Proyecto de Grado

Presentado ante la ilustre Universidad de Los Andes como requisito parcial para obtener el Título de Ingeniero de Sistemas

# Desarrollo de una aplicación móvil basada en redes neuronales para el diagnóstico de lesiones cariosas tomando en cuenta el Sistema internacional para la Detección y Evaluación de Caries (ICDAS)



Por

Br. Julio Mejias

Tutor: Prof. Jormany Quintero

Febrero 2021



UNIVERSIDAD
DE LOS ANDES
VENEZUELA

# Desarrollo de una aplicación móvil basada en redes neuronales para el diagnóstico de lesiones cariosas tomando en cuenta el Sistema internacional para la Detección y Evaluación de Caries (ICDAS)

Br. Julio Mejias

Proyecto de Grado — Sistemas Computacionales, 52 páginas
Escuela de Ingeniería de Sistemas, Universidad de Los Andes, 2021

**Resumen:** La caries dental es una enfermedad multifactorial que ocurre por el desbalance poblacional de las bacterias cariogénicas que se encuentran en la boca. Con la continua evolución de la tecnología, las herramientas de diagnóstico han ido evolucionando incorporando los teléfonos inteligentes en las actividades clínicas. Las aplicaciones móviles que soportan al odontólogo en tareas de diagnóstico son escasas y limitadas, en este sentido el propósito de este trabajo fue desarrollar un prototipo de aplicación móvil basada en Android para el diagnóstico visual de lesiones cariosas usando el Sistema Internacional para la Detección y Evaluación de Caries (*ICDAS*) y redes neuronales convolucionales. La aplicación móvil desarrollada implementó la librería *TensorFlow Lite* para la ejecución del modelo. El reconocimiento implementó el modelo *YOLO versión 4* basado en la implementación de *Darknet* junto con un banco de 933 imágenes obtenidas de la internet y diferentes cátedras de la Facultad de Odontología de La Universidad de Los Andes. Las pruebas de la aplicación se realizaron en cuatro equipos diferentes para evaluar el comportamiento. La identificación para las lesiones de grado 0 y grado 6 presentaron un desempeño aceptable (>60%) y un desempeño regular (<60%) para la identificación de las lesiones grado 1,2,3,4 y 5. La aplicación móvil desarrollada demostró un buen comportamiento a pesar de la baja cantidad de imágenes usadas en el entrenamiento.

**Palabras clave:** detección, diagnóstico, caries, ICDAS, imagen, dispositivo móvil, aplicación, telemedicina.

Este trabajo fue procesado en LaTeX.

# Índice

# Índice de Tablas

# Índice de Figuras

# Capítulo 1

# Introducción

La caries dental es una enfermedad muy común que afecta a toda la población mundial, la cual si no es tratada a tiempo puede generar graves problemas de salud. En el año 2005 un grupo de especialistas odontólogos crearon el Sistema Internacional de Detección y Diagnóstico de Caries (ICDAS por sus siglas en ingles) para identificar la caries y sus diferentes estadíos Estai and Tennant (2016b).



Adicionalmente se han publicado varios trabajos sobre la detección y diagnóstico de la caries y sus diferentes etapas haciendo uso de dispositivos electrónicos, los cuales han sido un gran aporte al área odontológica mejorando los tratamientos y la prevención de esta enfermedad. Este capítulo incluye los antecedentes, planteamiento del problema, alcance, objetivos, metodología y la justificación con los cuales se desarrollará el proyecto para el desarrollo de una aplicación móvil para el diagnóstico de caries dental.

## 1.1 Antecedentes

El estudio y la preservación de la salud bucal ha sido un factor muy importante en el campo médico desde la invención del cepillo de dientes. El cuidado de los dientes ha ido evolucionando constantemente, y uno de los factores que ha resaltado notoriamente en estos últimos años es la búsqueda de la detección temprana y tratamiento adecuado de las caries dental. En la actualidad se puede encontrar investigaciones relacionadas con

la detección de caries haciendo uso de dispositivos electrónicos como lo expresa Estai and Tennant (2016b) en su trabajo *"A proof of concept evaluation of a cloud based store and forward telemedicine app for screening for oral disease"*, cuyo objetivo principal fue evaluar una aplicación de telemedicina basada en la nube, para la detección de enfermedades bucales. Esta aplicación se compone en dos partes el desarrollo de un método para el almacenamiento de información en la nube y la creación de una aplicación para dispositivos Android donde a través, del uso de la cámara de los teléfonos se captura la imagen para posteriormente enviarla a la nube. Seguidamente esta información es analizada por un personal calificado para detectar el estado de la salud bucal de los pacientes de una manera más eficaz. Cabe resaltar que para su diagnóstico no se usó el ICDAS como guía o patrón de comparación.

Por el contrario, Bottenberg and Jablonski-Momeni (2016) hace uso del ICDAS en su trabajo *"Comparison of occlusal caries detection using the icdas criteria on extracted teeth or their photographs"*. Este estudio consistió en determinar si el uso de fotografías de superficies oclusales en lugar de dientes extraídos para la detección de caries puede ser útil en estudios o educación. Para dicho objetivo se usó un panel de observadores, los puntajes de ICDAS en dientes o fotografías fueron evaluados contra el estándar de oro histológico. El resultado para ambos casos fue equivalente. En este trabajo de investigación se implementó el ICDAS, sin embargo, no se implementó una aplicación móvil para llevar a cabo dicha comparación; es por esto, que se puede pensar que quizá la implementación de la telemedicina es muy precisa para la detección de caries.



En el trabajo de investigación hecho por Estai and Tennant (2016a) se centró en investigar y comparar qué tan precisos son los métodos de telemedicina en comparación con los métodos habituales. Los autores concluyeron que a pesar de la ausencia de trabajos de investigación que implementen la telemedicina para la detección de caries, aquellos que cumplieron con el criterio QUADAS-2 domains, fueron catalogados con un desempeño de diagnóstico aceptable en la detección de la caries dental.

Cabe destacar que al momento de diagnosticar una enfermedad actualmente en el campo de la telemedicina se han implementados diversas técnicas para el reconocimiento de imágenes, entre estas destaca el uso de redes neuronales. Este algoritmo es usado ampliamente para diagnóstico por imágenes, un ejemplo de esto se

puede encontrar en el trabajo realizado por Quintero-Rojas and González (2021), el cual consistió en el desarrollo de una aplicación móvil para el reconocimiento de lesiones en la mucosa bucal implementando una red neuronal convolucional que reconoce lesiones.

Otra aplicabilidad de las redes neuronales se puede encontrar en el trabajo de Araújo and Campilho (2017), en el cual se demostró que el uso de las redes neuronales convolucionales para la deteccion del cáncer de mama pueden lograr un 95,6% de éxito en la detección temprana de esta enfermedad, permitiendo a médicos y pacientes iniciar un tratamiento adecuado.

La implementación de aplicaciones móviles para la detección de la caries dental es una herramienta que facilitaría el diagnóstico acertado basado en el uso del ICDAS con multiples aplicaciones en el campo de la salud y aprovechando las ventajas que ofrecen los telefonos móviles actualmente.

## 1.2 Planteamiento del Problema



La caries dental es una enfermedad mundial que afecta a la población sin distinción de sexo y edad. Según estimaciones publicadas en el estudio sobre la carga mundial de morbilidad 2016, las enfermedades bucodentales afectan a la mitad de la población mundial (3580 millones de personas) O.M.S (2020), y la caries dental en dientes permanentes es el trastorno más prevalente.

El tratamiento dental es costoso, y representa una media del 5% del gasto total en salud y el 20% del gasto medio directo en salud en la mayoría de los países de altos ingresos. ICDAS II es un sistema que tiene del 70% al 85% de sensibilidad y una especificidad del 80% al 90% en detectar caries, en dentición temporaria y permanente; dependiendo esta diferencia por el grado de entrenamiento y calibración del personal examinador.

Es por esto que resultaría de gran utilidad una aplicación móvil para dispositivos Android que implemente el ICDAS de tal forma que, cualquier examinador sea capaz de utilizarla eliminando así, la necesidad de entrenamiento previo y las limitantes que las faltas de dicho entrenamiento ocasiona. Como se desea diseñar una aplicación en la cual, el usuario no necesitará un entrenamiento previo, es fácil de utilizar y de gran

ayuda al momento de realizar levantamientos epidemiológicos por la unificación de criterios y disminución de sesgos en la toma de datos.

## 1.3 Justificación

Es de suma importancia para el odontólogo tener la capacidad de hacer un diagnóstico temprano de caries dental, con el fin de realizar tratamientos mínimamente invasivos que no comprometan la integridad del dental.

Dado que la caries dental es una enfermedad progresiva que cuenta con diferentes estadíos, es importante para los profesionales realizar detecciones tempranas para revertir el proceso de desmineralización con tratamiento preventivo.

Es por esto que se hace de notoria importancia la preparación de los odontólogos para el diagnóstico de esta enfermedad en estos estadíos. En consecuencia la aplicación como herramienta de diagnóstico les sería de gran utilidad para calibrar y mantener un mismo criterio al momento de diagnosticar en qué etapa se encuentra la enfermedad.



## 1.4 Alcance

Con la implementación de este proyecto se desarrolló una aplicación móvil, que mediante el uso de la cámara del teléfono celular permita la identificación de los distintos niveles de caries en los dientes según el ICDAS. La aplicación tiene entre de 50% a 85% de precisión y funciona de manera local en el dispositivo móvil.

Por otro lado, esta aplicación tiene el proposito de dar una detección y diagnóstico rápido de las caries y el nivel en el que se encuentran, frecuentemente se necesita de una persona capacitada para realizar dicho diagnóstico.

## 1.5 Objetivos

### 1.5.1 Objetivo General

- Desarrollar una aplicación móvil basada en redes neuronales para el diagnóstico visual de lesiones cariosas usando el Sistema Internacional para la Detección y

Evaluación de Caries (ICDAS)

### 1.5.2 Objetivos Específicos

- Definir el lenguaje de programación y *framework* para el desarrollo de la aplicación.
- Elegir la red neuronal a usar en la investigación.
- Entrenar la red neuronal para la detección de lesiones cariosas sobre las estructuras dentarias reconocidas.
- Diseñar la interfaz de usuario acorde para el uso de la aplicación.
- Conectar la red con la interfaz y realizar pruebas preliminares.

## 1.6 Metodología



Esta fase comprende las actividadesque fueron necesarias para cumplir con los objetivos descritos.

- Se estudió toda la documentación correspondiente al área de detección de caries en los dientes y sus diferentes tipos según el Sistema Internacional para la Detección y Evaluación de Caries (ICDAS).
- Se estudió toda la documentación correspondiente al uso y funcionamiento de una red neuronal para el reconcomiendo de imágenes.
- Se estudió toda la documentación correspondiente al uso y funcionamiento del entorno de desarrollo integrado (IDE) Android Studio y el lenguaje de programación Java para el desarrollo de la aplicación.
- Se comenzó el desarrollo de un producto mínimo viable (M.V.P por sus siglas en inglés) de la aplicación.
- Se probó el M.V.P en busca de errores y fallas.

- Se corrigieron los errores y fallas del M.V.P.
- Se realizó la versión final de la aplicación.
- Se usó la aplicación para recolectar datos de su precisión en la detección de caries en los dientes según el ICDAS.

# Capítulo 2

# Marco Teórico y Conceptual

En este capítulo se presentan conceptos teóricos básicos necesarios para comprender el trasfondo de la aplicación móvil desarrollada, junto con algunos conceptos usados en el área odontológica.



