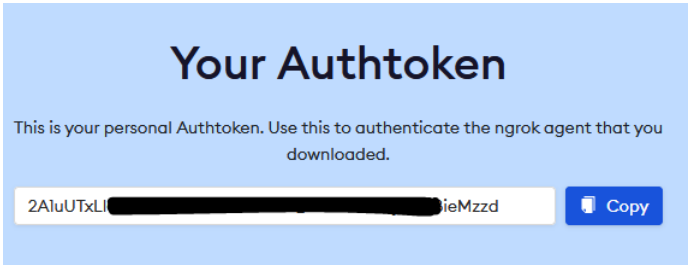


Figura 5.8: Token de Usuario en Ngrok.

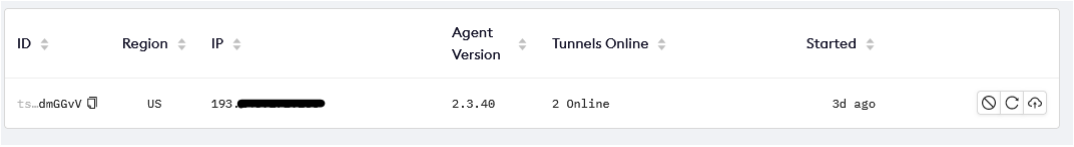


Figura 5.9: Ngrok con Web Activa.

El uso del toquen es el de poder vincular una web pública a tu cuenta personal de *Ngrok* para poder controlarla y gestionarla de una forma más cómoda. Además, también te permite tener acceso a las ventajas de tener una suscripción de pago, en caso de tener una activa.

[26] Ngrok: https://ngrok.com/

Finalmente, tras implementar correctamente el modelo y sus diversas funcionalidades, los resultados que devuelve la aplicación se pueden apreciar en la Figura 5.10.

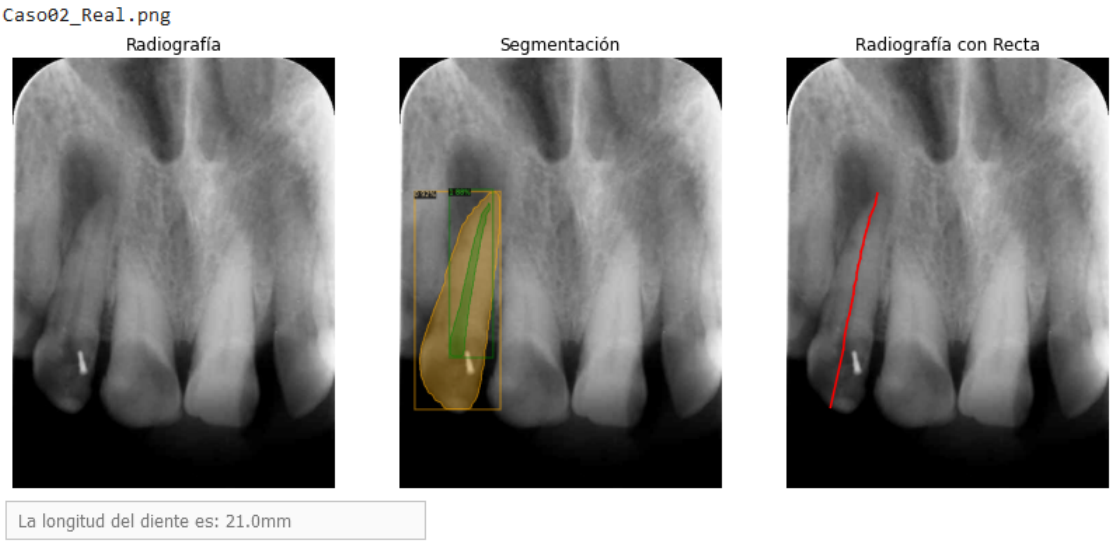


Figura 5.10: Resultado Aplicación Final.

Acceso a la aplicación

Los accesos a la aplicación del proyecto son:

- *Google Colab*: https://bit.ly/3R86dJc
- *Ngrok*:
  - Web: https://7b9a-193-146-172-150.ngrok.io
  - Contraseña acceso: `TfgUbu2122`

Trabajos relacionados

6.1. Application of Deep Learning in Dentistry and Implantology

Autores: Dae-Young Kang, Hieu Pham Duong y Jung-Chul Park

Este trabajo [5] recoge los resultados de la investigación de algoritmos existentes de *deep learning* en la rama de la odontología. Es una colaboración entre las Universidades de Dankook y de Vietnam, publicado en el año 2020.

Lo primero que resaltan es que en la odontología se requiere una precisión muy elevada, ya que se trabaja con medidas milimétricas y un error humano puede tener consecuencias irreparables. Es por ello, que defienden que la inteligencia artificial y el *deep learning* suponen una gran mejora de precisión y de detección de anomalías que pueden ser indetectables a la vista humana.

En la parte de investigación se habla sobre la diversidad de aplicaciones que tiene el *deep learning* en la odontología, y una de ellas es la segmentación e identificación de dientes (tarea que se ha llevado a cabo en ese proyecto), aunque no dice nada de cálculo de longitud de dientes.

La conclusión final del artículo es que se espera que la utilización de algoritmos de *deep learning* mejore los tiempos de trabajo de los dentistas y los resultados de los diferentes tratamientos, ayudando a los dentistas en casi todos los aspectos prácticos, como detección de dientes, detección y clasificación de enfermedades, evaluación de prótesis y reducción de errores de precisión.

6.2. Software Second-Opinion

La aplicación *Second-Opinion*[27] es un *software* de pago que permite a los odontólogos que la utilizan cargar sus radiografías y la aplicación, a través de la inteligencia artificial, calcula o muestra los valores que los odontólogos necesitan.