## 2.1 Red Neuronal Artificial

Existen numerosas formas de definir a las redes neuronales; desde las definiciones cortas y genéricas hasta las que intentan explicar más detalladamente qué son las redes neuronales. González (2019)

1. Una red neuronal es un nuevo sistema para el tratamiento de la información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano: la neurona, la cual está formada por unidades de procesamiento que intercambian datos o información y se utiliza para para reconocer patrones, incluyendo imágenes, manuscritos y secuencias de tiempo (por ejemplo: los tipos de caries en los dientes ).

2. Las redes neuronales artificiales son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico.

Ahora que se tiene un concepto más claro de lo que es una red neuronal y una red neuronal artificial, se puede definir el tipo de modelo de la red neuronal.

- El modelo de tipo biológico: Este comprende las redes que tratan de simular los sistemas neuronales biológicos, así como las funciones auditivas o algunas funciones básicas de la visión.

- El modelo dirigido a aplicación: Este modelo no tiene por qué guardar similitud con los sistemas biológicos. Su arquitectura está fuertemente ligada a las necesidades de las aplicaciones para la cual es diseñada (este será el modelo a seguir para el desarrollo de este proyecto de investigación).

Habiendo definido que es una red neuronal y sus modelos se puede analizar con mayor detalles sus características básicas y de funcionamiento.

El perceptrón es la unidad básica de una red neuronal, que toma como base la neurona, la comparación se puede apreciar en la Figura 2.1:



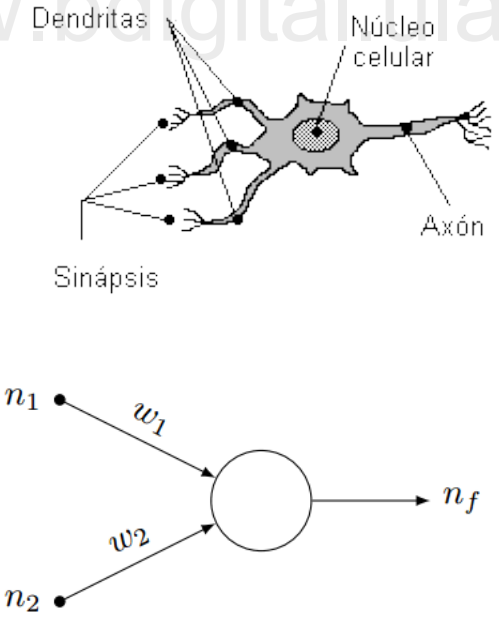



Figura 2.1: Diagrama básico de una neurona y un perceptrón

En la Figura 2.2 se puede observar la estructura de una neurona, que tiene múltiples entradas conocidas como dendritas, el soma es el núcleo donde se ejecutarán algunos procesos que trabajan a partir de todo lo que llega como entrada y el resultado se envía como respuesta a través del axón hasta los terminales para el proceso llamado sinapsis que activará otra neurona. En este sentido, el perceptrón emula estos procesos con una estructura similar, el centro hará una suma algebraica de las entradas y luego que pasarán a una función de activación que procesa las entradas para luego emitir una única respuesta en su salida que puede estar conectada a una o varias neuronas subsecuentes.

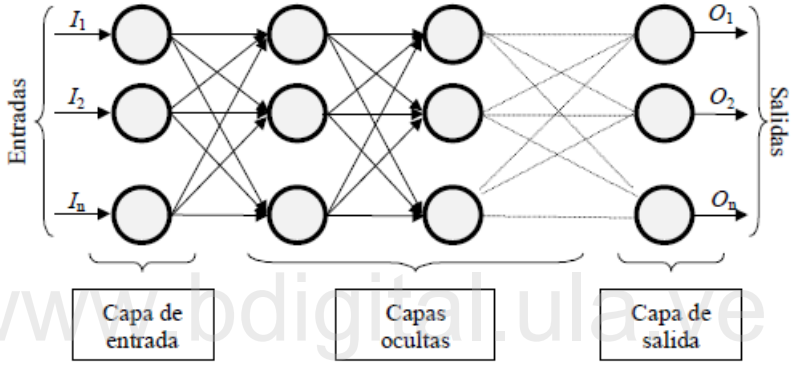




Figura 2.2: Diagrama de una red neuronal totalmente conectada

## 2.2 Red neuronal convolucional

Las redes neuronales convolucionales *(CNN)* son una variante de un perceptrón multicapa que, al igual que el perceptrón, tiene su origen en investigaciones biológicas, particularmente en la forma en que la corteza visual percibe las imágenes que observamos, en síntesis, mostraron que la corteza visual de los mamíferos, identifica pequeñas porciones de una imagen vista y los procesa en una estructura jerárquica. Las pequeñas porciones se unen hasta elaborar la imagen, y es esta entonces, la base de este tipo de redes. Una red neuronal convolucional se caracteriza por extraer características en diversas partes de la imagen, estas características, luego de extraídas, son procesadas y filtradas para poder luego formar parte de la información de entrenamiento. Generalmente, estas redes no están completamente conectadas (no son densas) sino hasta en las

últimas capas González (2019) . El diagrama básico de una red neuronal convolucional (*CNN*) se muestra en la Figura 2.3:

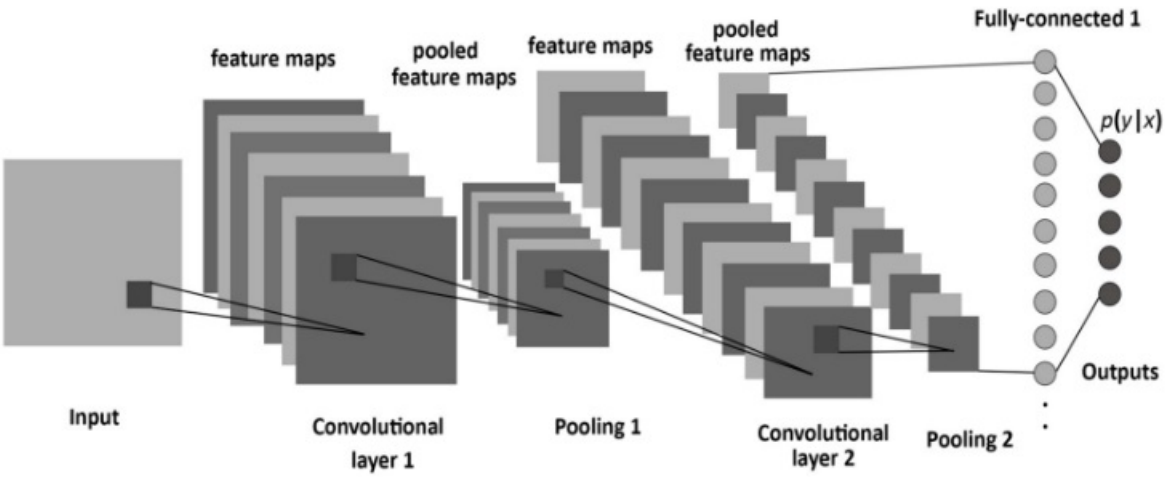



Figura 2.3: Arquitectura de una red neuronal convolucional

Otro par de conceptos que serán de mucha utilidad son los de:



- Aprendizaje profundo: es un campo perteneciente a la inteligencia artificial que se desenvuelve en estas 'capas' o niveles. Las primeras capas reconocen detalles concretos, mientras que las últimas capas reconocen patrones más abstractos y generan un resultado final.apd (2018).

- Aprendizaje supervisado: el aprendizaje supervisado es un conjunto de técnicas que permite realizar predicciones futuras basadas en comportamientos o características analizadas en datos históricos etiquetados.Calvo (2017)

  Una etiqueta no es más que la salida que ha mostrado el conjunto de datos para datos históricos, ya conocidos.

  La predicción obtenida es representada por medio de una función donde las entradas representan las características analizadas y la salida representa la variable que se quiere predecir.

  Esta función de salida es de tipo numérico en problemas de regresión y de tipo categórico en los problemas de clasificación.

## 2.3 Transferencia de aprendizaje

La transferencia de aprendizaje se muestra como una de las técnicas más importantes del aprendizaje profundo para el aprendizaje automático en inteligencia artificial. Consiste, en esencia, en aprovechar una gran cantidad de información relacionada con la resolución de un problema y utilizarla sobre otro distinto, pero que comparta ciertas características con este. En otros términos, modificar patrones ya entrenados (o redes neuronales) para reconocer ciertas características, para poder reconocer otras similares. La visión artificial y el procesamiento del lenguaje natural son las misiones para las que reservan los mejores resultados González (2019).

## 2.4 *Tensorflow*

*Tensorflow* es una herramienta de computación numérica creada por Google. Actualmente se usa para cientos de proyectos de aprendizaje atuomático, tanto dentro como fuera de Google. *Tensorflow* tiene un excelente equilibrio de flexibilidad y escalabilidad. La flexibilidad permite a los desarrolladores e investigadores probar ideas nuevas en un tiempo corto. Aunque muchos piensan que *Tensorflow* sólo sirve para redes neuronales, realmente te permite hacer todo tipo de cosas como aprendizaje por refuerzo. La escalabilidad permite que los modelos desarrollados puedan ser usados por millones de usuarios. *Tensorflow*, además, es portable, por lo que puede ser utilizado en todo tipo de dispositivos.



Por otro lado, la API de alto nivel de *Tensorflow* provee modelos de topología complejos listos para comenzar a probar. Esto supone una ventaja cuando se prueban ideas, o cuando se requiere poder de cómputo para algunas operaciones, en ambos casos, es la herramienta ideal.

La API de redes neuronales de *Tensorflow* está disponible en versiones de Android de 8.1 en adelante, esta API funciona como un acelerador de hardware que mejora el proceso de inferencia, gráficamente, la arquitectura de este acelerador de hardware se aprecia en la Figura 2.4 González (2019).

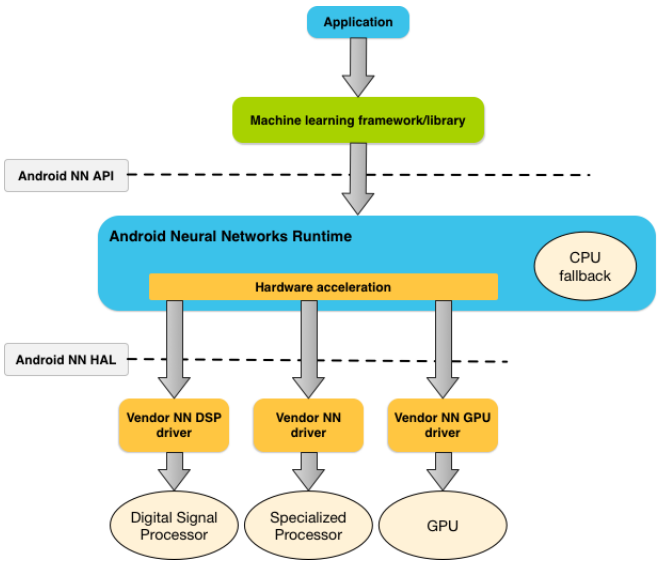



Figura 2.4: Arquitectura de la Api de *Tensorflow*

## 2.5 *YOLO*

El algoritmo *You Only Look Once* (*YOLO*), es un sistema de código abierto del estado del arte para detección de objetos en tiempo real, el cual hace uso de una única *CNN* para detectar objetos en imágenes. Para su funcionamiento, la red neuronal divide la imagen en regiones, prediciendo cuadros de identificación y probabilidades por cada región; las cajas son ponderadas a partir de las probabilidades predichas. El algoritmo aprende representaciones generalizables de los objetos, permitiendo un bajo error de detección para entradas nuevas, diferentes al conjunto de datos de entrenamiento. Bochkovskiy et al. (2020) La Figura 2.5 muestra la configuración de la arquitectura de *YOLO V4*



### 2.5.1 Arquitectura *YOLO V4*

- Columna: utiliza *CSPDarknet53* como modelo de extractor de funciones para la versión de *GPU*.

- Cuello: utilizan la combinación de pirámides espaciales *(SPP)* y la red de agregación de rutas *(PAN)*. Este último no es idéntico al PAN original, sino una versión modificada que reemplaza la adición con un concatenamiento. En la

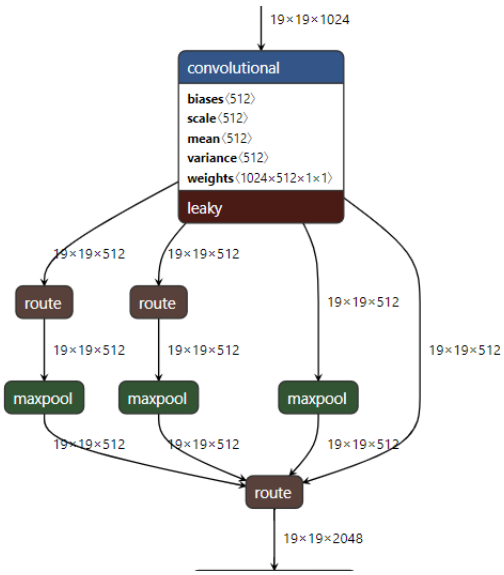



Figura 2.5: *SPP* obetenido del archivo yolov4.cfg

Figura 2.6 se muestra la configuración para *PAN* modificado y en la Figura 2.7 la configuración para *PAN*



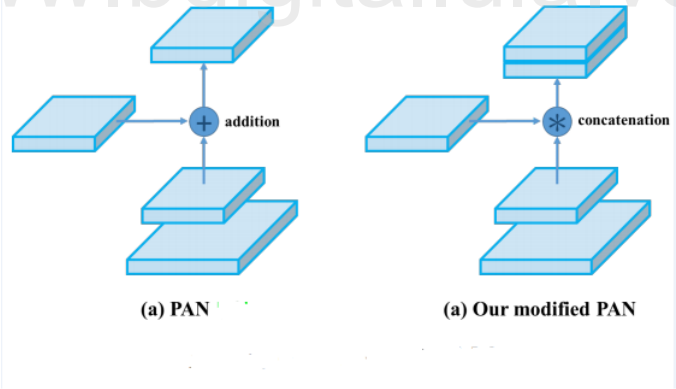



Figura 2.6: PAN modificado

### 2.5.2 *CSPDarknet53*

Es un tipo de red neuronal convolucional la cual se define como una red parcial de etapas cruzadas y su principal aplicación es como columna del *YOLO* para identificación de objetos, imágenes o video en tiempo real, la Figura 2.8 muestra la arquitectura de la

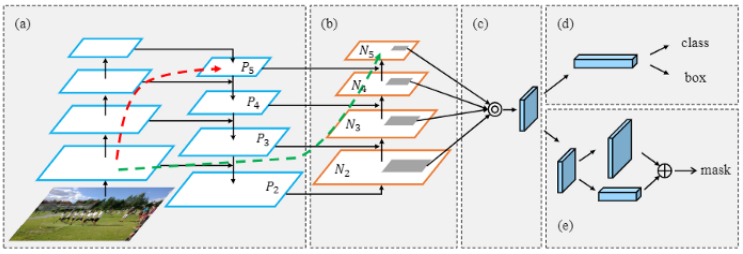



Figura 2.7: *Path Aggregation Network (PAN)*

red *CSPDarknet53* .Wang et al. (2020)

**CSPDarknet53 网络结构图**

| | Type | Filters | Size | Output |
|---|---|---|---|---|
| | Convolutional | 32 | 3x3 | 256x256 |
| | Convolutional | 64 | 3x3/2 | 128x128 |
| | CrossStagePartial | | | |
| 1x | Convolutional | 32 | 1x1 | |
| | Convolutional | 64 | 3x3 | |
| | Residual | | | 128x128 |
| | Convolutional | 128 | 3x3/2 | 64x64 |
| | CrossStagePartial | | | |
| 2x | Convolutional | 64 | 1x1 | |
| | Convolutional | 64 | 3x3 | |
| | Residual | | | 64x64 |
| | Convolutional | 256 | 3x3/2 | 32x32 |
| | CrossStagePartial | | | |
| 8x | Convolutional | 128 | 1x1 | |
| | Convolutional | 128 | 3x3 | |
| | Residual | | | 32x32 |
| | Convolutional | 512 | 3x3/2 | 16x16 |
| | CrossStagePartial | | | |
| 8x | Convolutional | 256 | 1x1 | |
| | Convolutional | 256 | 3x3 | |
| | Residual | | | 16x16 |
| | Convolutional | 1024 | 3x3/2 | 8x8 |
| | CrossStagePartial | | | |
| 4x | Convolutional | 512 | 1x1 | |
| | Convolutional | 512 | 3x3 | |
| | Residual | | | 8x8 |
| | Avgpool | | Global | |
| | Connected | | 1000 | |
| | Softmax | | | |

Made By Jaredzzz (1143020206@qq.com)



Figura 2.8: Definición de la arquitectura de la *CSPDarknet53*

## 2.6 Android Studio

Android Studio es el entorno de desarrollo integrado oficial para el desarrollo de aplicaciones Android (Conoce Android Studio 2019). Fue oficialmente presentado en 2013

como reemplazo oficial de su antecesor Eclipse, sus primeras versiones 0.1.x datan de mayo del 2013 y su primera versión estable fue presentada en diciembre de 2014, siendo Android Studio v1.0. A partir de esta fecha se ha actualizado constantemente hasta la versión actual que es la v4.1.1. Está basado en el software IntelliJ IDEA que por sí mismo es un IDE cuya primera versión fue publicada en 2001. Es una herramienta muy completa que proporciona todo lo necesario para desarrollar un aplicación móvil para Android, dentro de sus características se puede encontrar: González (2019)

- Sistema de compilación automatizado (basado en Graddle).
- Emulador con variedad de dispositivos y tamaños de pantalla.
- Instant Run para aplicar cambios mientras la aplicación se ejecuta sin la necesidad de compilar un nuevo APK.
- Integración con *github*.
- Editor visual para ver en todo momento la parte visual de la aplicación.
- Soporte para la herramienta en la nube de Google (Google Cloud Platform).



## 2.7 Aplicación móvil

El concepto de aplicación no es muy reciente, pues ha ido evolucionando a medida que el teléfono móvil se fue introduciendo en la vida cotidiana del ser humano. Uno de los cambios que revolucionó las aplicaciones móviles fue la introducción de internet a los dispositivos móviles. A partir de allí surgió el nacimiento de plataformas como iOS de Apple Inc. que introdujeron el iPhone (2007) y un mercado en el que desarrolladores externos tienen la posibilidad de publicar su contenido en la Appstore. Este lanzamiento también introduce un mercado de aplicaciones llamado para entonces Android market que hoy en día se llama Google Play González (2019). Una aplicaion móvil se estructura de la siguiente forma:

- *Manifest*: El archivo manifest de una aplicación, lleva por nombre: *AndroidManifest.xml*, que, en esencia, es el lugar con los permisos necesarios para ejecutar una

aplicación, permisos que pueden incluir, pero no están limitados a: uso de hardware (camara, microfono, *GPS*), acceso especial a otras aplicaciones(contactos, galera multimedia).

- Carpeta *Java*: En ésta carpeta se encuentran todos los archivos relacionados con cada activity o actividad, como código fuente y clases auxiliares. Uno de los archivos que se ven al crear un proyecto es el archivo *MainActivity.java*, que contiene la lógica esencial para que el usuario interactúe con la aplicación.

- Carpeta de recursos *res*: Esta carpeta contiene todos los recursos adicionales para la completación de una aplicación móvil, esto es, la parte gráfica y las cadenas de texto que son generalmente usadas en cada actividad, en cada título, menú, lista desplegable, etc. Un archivo XML[1] es un lenguaje de marcado que, a diferencia del HTML[2], otro lenguaje de etiquetado que da forma y estilo a los datos que contiene, este defíne la estructura y significado de los datos que contiene gracias a la forma en que se etiquetan,y es ampliamente utilizado gracias a su portabilidad.



- *Activity*: Una actividad en una aplicación es toda pantalla que se puede observar y con la que se puede interactuar, por ejemplo las llamadas, el directorio de contactos, los mensajes de texto (o en general cualquier aplicación de mensajería). Una actividad por lo general abarca toda una ventana. La ventana que se ve inmediatamente se abre una aplicación, se llama Main Activity. La primera vez que se crea un proyecto para una aplicación (El clásico Hola Mundo) Android Studio se encarga de enlazar los archivos necesarios para que la actividad pueda ejecutarse. Para mejor comprensión la documentación de Android Studio ilustra mediante un diagrama de flujo los cambios de estado de una actividad gracias a los diferentes tipos de llamado, como en la Figura 2.9

[1] del inglés *eXtensible Markup Language*
[2] *HyperText Markup Language*

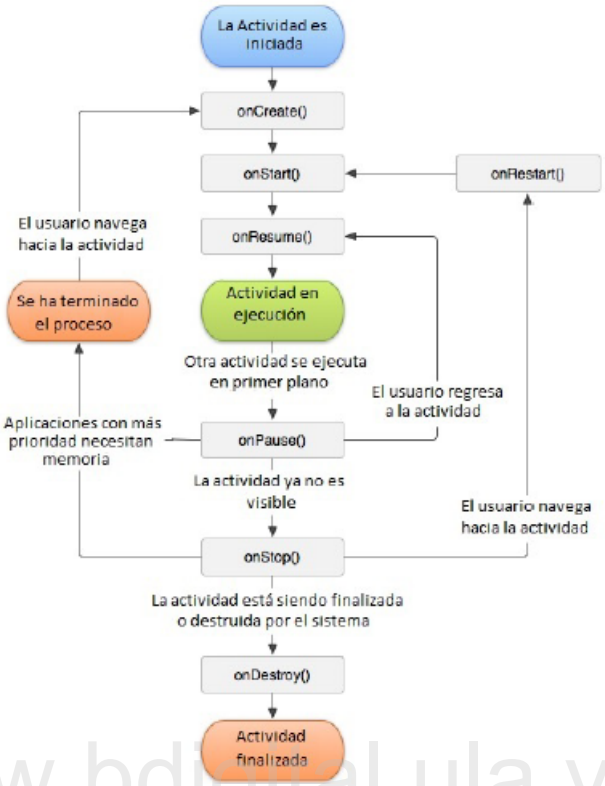





Figura 2.9: Ciclo de vida de una actividad

## 2.8 Caries dental

La caries dental es una enfermedad multifactorial que ocurre por el desbalance poblacional de las bacterias cariogénicas que se encuentran en la boca, con la capacidad de producir ácidos que atacan la superficie del diente o esmalte. Esto puede resultar en una cavidad o agujero en el diente, si la caries dental no se trata, puede causar dolor, una infección e incluso la pérdida del diente.

Una vez erupcionados los dientes, las personas de todas las edades, desde los niños hasta las personas mayores, pueden tener caries, por aumentar alguno de los factores predisponentes los niños pequeños corren el riesgo de padecer “caries de la primera infancia”, algunas veces llamada caries del biberón, que es la caries dental grave en los dientes de leche. Muchas personas mayores tienen las encías retraídas. Esto permite que las bacterias que causan caries, que se encuentran en la boca, tengan contacto con la raíz del diente. Esto puede causar caries en las superficies expuestas de las raíces

dentales Colgate (2021) .

## 2.9 ICDAS (*International Caries Detection and Assessment System*)

Es un sistema internacional de detección y diagnóstico de caries, consensuado en Baltimore, Maryland. USA en el año 2005, para la práctica clínica, la investigación y el desarrollo de programas de salud pública.

El objetivo fue desarrollar un método visual para la detección de la caries, en fase tan temprana como fuera posible, y que además detectara la gravedad y el nivel de actividad de la misma.