Las principales tareas que realiza la aplicación son:

- Detección de enfermedades.
- Validación de implantes.
- Segmentación de dientes.
- Obtención del canal radicular.

Pese a ser un software muy completo, no incluye el cálculo de la longitud del diente, cosa que sí que realiza la aplicación de este proyecto.

Para finalizar, este *software* ha sido aprobado por la FDA (*Food and Drugs Administration*, Administración de Alimentos y Medicamentos en español, en Estados Unidos)[28], para que los dentistas del país puedan utilizarlo con los pacientes. Además, esta aplicación ha sido la primera de este tipo en aprobarse en Estados Unidos por lo que se puede ver como es una rama en la que aún queda mucho por explorar.

[27] Second-Opinion: https://www.hellopearl.com/products/second-opinion

[28] Artículo aprobación software: https://www.hellopearl.com/press-release/fda-clears-worlds-first-ai-software-to-read-dental-x-rays

Conclusiones y Líneas de trabajo futuras

7.1. Conclusiones

Este proyecto surge de la mano del odontólogo Álvaro Zubizarreta Macho, quien propuso la creación de una aplicación que le ayudara en la endodoncia al ser un proceso totalmente artesanal y con una gran posibilidad a fallo.

El primer problema que hubo fue crear un modelo con tan pocas radiografías, pero, pese a ello, se solucionó transformando las radiografías prestadas, gracias al *data augmentation*.

Tras conseguir el modelo se mostraron los primeros resultados de la segmentación de dientes y nervios al odontólogo, quien reconoció que eran bastantes buenos. Este profesional fue de ayuda en lo relacionado con las técnicas de cálculo, permitiendo encontrar la forma más óptima. Lamentablemente, los resultados obtenidos no se han podido probar en su totalidad debido a diversos problemas en la resolución y tamaño de las radiografías, los cuales no han sido solucionados a lo largo del proyecto.

A pesar de esto, los resultados son gratamente satisfactorios debido a que el odontólogo afirmó que las predicciones del modelo eran correctas. Sin embargo, hubiera sido más gratificante poder analizar la calidad de los resultados. Durante este proceso se ha disfrutado del aprendizaje del *deep learning* en *Python*, debido a ser algo de lo que no se tenía mucho conocimiento. Además, se han conocido una gran cantidad de herramientas útiles de cara al futuro, como *Ngrok* y *tmux*.

Finalmente, cabe destacar que se ha desarrollado una aplicación sencilla y fácil de usar para usuarios alejados del mundo de la programación y los

notebooks, permitiendo que esta herramienta pueda ser usada por cualquier cliente.

7.2. Líneas de trabajo futuras

En mi opinión, unas mejoras que se podrían realizar al proyecto en el futuro serían:

1. **Cálculo de longitud de los demás dientes de la radiografía:** actualmente se devuelve la longitud del diente y del nervio con mayor precisión por parte del modelo de *Detectron2*. Es por ello, que se podría calcular la longitud de todos los dientes que aparezcan en las radiografías con el fin de poder ayudar a los odontólogos en caso de que necesiten más longitudes.

2. **Aumentar el número de radiografías con el que entrenar al modelo:** cómo se ha comentado en puntos anteriores, el proyecto tan solo ha contado con 10 radiografías, las cuales han tenido que ser transformadas para poder contar con un gran número de imágenes con las que entrenar al modelo. Pero realmente, no dejan de ser 10 imágenes reales con sus variaciones. Es por ello, que sería buena idea aumentar considerablemente el número de radiografías para conseguir un modelo más robusto y capaz de detectar mejor los dientes y nervios en aquellos casos más extraños.

3. **Desarrollar una aplicación web:** la aplicación final se ha desarrollado en un *notebook*, por lo que sería buena idea transportarla a una aplicación web de forma que resulte más cómodo el acceso a los clientes. Evitando, por ejemplo, que tengan que esperar tiempos de espera en la instalación de *Detectron2* en *Google Colab*.

4. **Añadir nuevas funcionalidades a la aplicación:** el proyecto se ha centrado únicamente en crear una aplicación capaz de calcular la longitud del diente, pero en el futuro se podrían implementar más funcionalidades con las que ayudar en las endodoncias o incluso en alguna otra rama de la odontología.

Bibliografía

[1] Raúl Castilla Bravo et al. Reconocimiento de logotipos de marcas mediante redes neuronales de convolución (CNN). 2021.

[2] Clínica Dental Riosdent. https://clinicadentalriosdent.com/, 2020.

[3] Wikimedia Commons. File:convolution arithmetic - padding strides odd.gif — wikimedia commons, the free media repository, 2020. [Online; fecha de acceso 27-Junio-2022].

[4] Wikimedia Commons. File:max pooling.png — wikimedia commons, the free media repository, 2021. [Online; fecha de acceso 27-Junio-2022].

[5] Dae-Young Kang, Hieu Pham Duong, and Jung-Chul Park. Application of deep learning in dentistry and implantology. 2020.

[6] Librería Servicio Médico. http://www.libreriaserviciomedico.com/files/9676, 2020.

[7] Tom M Mitchell and Tom M Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[8] Nora La Serna Palomino and Ulises Norberto Román Concha. Técnicas de segmentación en procesamiento digital de imágenes. *Revista de investigación de Sistemas e Informática*, 6(2):9–16, 2009.

[9] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[10] Ilson José Soares and Fernando Goldberg. *Endodoncia. Técnica y fundamentos.* Ed. Médica Panamericana, 2002.

[11] Wikipedia. Visión artificial — wikipedia, la enciclopedia libre, 2022. [Internet; descargado 24-junio-2022].

[12] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.