El sistema tiene 70 al 85 % de sensibilidad y una especificidad de 80 al 90 %; en la detección de caries, en dentición temporaria y permanente; dependiendo esta diferencia por el grado de entrenamiento y calibración del personal examinador. Marcelo (2020)

En la Figura 2.10 se pueden apreciar las reglas y parámetros seguidos según Marcelo (2020) para determinar el nivel de las lesiones cariosas en los dientes las cuales fueron implementadas.

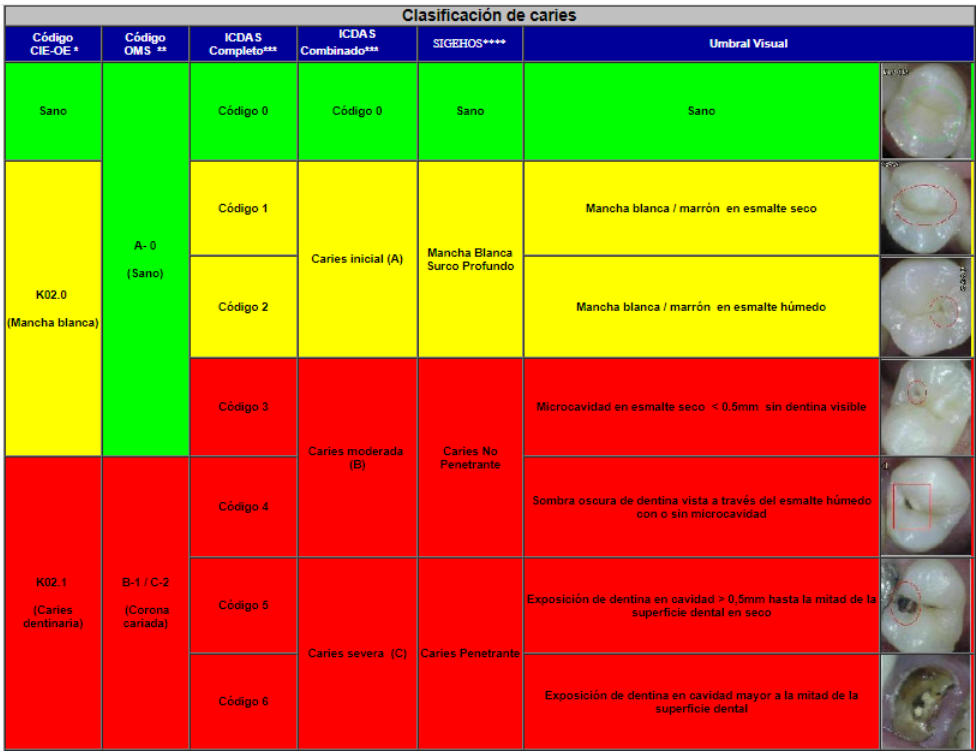



Figura 2.10: Tabla de los niveles de las lesiones cariosas según el ICDAS



## 2.10 Métricas para evaluación

En esta sección se especifican las métricas que se usaron en el proyecto desarrollado con la finalidad de comprender un poco más los resultados que se obtuvieron.

**Precisión**: Se refiere a la dispersión del conjunto de valores obtenidos a partir de mediciones repetidas de una magnitud. Cuanto menor es la dispersión mayor la precisión. Se representa por la proporción entre el número de predicciones correctas (tanto positivas como negativas) y el total de predicciones. Se calcula como: Arce (2021)

$$Precisión = \frac{VP}{(VP + FP)} \tag{2.1}$$

**Exactitud**: Se refiere a lo cerca que está el resultado de una medición del valor verdadero. En términos estadísticos, la exactitud está relacionada con el sesgo de una estimación. También se conoce como Verdadero Positivo (o "True positive rate"). Se representa por la proporción entre los positivos reales predichos por el algoritmo y

todos los casos positivos.

En forma práctica la Exactitud es la cantidad de predicciones positivas que fueron correctas y se define como: Arce (2021)

$$Exactitud = \frac{(VP + VN)}{(VP + FP + FN + VN)} \tag{2.2}$$

**Sensibilidad**: Es la proporción de casos positivos que fueron correctamente identificadas por el algoritmo. Se calcula: Arce (2021)

$$Sensibilidad = \frac{VP}{(VP + FN)} \tag{2.3}$$

**Especificidad**: Se trata de los casos negativos que el algoritmo ha clasificado correctamente. Expresa cuan bien puede el modelo detectar esa clase. Se calcula: Arce (2021)

$$Especificidad = \frac{VN}{(VN + FP)} \tag{2.4}$$

**Valor F1**: Es una métrica de desempeño que se usa para buscar un equilibrio entre precisión y sensibilidad cuando la distribución de las clases no está balanceada e indica qué tan preciso y sensible es un modelo respecto a una clase específica. Se calcula: Arce (2021)



$$F1 - score = \frac{2((Sensibilidad)(Precisión))}{(Sensibilidad + Precisión)} \tag{2.5}$$

**Curva ROC (característica de funcionamiento del receptor)**: Es una métrica que sirve para verificar el rendimiento de cualquier modelo de clasificación. Con la curva ROC se consigue saber que tan bueno es un modelo al momento de identificar entre varias clases AprendeIA (2021), se obtiene graficando:

$$sensibilidad\ vs\ (1 - Especificidad) \tag{2.6}$$

**Área bajo la curva (ABC)**: Es el área bajo la curva ROC. Este puntaje nos da una buena idea de qué tan bien funciona el modelo y se calcula: AprendeIA (2021)

$$ABC = \frac{(1 - Especificidad)(Sensibilidad)}{2} + \frac{(1 + Sensibilidad(1 - (1 - Especificidad)))}{2} \tag{2.7}$$

# Capítulo 3

# Proceso de elección y entrenamiento de la red

En este capítulo se describe el tipo de red neuronal artifical (RNA) que se utilizó para el caso particular de la detección de lesiones cariosas en los dientes, así como la descripción para el entrenamiento de la RNA y las herramientas utilizadas para su funcionamiento.



## 3.1 Elección de la red neuronal

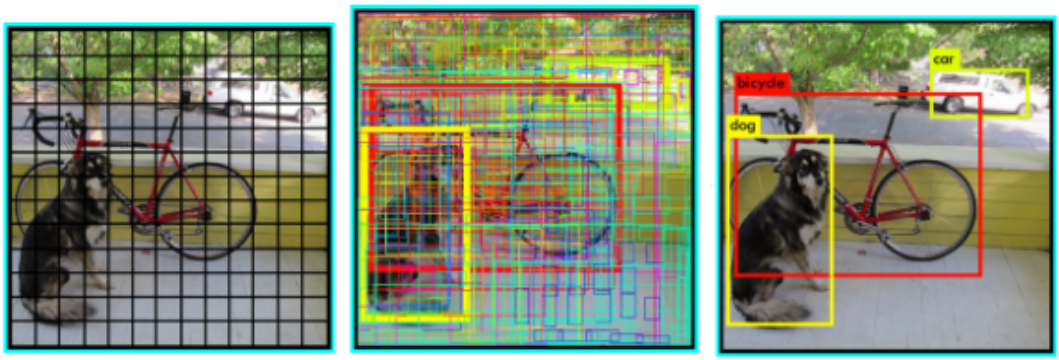
Para este proyecto se intentó crear una RNA desde cero, sin embargo, este acercamiento se descartó debido a que para obtener los resultados esperados en la precisión de identificación de la RNA, era necesario un nivel de hardware y software de alta capacidad de cálculo. El computador usado, una laptop VIT modelo M2420, no presentó la capacidad de para soportar los cálculos requeridos para el entrenamiento, por lo que se decidió el uso de transferencia de aprendizaje, entrenando una capa adicional sobre una RNA preexistente. Se entrenó una red personalizada debido a que para el momento de la realización de este proyecto no existía alguna *CNN* entrenada para la detección de lesiones cariosas en las estructuras dentales, en tal sentido se realizó un entrenamiento cuyo tiempo de duración fue de 48 horas.

Se decidió el uso de una RNA convolucional debido a que es una red multicapa

formada por capas alternadas de convolución y reducción, en las que cada capa es entrenada para realizar una tarea, esto reduce significativamente el número de capas ocultas, por lo que el tiempo de entrenamiento de la red es más rápido, además, presenta invarianza a la traslación de los patrones a identificar. Este tipo de RNA tienen la capacidad de detectar características simples como por ejemplo bordes, líneas, entre otros y componer en características más complejas hasta detectar las clases deseadas.

Para este proyecto se usó *YOLO* en su versión 4 el cual es un modelo de reconocimiento de objetos que cuenta con una implementación de la *CNN CSPDarknet53*. Para llevar a cabo la detección *YOLO* primero divide la imagen en una cuadrícula de SxS Figura 3.1a. En cada una de las celdas predice N posibles cajas circundantes y calcula el nivel de probabilidad de cada caja 3.1b, es decir, se calculan SxSxN diferentes cajas, la gran mayoría de ellas con un nivel de probabilidad muy bajo. Después de obtener estas predicciones se eliminan las cajas que estén por debajo de un límite de probabilidad que por defecto es del 30%. A las cajas restantes se les aplica un paso de *"non-max suppression"* que sirve para eliminar posibles objetos que fueron detectados por duplicado y así dejar únicamente el que tenga el porcentaje mayor de exactitud entre ellos Figura3.1. Las S divisiones de la cuadrícula y las N posibles cajas son parámetros predefinidos que van a depender del tamaño de la imagen de entrada, una imagen más grande va a tener mayor número de S divisiones y N predicciones.






(a) (b) (c)

Figura 3.1: Puesta en marcha de *YOLO* (a) Segmentación imagen (b) Búsqueda de objetos (c) Reconocimiento de objetos

## 3.2 Recursos empleados para el entrenamiento de la RNA

Para el entrenamiento de la RNA se necesitó un conjunto de herramientas computacionales para facilitar el entrenamiento. En este caso se usó

- *Google Colaboratory*: también llamado *"Colab"*, el cual da los recursos mínimos para ejecutar las herramientas necesarias y realizar el entrenamiento de la *CNN* esta herramienta genera un entorno de programación en *Python* desde el navegador web con las siguientes ventajas: No requiere de configuración previa, da acceso gratuito por tiempo limitado a unidades de procesamiento gráfico (*GPUs*) y permite compartir contenido de manera fácil Google (2020). La herramienta se encuentra en la direccion web https://colab.research.google.com/notebooks/intro.ipynb y uso requiere de conección a internet, para el proyecto se usó la versión paga de la herramienta llamda *"Colab PRO"* la cual permite tiempo de acceso ilimitado a *GPUs*



- labelImg: es una herramienta gráfica de anotación de imágenes. Está hecha con el lenguaje de programación *Python* y usa el *framework* de desarrollo de aplicaciones multiplataforma *Qt* para su interfaz gráfica, y los archivos que genera cuando se usa para *YOLO* son de extensión .txt donde se detalla la información sobre la clasificación de los objetos en la imagen, incluyendo las clases para su reconocimiento. (Darrenl (2020)).Es de código abierto y se puede descargar del repositorio de *github*: https://github.com/tzutalin/labelImg. Para este proyecto se uso la version 1.8.4 en el sistema operativo Xubuntu Figura: 3.2

- *Google Drive*: es un servicio de alojamiento de archivos en línea creado por Google que da hasta 15 Gb de espacio de almacenamiento gratuito, para su uso se debe tener una cuenta de Google y acceder por medio de la direccion web https://drive.google.com/drive/my-drive o descargar la versión para dispositivos móviles desde la tienda de aplicaciones de Google.

- *Yolo Version 4:* es un detector de objetos que puede realizar detecciones en

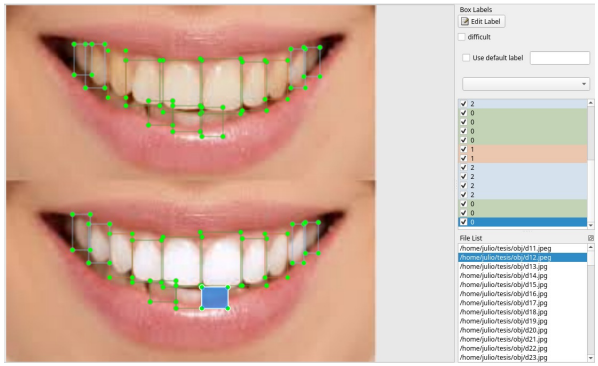



Figura 3.2: Etiquetado de objetos con labelImg

video y en imágenes, la implementación usada para el proyecto se basó en *Darknet*, que es un *framework* de RNA de código abierto escrito en los lenguajes de programación C y CUDA. *Darknet* establece la arquitectura subyacente de la red y se utiliza como *framework* para entrenar el modelo basado en *YOLO*. Esta herramienta es de uso gratuito y se puede usar tanto en *"Colab"* como en un computador sin conexión a internet, esta disponible en la direccion web https://github.com/AlexeyAB/darknet



- *Tensorflow*: Esta herramienta es una biblioteca de codigo abierto escrita en el lenguaje de programación *python* y diseñada por Goolge que se puede usar en *"Colab"* o en el computador sin conexión a internet su uso es gratuito y es accesible escribiendo en un navegador dirección web https://www.tensorflow.org/. Para este proyecto se usó junto con un *script* de *python* llamado *convert.py* que se encuentra disponible en el repositorio de *github*: https://github.com/theAIGuysCode/tensorflow-yolov4-tflite y sirve para cambiar el formato de los pesos de la red de *YOLO* a *Tensorflow* lo que facilitó implementación de los pesos de la red personalizada en la aplicación móvil

## 3.3 Conjunto de datos

Se creó el conjunto de datos personalizado que se generó con imagenes obtenidas de la busqueda manual en *Google images* e imágenes proporcionadas por diferentes cátedras

de la Facultad de Odontología de la Universidad de Los Andes obteniendo así un conjunto de 933 imágenes, al finalizar el proceso de etiquetado con la herramienta labelImg, que tomó un tiempo de 56 horas, se obtuvo que el conjunto de datos contenía 7990 objetos distribuidos como se muestra en la Tabla 3.1. Se crearon dos carpetas de

Tabla 3.1: Distribución de objetos clasificados por clases

| Clases identificadas | Número de objetos |
|---|---|
| ICDAS0 | 5287 |
| ICDAS1 | 1046 |
| ICDAS2 | 455 |
| ICDAS3 | 200 |
| ICDAS4 | 112 |
| ICDAS5 | 146 |
| ICDAS6 | 744 |



imágenes con los nombres de *obj* y *test* con la finalidad de tener una parte del conjunto de datos para el entrenamiento de la RNA y otra para la validación de la RNA, en la carpeta *obj* se almacenaron las imágenes destinadas al entrenamiento de la *CNN* y en la carpeta *test* las imágenes para la validación de la red. Estas imágenes de validación se usaron para las pruebas de la *CNN*.

## 3.4 Proceso de entrenamiento de la red

Se generó una versión comprimida de ambas carpetas con los mismos nombres y de extensión .zip con la finalidad de poder exportar ambos conjuntos de datos a *Google Drive*.

Estos archivos fueron almacenados en una carpeta con el nombre *yoloV4* permitiendo el acceso de *Colab* al conjunto de datos. En *Colab* en las configuraciones del área de trabajo se habilitó el uso de *GPU* y se cargó *Darknet* desde el repositorio de *github*. Una vez cargado *YOLO* en *Colab* se accedió a la carpeta del repositorio llamada *darknet* y se modificó el archivo con el nombre *Makefile* para habilitar el uso de *GPU*

con *Darknet*, el cual fue usado para compilar *Darknet* haciendo uso del comando *make*. Se definió una función en *python* con el nombre *imShow* que mostraba por pantalla los resultados de la detección sobre la imagen que se le da a la red para identificar. Se adaptó el archivo *yolov4-custom.cfg* ubicado dentro de la carpeta *cfg* de *Darknet* para cumplir con los parámetros de el entrenamiento de una *CNN* personalizada sugeridos en la documentación de Guy (2020), que se basa en el documento oficial de *YOLOV4* Bochkovskiy et al. (2020) donde se define los valores de width = 416, height = 416, usadas para definir la resolución de la imagen al momento de ser analizada durante el entrenamiento. En este caso 416 píxeles de largo por 416 píxeles de ancho definen el tamaño de entrada estándar, según el documento oficial el uso tamaños de entrada mayores generan mayor precisión, pero incrementan exponencialmente el tiempo de duración del entrenamiento. $max_batches$ que se define como el número de clases del conjunto de datos con el que se entrena la red, multiplicado por 2000 Ecuación 3.1 y se usa para definir el número de iteraciones que el entrenamiento debe tener Ecuación 3.1, y *filters* que es igual al número de clases del conjunto de datos con el que se entrena la red más 5 y multiplicado por 3. Ecuación 3.2



$$max\_batches = (Num\ de\ clases)2000 \tag{3.1}$$

$$filters = (Num\ de\ clases + 5)3 \tag{3.2}$$

Se creó un archivo local con el nombre *obj.names* en el que se guardó el nombre de cada una de las clases por línea y se cargó en la carpeta*yolo V4*. Se descargaron en el computador los archivos *train.txt* y *test.txt* del repositorio https://github.com/theAIGuysCode/YOLOv4-Cloud-Tutorial que sirven para contener las rutas relativas al conjunto de datos de entrenamiento y validacion, una vez editados con la ruta correcta se cargaron en la carpeta *yolo V4*.

Para realizar la transferencia de aprendizaje y reducir el tiempo de entrenamiento de la red personalizada se descargó en *Colab*, dentro de la carpeta *darknet* los pesos pre entrenados de *YOLOV4* pertenecientes al conjunto de datos COCO que cuenta con más de 80 clases entrenadas para la identificación de diferentes tipos de objetos y se encuentra disponible en la direccion web https://cocodataset.org/ se copiaron los archivos guardados en *Google Drive* para *Colab* y así poder usarlos en el proceso

de entrenamiento. Se descomprimieron los archivos *obj.zip* y *test.zip* y se inició el entrenamiento con *Darknet* de la red personalizada sobre la red *YOLO V4* que usó los pesos pre entrenados en sus capas de convolución, lo que generó un entrenamiento más rápido y preciso.

# Capítulo 4

# Diseño de la aplicación móvil

Actualmente el desarrollo de aplicaciones móviles es una de las áreas de mayor crecimiento a nivel económico y educacional, siendo así un área de la tecnología accesible para cualquier persona sin importar su nivel de conocimientos técnicos. En este capítulo se encuentra lo concerniente al diseño e implementación de la aplicación móvil para la detección de caries dental. La aplicación móvil se desarrolló en Android, haciendo el uso del IDE Android Studio que es el entorno de programación anunciado por Google Inc.



## 4.1 Entorno de desarrollo

En el diseño de la aplicación móvil se usó el entorno de desarrollo *Android Studio IDE* en su versión 4.1.2, su elección fue debida por ser un entorno bastante completo especializado en el desarrollo de aplicaciones móviles, cuenta con gran variedad de funcionalidades que facilitan la programación de las aplicaciones y cuenta con su propio editor.

Las aplicaciones que se desarrollan en *Android Studio* se denominan proyectos, cada proyecto puede contener uno o varios elementos, los cuales pueden ser archivos de código fuente, un formulario que use la aplicación, entre otras cosas. Los módulos que componen un proyecto tienen 3 carpetas esenciales que son: la carpeta de manifiestos (*Manifest*) que dentro de la aplicación lleva por nombre *AndroidManifest.xml*,

y es el lugar donde se declaran los permisos necesarios para ejecutar la aplicación, permisos que pueden incluir, pero no están limitados a: uso de hardware (cámara, microfono,*GPS*), acceso especial a otras aplicaciones (contactos, galería multimedia). La carpeta *Java* en la que se encuentran todos los archivos relacionados con cada *activity* o actividad, como código fuente y clases auxiliares. En esta carpeta se encuentra el archivo *MainActivity.java* que contiene la lógica esencial para que el usuario interactúe con la aplicación. La carpeta de recursos *res* que contiene todos los recursos adicionales para la completación de una aplicación móvil, esto es, la parte gráfica y las cadenas de texto que son generalmente usadas en cada actividad, en cada título, menú, lista desplegable, etc.

Otra de las características importantes de un proyecto son los archivos de extensión *XML* es un lenguaje de marcado que, a diferencia del *HTML*, otro lenguaje de etiquetado que da forma y estilo a los datos que contiene, este define la estructura y significado de los datos que contiene gracias a la forma en que se etiquetan, y es ampliamente utilizado por su portabilidad. Estos archivos se encuentran dentro de la carpeta *res* distribuidos como se muestra en la Tabla 4.1.

Tabla 4.1: Especificaciones de la carpeta *res*

| Carpeta | Descripción |
|---|---|
| /res/drawable/ | Contiene las imágenes y otros elementos gráficos usados por la aplicación. Para poder definir diferentes recursos dependiendo de la resolución y densidad de la pantalla del dispositivo. |
| /res/mipmap/ | Contiene los iconos de lanzamiento de la aplicación (el icono que aparecerá en el menú de aplicaciones del dispositivo) para las distintas densidades de pantalla existentes. Al igual que en el caso de las carpetas /drawable, se dividirá en varias subcarpetas dependiendo de la densidad de pantalla. |
| /res/layout/ | Contiene los ficheros de definición XML de las diferentes pantallas de la interfaz gráfica. Para definir distintos *layouts* dependiendo de la orientación del dispositivo. |
| /res/color/ | Contiene ficheros XML de definición de listas de colores según su estado. |
| /res/menu/ | Contiene la definición XML de los menús de la aplicación. |
| /res/xml/ | Contiene otros ficheros XML de datos utilizados por la aplicación. |
| /res/raw/ | Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos. |
| /res/raw/ | Contiene otros ficheros XML de recursos de la aplicación, como por ejemplo cadenas de texto (strings.xml), estilos (styles.xml), colores (colors.xml), arrays de valores (arrays.xml), tamaños (dimens.xml), etc. |



La programación de las funcionalidades se realiza en las actividades, definida como toda pantalla que se puede observar y con la que el usuario puede interactuar. Una actividad por lo general abarca toda una ventana en el dispositivo móvil. Es por esto que, una aplicación elaborada se puede definir como un conjunto de actividades enlazadas entre sí. En cuanto a la forma de operación de una aplicación vista desde sus actividades, la documentación dice que cada vez que se hace una acción que resulte en un cambio de pantalla (cambio de actividad) la actividad anterior se detiene, pero no sin antes almacenar esa actividad previa en la pila de actividades, por lo que, puede

recuperarse cuando el usuario use el botón tan conocido de ir hacia “atrás” González (2019).

## 4.2 Diseño de la interfaz de la aplicación móvil

La interfaz de usuario se hizo con 4 elementos. Este diseño se aprecia en la Figura 4.1 y se compone de las siguientes características:

- Los botones: El botón analizar que ejecuta la detección sobre la imagen y el botón cargar el cual abre el menú para seleccionar si tomar una foto con la cámara o elegir una imagen de la galería .
- La cabecera: En donde se encuentra la información general de **CariesApp** como el nombre y la versión.
- El visor de la imagen: en donde carga la imagen y los resultados junto con los rectángulos de clasificación.
- La barra del nivel de las lesiones: Una barra informativa para que el usuario tenga una guía de color con la correspondencia entre la detección y la clase equivalente.



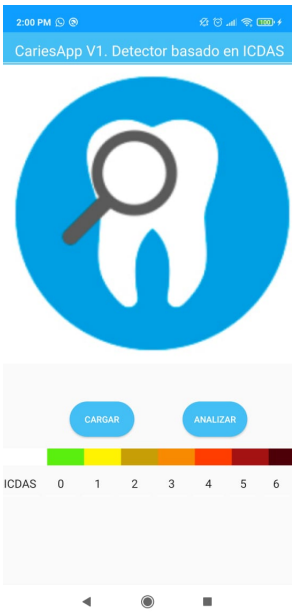



Figura 4.1: Interfaz de CariesApp

## 4.3 Arquitectura de la aplicación móvil

Para cumplir el objetivo de la aplicación **CariesApp** se usó como base la aplicación de detección de objetos creada por Shakeel (2020) que se encuentra disponible en el repositorio de *github*: https://github.com/haroonshakeel/tensorflow-yolov4-tflite que cuenta con tres actividades, la actividad principal definida en el archivo *MainActivity.java* la actividad del detector definida en el archivo *DetectorActivity.java*, y la actividad de la cámara definida en el archivo *CameraActivity.java*.

A su vez está aplicación esta basada en un clasificador de Google (**TF classify**, basado en la versión para móviles de *TensorFlow*, *TensorFlow lite* ) la cual se encuentra disponible en la direccion web: https://www.tensorflow.org/tutorials/images/classification

Se modificó la actividad *MainActivity.java*, agregando la función *selectImage* la cual sirve para acceder a la cámara o a la galería del dispositivo móvil dependiendo de la opción que el usuario elija y muestra la imagen en la ventana de la actividad principal haciendo uso del objeto *imageView* el cual está predefinido en *Android Studio* para el manejo de imágenes.



Se modificó la actividad *MainActivity.java* para que la función *detector* que es la que se encarga del título de la etiqueta y la probabilidad acierte en la detección de la imagen. Estos datos son almacenados en una cola de prioridad, además, la probabilidad se obtiene normalizada entre 0 y 1. La suma de las probabilidades debe sumar 1 cuando está normalizado gracias a un método llamado *softmax* que infiere en la convergencia de la identificación. Para que funcionara con un modelo de *YOLO V4* en formato *TensorFlow lite*, se hizo uso del servicio *YoloV4Classifier.java* que se encuentra ubicado en la carpeta *tflite* del proyecto, este servicio es el encargado de procesar la imagen transformándola en un vector de una sola dimensión con 8 bytes para cada canal de color, tomando como base los colores primarios de la luz que son rojo,verde y azul

Esto se hace de forma secuencial y cada pixel debe tener un valor entre [0,1], para esto, el valor de cada pixel se normaliza dividiendo su valor entre 255. Para el manejo de los resultados se modificó la función *handleResult* la cual recibe como parámetros el mapa de bits de la imagen ( *bitmap*) y una lista de las clases detectadas en la imagen

junto con su posición obtenidas con la función *detector*. Esta modificación se basó en darle un color especificó a cada uno de los rectángulos de las cajas circundantes que se generan sobre la imagen dependiendo del grado de la lesión que se identificó como se detalla en la Tabla 4.2.

Tabla 4.2: Colores de las cajas circundantes

| Clase identificada | Color |
|---|---|
| ICDAS0 | Verde |
| ICDAS1 | Amarillo |
| ICDAS2 | Amarillo oscuro |
| ICDAS3 | Naranja |
| ICDAS4 | Naranja oscuro |
| ICDAS5 | Rojo |
| ICDAS6 | Vinotinto |



Para que el modelo fuese compatible con la aplicación móvil se cambió el formato de *YOLO* a *TensorFlow Lite* con la ayuda del *script convert.py* ubicado en el repositorio de *github*: https://github.com/theAIGuysCode/tensorow-yolov4-tite el cual al finalizar su ejecución genera un archivo de nombre *custom-416.tflite* que contiene el modelo personalizado en formato *TensorFlow Lite*.

Fue agregado a la carpeta *assets* del proyecto el archivo *custom-416.tflite* junto con el archivo *coco.txt* que contiene las etiquetas de las clases para el identificador, para habilitar su uso dentro de la aplicación y hacer posible ejecutar la detección sobre estructuras dentales con lesiones cariosas de diferentes grados. Se modificó el archivo *DetectorActivity.java* cambiando el modelo por defecto por el modelo del archivo *custom-416.tflite* y las clases por defecto se cambiaron por las clases del *archivo coco.txt* Figura 4.2

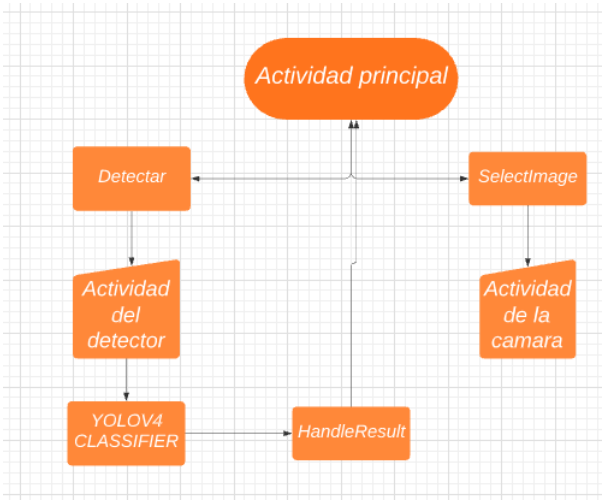



Figura 4.2: Diagrama del funcionamiento interno de la arquitectura de **CariesApp**

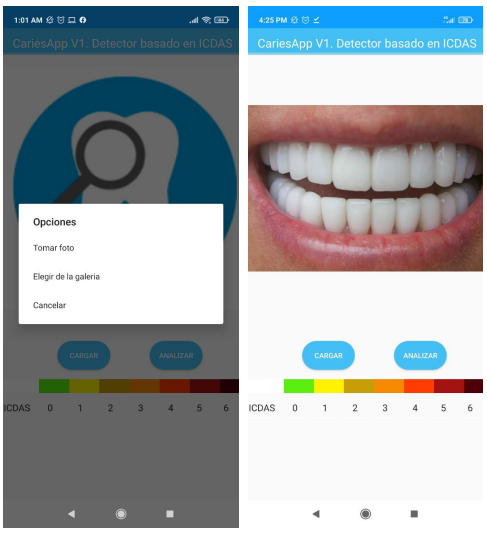
## 4.4 Funcionamiento de la aplicación móvil

La aplicación desarrollada lleva por nombre **CariesApp**, su núcleo se centra en el reconocimiento de los diferentes grados de las lesiones cariosas en las diferentes estructuras dentales basado en el *ICDAS*, haciendo uso de la cámara de un dispositivo móvil con sistema operativo Android y la implementación de una *CNN* personalizada especializada en detección de objetos, esto debido a que en una imagen se pueden apreciar más de una estructura dental y más de una lesión por lo que se clasifican como objetos reconocibles dentro de la imagen.



En líneas generales **CariesApp** da al usuario la opción de tomar una foto o elegir una imagen de la galería mostrarla en pantalla y luego efectuar la detección sobre dicha imagen mostrando los resultados obtenidos sobre la misma imagen dibujando las cajas circundantes sobre las lesiones identificadas. Este funcionamiento se divide en 3 procesos los cuales se definen como:

- El inicio de la aplicación:en donde el usuario tiene acceso a la actividad principal, la cual genera la vista principal de **CariesApp** Figura 4.1.
- La carga de la imagen: que se lleva a cabo mediante la interacción con el botón cargar el cual abre un menú con las opciones de tomar foto o cargar imagen de la galería Figura 4.3.




(a) (b)

Figura 4.3: Proceso de selección y carga de la imagen (a) opciones de selección (b) imagen lista para ser analizada

- La detección de los objetos en la imagen: Este proceso se ejecuta cuando el usuario oprime el botón analizar el cual hace uso de la función *detector* y está a su vez usa el servicio *YOLOV4Classifier* para obtener los resultados de la detección los cuales son mostrados en la pantalla de la actividad principal gracias a la función *handleResults*.

  Una vez presionado el botón de analizar se generaran los rectángulos correspondientes a la clasificación efectuada por la *CNN* y por la probabilidad de *Softmax*, resultando en los colores equivalentes a cada clase, como se muestra en la Figura 4.4

.

## 4.5 Requerimientos y pruebas de la aplicación móvil

Se probó **CariesApp** en los equipos móviles xiaomi redmi 9s, xiaomi redmi note 8, LG K8, Motorola Moto G6 y se determinó que los requemientos mínimos para el

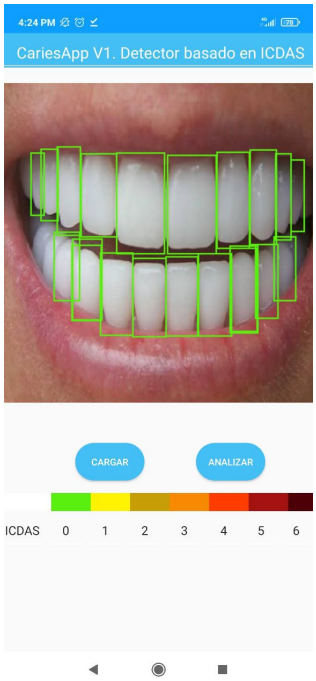



Figura 4.4: Resultados de la detección



funcionamiento de la aplicación fueron un sistema operativo Android 8 o superior, un procesador mayor o igual a 1.8GHz, memoria RAM de al menos 3Gb y espacio libre de almacenamiento disponible de 300Mb.

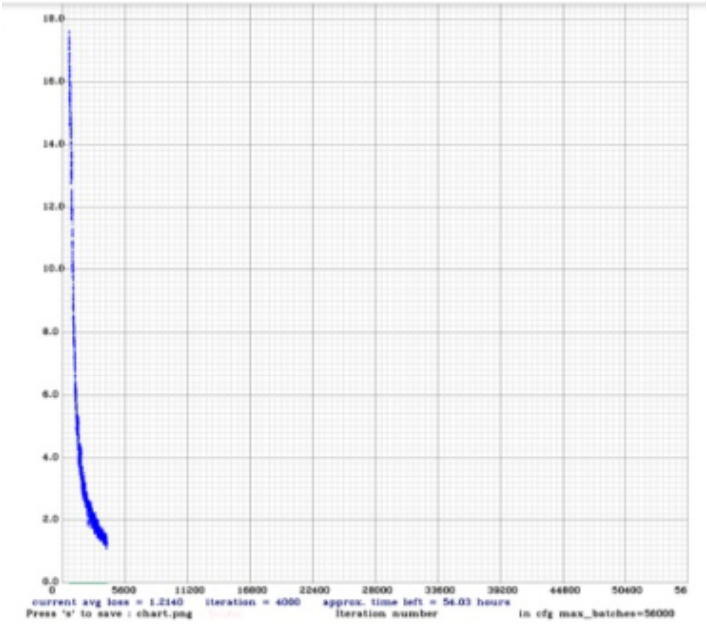
# Capítulo 5

# Resultados

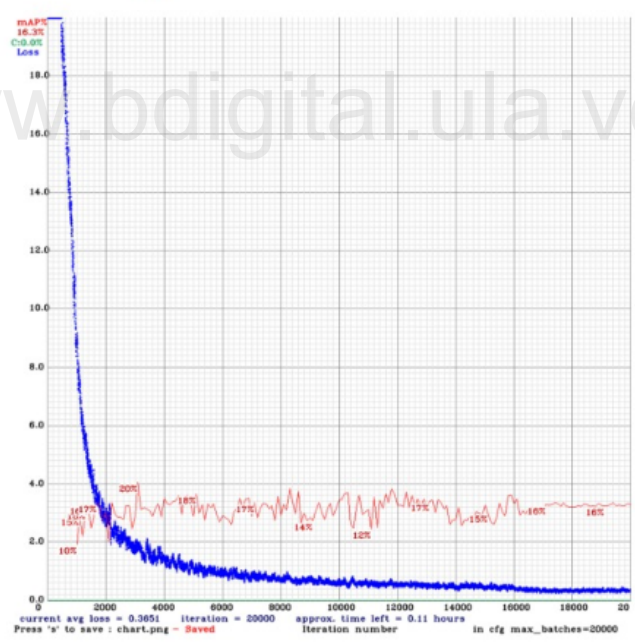
Con el objetivo de verificar la eficacia del entrenamiento de la *CNN* se crearon dos modelos, uno con una proporción de 70% imágenes de entrenamiento y 30% imágenes de validación llamado modelo 1 y otro con una proporción de 80% imágenes de entrenamiento y 20% imágenes de validación llamado modelo 2, para ambos modelos se usó el conjunto de datos de 933 imágenes.



Se obtuvieron las gráficas de la pérdida promedio durante el entrenamiento, con el uso de la función *imgShow*, para ambos modelos se obtuvo que la pérdida promedio fue menor al 2% lo que indica que ambos modelos se entrenaron correctamente según lo indicado en la documentación de Guy (2020) . Debido a que el tiempo de duración del entrenamiento de ambos modelos fue muy extenso se realizó por partes y se graficó la etapa final del mismo para ambos modelos, el número de iteraciones para ambos modelos fue de 20000. Figura 5.1




(a)




(b)

Figura 5.1: Gráficas de pérdida (a) Modelo 1 (b) Modelo 2

El desempeño en la identificación para ambos modelos se midió utilizando una matriz de confusión multiclase para el modelo 1 (Tabla 5.1) y modelo 2 (Tabla 5.2), a

partir de ambas matrices se obtuvo la tasa de verdaderos positivos (VP), verdaderos negativos (VN), falsos positivos (FP) y falsos negativos (FN) para ambos modelos (Tabla 5.3). Para la construcción de estas matrices se ejecutó la aplicación móvil desarrollada con cada una de las imágenes del conjunto de validación para cada modelo.

Tabla 5.1: Matriz de confusión del modelo 1

| | | Datos verdaderos | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Clases | C0 | C1 | C2 | C3 | C4 | C5 | C6 | Total predichos |
| Predicción | C0 | 164 | 33 | 32 | 46 | 32 | 10 | 4 | 321 |
| | C1 | 1 | 35 | 3 | 3 | 1 | 1 | 0 | 44 |
| | C2 | 0 | 0 | 3 | 1 | 0 | 0 | 0 | 4 |
| | C3 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 2 |
| | C4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | C5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| | C6 | 3 | 1 | 1 | 15 | 15 | 5 | 36 | 76 |
| | Total verdaderos | 168 | 70 | 39 | 66 | 48 | 17 | 40 | 448 |

Tabla 5.2: Matriz de confusión del modelo 2

| | | Datos verdaderos | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Clases | C0 | C1 | C2 | C3 | C4 | C5 | C6 | Total predichos |
| Predicción | C0 | 270 | 39 | 29 | 13 | 9 | 8 | 0 | 368 |
| | C1 | 14 | 26 | 2 | 6 | 3 | 0 | 0 | 51 |
| | C2 | 6 | 0 | 9 | 2 | 0 | 0 | 0 | 17 |
| | C3 | 1 | 0 | 0 | 13 | 4 | 2 | 0 | 20 |
| | C4 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| | C5 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 3 |
| | C6 | 2 | 1 | 1 | 14 | 6 | 4 | 28 | 56 |
| | Total verdaderos | 293 | 66 | 41 | 49 | 22 | 17 | 29 | 517 |



Tabla 5.3: Resultados modelo 1 y modelo 2

| | | modelo 1 | | | | modelo 2 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | VP | FP | VN | FN | VP | FP | VN | FN |
| Clases | C0 | 164 | 157 | 123 | 4 | 270 | 98 | 125 | 23 |
| | C1 | 35 | 9 | 369 | 35 | 26 | 25 | 424 | 40 |
| | C2 | 3 | 1 | 408 | 36 | 9 | 8 | 467 | 32 |
| | C3 | 1 | 1 | 381 | 65 | 13 | 7 | 461 | 36 |
| | C4 | 0 | 0 | 400 | 48 | 0 | 2 | 493 | 22 |
| | C5 | 1 | 0 | 431 | 16 | 2 | 1 | 499 | 15 |
| | C6 | 236 | 40 | 368 | 4 | 28 | 28 | 460 | 1 |

En la Tabla 5.3 se aprecia como el modelo 2 tuvo un mayor número de VP, con un total de 348 aciertos sobre el modelo 1 que tiene un total de 240 aciertos, esto indica que el modelo 2 tuvo mayores posibilidades de acertar la identificación para cualquier clase a identificar sobre una imagen aleatoria en comparación con el modelo 1. Para la tasa de FP se apreció que el modelo 2 obtuvo un total de 169 FP mientras que

el modelo 1 obtuvo un total de 208 FP esto señala que el modelo 1 tiene mayores probabilidades de identificar un código ICDAS de manera errónea que el modelo 2. En relación a la tasa de VN el modelo 2 mostró un mejor desempeño con un total de 2929 VN, mientras que el modelo 1 obtuvo un total de 2480 VN, con base a este resultado y con la tasa de VP se puede decir que el modelo 2 tiene una mejor exactitud. Para la tasa de FN el modelo 1 obtuvo un total 208 aciertos, mientras que el modelo 2 un total de 169 FN lo que indica que el modelo 2 tiene menores probabilidades de errar al momento de identificar una clase.

Con estos resultados se pudo obtener los valores de la precisión, exactitud, sensibilidad, especificidad, el área bajo la curva (ABC) y el valor F1, con los cuales se consiguió medir el desempeño de ambos modelos para la identificación de cada una de las clases Tablas 5.4 y 5.5.

En relación a los resultados obtenidos para cada una de las clases se aprecia que la clase C0 del modelo 1 su sensibilidad fue de 0,9800 mientras que para el modelo 2 fue de 0,9200, lo que indica que el modelo 1 consiguió efectuar una mejor identificación de VP que el modelo 2. Para esta misma clase la especificidad fue de 0,4393 para el modelo 1 y de 0,5605 para el modelo 2, esto indica que el modelo 2 tiene una mejor detección de la clase C0. Esto es debido a que la cantidad de imágenes de la clase C0 fue superior en el modelo 2 que en el modelo 1.



Para la clase C1 del modelo 1 la especificidad fue de 0,9762 y la del modelo 2 fue de 0,9443, ambos valores se consideran buenos, sin embargo el modelo 1 realiza una mejor identificación de la tasa de VN que el modelo 2 para esta clase. En este sentido en esta clase el modelo 2 obtuvo una sensibilidad de 0,3900 lo que se considera como un valor bajo y malo y el modelo 1 una sensibilidad de 0,5000 que aunque es mejor que la del modelo 2, se sigue considerando como baja por lo que ambos modelos no tienen un buen desempeño en la identificación de la proporción de los VP para esta esta clase.

Para las clase C2, C3, C4, C5 y C6 se observa el mismo comportamiento en relación a la especificidad siendo el modelo 1 superior al modelo 2 ambos con valores de especificidad para estas clases mayores de 0,90 entendiéndose como un buen resultado en especificidad, pero en líneas generales el modelo 1 es mejor para la identificación de la tasa de VN que el modelo 2.

Al analizar las otras métricas se aprecia que el modelo 1 para la clase C2 tiene un valor F1 de de 0,1400 y el modelo 2 un valor F1 de 0,3900 si bien el valor F1 para el modelo 2 clase C2 es mayor que el de la clase 1 ambos se consideran malos resultados lo que indica que para esta clase ambos modelos tienen un mal desempeño en cuanto a la precisión y la sensibilidad.

Para la clase C3 el modelo 1 obtuvo un valor de sensibilidad de 0,0150 y el modelo 2 un valor de sensibilidad de 0,2700 ambos considerados malos por lo que si bien el modelo se desempeña un poco mejor que el modelo 1 sigue teniendo un reconocimiento malo de la tasa de verdaderos positivos para esta clase.

En ambos modelos se observa que la clase C4 tiene el valores más bajo en sensibilidad, precisión y valor F1 el cual es 0 lo que indica que ambos modelos no son capaces de identificar correctamente esta clase en cualquier imagen aleatoria, esto se debió a que durante el proceso de clasificación y etiquetado de las imágenes se obtuvieron solo 42 objetos de la clase C4.

Para la clase C5 el valor F1 del modelo 1 fue de 0,1100 y el de el modelo 2 fue de 0,2000, esto indica que aunque el modelo 2 tuvo un mejor desempeño que el modelo 1 en cuanto a la precisión y la sensibilidad. se sigue considerando un valor malo por lo que se determinó que el reconocimiento para la clase C5 en general es malo, esto debido a la falta de imágenes con objetos de la clase C5 durante el proceso de etiquetado y entrenamiento.

Para la clase C6 se pudo apreciar que el modelo 1 tuvo un valor de precisión de 0,4700, mientras que el modelo 2 un valor de precisión 0,5000, siendo el modelo 2 superior al modelo 1 pero insuficiente para considerarse bueno en la detección del porcentaje de VP para esta clase

Tabla 5.4: Métrica modelo 1

| Clases | Precisión | Exactitud | Sensibilidad | Especificidad | Valor F1 | ABC |
|---|---|---|---|---|---|---|
| C0 | 0,5100 | 0,5100 | **0,9800** | 0,4393 | 0,6700 | 0,7077 |
| C1 | 0,8000 | 0,7954 | 0,5000 | **0,9762** | 0,6100 | 0,7381 |
| C2 | 0,7500 | 0,7500 | 0,0770 | **0,9976** | 0,1400 | 0,5372 |
| C3 | 0,5000 | 0,5000 | 0,0150 | **0,9974** | 0,0290 | 0,5063 |
| C4 | 0,0000 | 0,0000 | 0,0000 | **1,0000** | 0,0000 | 0,5000 |
| C5 | **1,0000** | 0,1000 | 0,5600 | **1,0000** | 0,1100 | 0,5294 |
| C6 | 0,4700 | 0,4736 | 0,9000 | **0,9020** | 0,6200 | 0,9010 |

Tabla 5.5: Métrica modelo 2

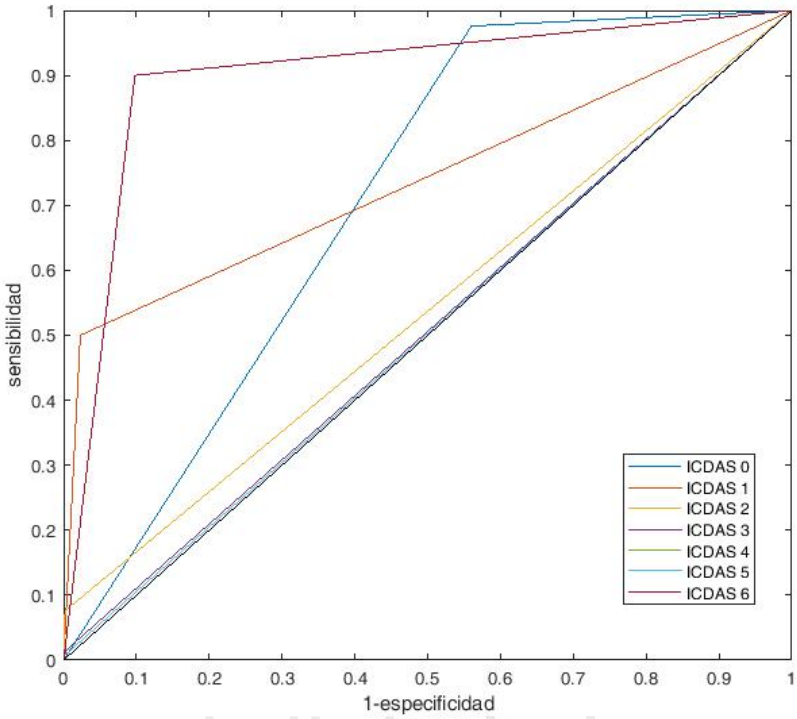
| Clases | Precisión | Exactitud | Sensibilidad | Especificidad | Valor F1 | ABC |
|---|---|---|---|---|---|---|
| C0 | 0,7300 | 0,7660 | **0,9200** | 0,5605 | 0,8200 | 0,7410 |
| C1 | 0,5100 | 0,8743 | 0,3900 | **0,9443** | 0,4400 | 0,6690 |
| C2 | 0,5300 | 0,9226 | 0,2200 | **0,9832** | 0,3100 | 0,6010 |
| C3 | 0,6500 | 0,9168 | 0,2700 | **0,9850** | 0,3800 | 0,6250 |
| C4 | 0,0000 | 0,9536 | 0,0000 | **0,9960** | 0,0000 | 0,4980 |
| C5 | 0,6700 | 0,9691 | 0,1200 | **0,9980** | 0,2000 | 0,5580 |
| C6 | 0,5000 | 0,9483 | **0,9700** | 0,9426 | 0,6600 | 0,9540 |



Los valores obtenidos de ABC indican que la clase C4 con un valor de 0,498, y la clase C5 con un valor de 0,558 lo que representa una probabilidad cercana del 50% de reconocer un caso aleatorio como positivo frente a uno negativo. En tal sentido, este comportamiento para estas clases demuestran que no hay certeza en la detección para este par de clases, lo que se considera como un desempeño desfavorable o malo en para estas dos clases. La clase C6 presenta un ABC de 0,954 siendo el código más reconocible por la aplicación móvil, esto es indicativo de un buen desempeño, lo que se traduce como una clase mas sensible, con menos casos de FN

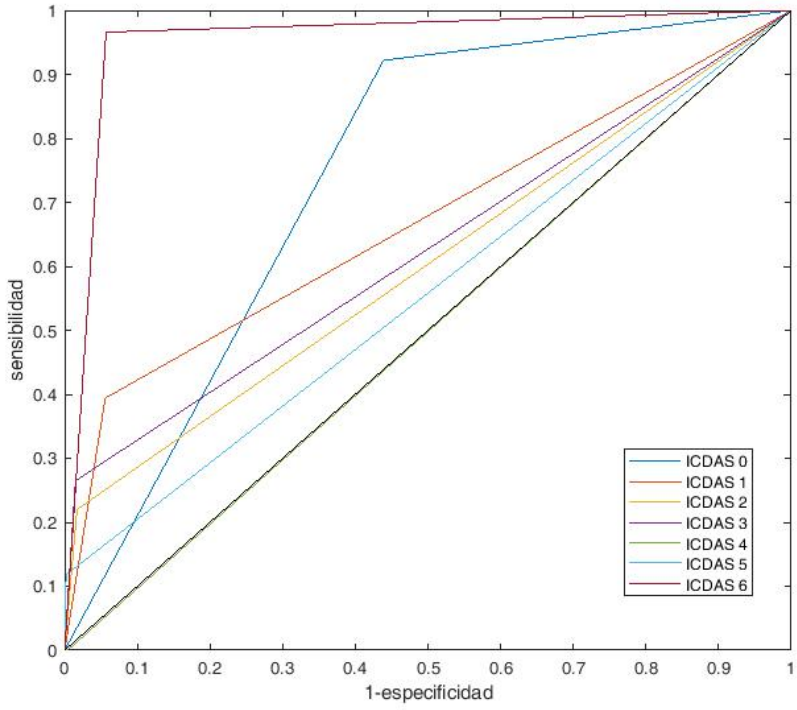
El modelo 2 es superior al modelo 1, porque tuvo más imágenes en el entrenamiento y las métricas de sensibilidad, precisión, ABC y Valor F1 fueron superiores al modelo

1. Estos valores obtenidos son coherentes con las métricas de rendimiento anteriores. Demostrando así que esta aplicación móvil tiene un buen desempeñoo para el reconocimiento de los las clases C0 y C6, un desempeño regular para las clases C1,C2 y C3 y un mal desempeño para las clases C4 y C5.






(a)




(b)

Figura 5.2: Gráficas de curvas ROC (a) modelo 1 (b) modelo 2

# Capítulo 6

# Conclusiones y recomendaciones

## 6.1 Conclusiones

Se desarrolló una aplicación móvil con éxito, la cual se implementó usando tecnología novedosa, en los lenguajes de programación más populares de la actualidad. El entrenamiento de la *CNN* que usa la aplicación para la detección de las clases se logró gracias al correcto uso e implementación del método de transferencia de aprendizaje usando tecnología recientemente disponible y de acceso gratuito. Fue una implementación sencilla ejecutar gracias a las ventajas que ofrece *YOLO V4* para el entrenamiento de *CNN* aplicadas a detección de objetos.



La conexión entre esta red y la aplicación móvil fue efectiva, por lo que la comunicación entre la aplicación móvil y la red se ejecuta de forma eficiente haciendo uso de los recursos del dispositivo instalado lo cual generó que los resultados en pantalla se mostrarán en un tiempo de aproximadamente 2 segundos considerado corto para el fin de la aplicación.

Con base a los resultados se determinó que la red logró identificar correctamente la mayoría de las estructuras dentales como objetos separados de acuerdo a las clases especificadas en el proceso de etiquetado y entrenamiento. También se observó que la red tuvo un mejor desempeño mostrando alta precisión y sensibilidad para el reconocimiento de estructuras dentarias de la clase 0, mientras que la clase 6 fue la más sensible.

Se determinó que, aunque se obtuvo un buen desempeño para algunas de las clases, en general según las métricas demuestran un desempeño desfavorable en la identificación, esto se debió a varios factores como la falta de imágenes, la calidad de las imágenes, la variabilidad de ángulos con que fueron tomadas las imágenes y el tipo de imágenes para cada clase.

## 6.2 Recomendaciones

Con la finalidad de mejorar la implementación de una CNN para la identificación de lesiones cariosas en estructuras dentarias se recomiendan las siguientes pautas:

- Para mejorar el desempeño de la aplicación móvil en la identificación de caries en las estructuras dentarias, es recomendable entrenar varias redes neuronales específicas según la vista (lingual, oclusal y vestibular) en que se toma la fotografía.
- Para mejorar el desempeño general de la CNN implementada en este proyecto se recomienda aumentar el conjunto de datos a por lo menos 7000 imágenes dividiéndolo en 1000 imágenes para cada clase, dada la forma en que trabajan las capas internas de la CNN y el reconocimiento de objetos se requiere de un conjunto grande para cada objeto que se quiera reconocer.
- Realizar aumento de imagenes usando rotaciones, reflejo, desplazamiento y zoom de las imágenes con conjunto de datos pequeño, para mejorar el reconocimiento de esas clases donde las métricas fueron desfavorables.
- Seguir investigando y desarrollando en este tipo de herramientas que sirven para aumentar el número de recursos disponibles para el diagnóstico de estas patologías.

# Bibliografía

apd (2018). ¿qué es el deep learning y por qué es clave para la inteligencia artificial? Disponibel en:https://www.apd.es/que-es-deep-learning.

AprendeIA (2021). Curvas roc y Área bajo la curva (auc). Disponibel en:https://aprendeia.com/curvas-roc-y-area-bajo-la-curva-auc-machine-learning.

Araújo, Teresa; Aresta, G. C. E. R. J. A. P. E. C. P. A. and Campilho, A. (2017). Classification of breast cancer histology images using convolutional neural networks. *PloS one*, 12(6):e0177544.



Arce, J. I. B. (2021). La matriz de confusión y sus métricas. Disponibel en:https://aprendeia.com/curvas-roc-y-area-bajo-la-curva-auc-machine-learning.

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

Bottenberg, P. Jacquet, W. B. C. S. V. and Jablonski-Momeni, A. (2016). Comparison of occlusal caries detection using the icdas criteria on extracted teeth or their photographs. *BMC oral health*, 16(1):93.

Calvo, D. (2017). Clasificación de redes neuronales artificiales. Disponibelen: https://www.diegocalvo.es/clasificacion-de-redes-neuronales-artificiales.

Colgate (2021). La caries dental. Disponibel en: https://www.colgate.com/es-py/oral-health/cavities/what-are-cavities.

Darrenl (2020). Labelimg. Disponibel en:https://github.com/tzutalin/labelImg.

Estai, Mohamed. Bunt, S. K. Y. K. E. and Tennant, M. (2016a). Diagnostic accuracy of teledentistry in the detection of dental caries: a systematic review. *Journal of Evidence Based Dental Practice*, 16(3):161–172.

Estai, Mohamed. Kanagasingam, Y. X. D. V. J. H. B. K. E. and Tennant, M. (2016b). A proof-of-concept evaluation of a cloud-based store-and-forward telemedicine app for screening for oral diseases. *Journal of telemedicine and telecare*, 22(6):319–325.

González, J. (2019). Diseño e implementación de una aplicación móvil inteligente en android para reconocimiento de lesiones y enfermedades cutáneas y en la mucosa bucal.

Google (2020). Colaboratory. Disponibel en:https://cutt.ly/nkBdRls.

Guy, T. A. (2020). Running a yolov4 object detector with darknet in the cloud. Disponibel en:https://cutt.ly/6kHzNeB.

Marcelo, I. (2020). Icdas-iccms: Sistema internacional para la detección y evaluación de caries incipiente. Disponibel en: https://www.sdpt.net/ICDAS.htm.

O.M.S (2020). Salud bucodental. Disponibel en: https://www.who.int/es/news-room/fact-sheets/detail/oral-health.

Quintero-Rojas, J. and González, J. D. (2021). Use of convolutional neural networks in smartphones for the identification of oral diseases using a small dataset. *Revista Facultad de Ingeniería*, 30(55):e11846–e11846.

Shakeel, H. (2020). tensorflow-yolov4-tflite. Disponibel en:https://github.com/haroonshakeel/tensorflow-yolov4-tflite.

Wang, C.-Y., Liao, H.-Y. M., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., and Yeh, I.-H. (2020). Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391.

# Apéndice A

# Comandos utilizados para entrenar la CNN

Para clonar el repositorio de Darknet

```
!git clone https://github.com/AlexeyAB/darknet
```



Para Habilitar el GPU y OPENCV repositorio de Darknet

```
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
```

Para Habilitar el GPU y OPENCV repositorio de Darknet

```
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
```

```
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
```

Para generar el archivo ejecutable de Darknet y poder correr el entrenamiento

```
!make
```

Para descargar los pesos preentrenados
de YOLO V4 entrenamiento

```
!wget https://github.com/AlexeyAB/darknet/releases/download/
darknet_yolo_v3_optimal/yolov4.weights
```



Para acoplar google drive con colab

```
%cd ..
from google.colab import drive
drive.mount('/content/gdrive')
!ln -s /content/gdrive/My\ Drive/ /mydrive
!ls /mydrive
```

Para copiar las carpetas de Drive a colab

```
!cp /mydrive/yolov4/obj.zip ../
!cp /mydrive/yolov4/test.zip ../
```

Para Iniciar el entrenamiento de la red

```
 %%capture
!./darknet detector train data/obj.data cfg/
yolov4-obj.cfg yolov4.conv.137 -dont_show -map
```