Desarrollo de aplicación Android móvil utilizando Visión Artificial y Deep Learning para la identificación de aguacates Hass con la plaga Monalonion, en la finca “Las Palmas”, ubicada en el municipio de San Agustín, Huila

William Ricardo Erazo Samboni.

20441514074

Universidad Antonio Nariño.

Programa Ingeniería Electrónica.

Facultad de Ingeniería Mecánica, Electrónica y Biomédica.

Neiva, Colombia.

2022

Desarrollo de aplicación Android móvil utilizando Visión Artificial y Deep Learning para la identificación de Aguacates Hass con la plaga Monalonion, en la finca “Las Palmas”, ubicada en el municipio de San Agustín, Huila

William Ricardo Erazo Samboni

Proyecto de grado presentado como requisito parcial para optar al título de:
Ingeniero Electrónico

Codirector (a):
Ph.D. Francisco Maximiliano Fernández Periche.

Línea de Investigación:
Mecatrónica y Robótica
Grupo de Investigación:
Percepción y robótica (GEPRO)

Universidad Antonio Nariño.
Programa Ingeniería Electrónica.
Facultad de Ingeniería Mecánica, Electrónica y Biomédica.
Neiva, Colombia.
2022

NOTA DE ACEPTACIÓN

El trabajo de grado titulado
_____, Cumple con
los requisitos para optar
Al título de _____.

_____

Firma del Tutor

_____

Firma Jurado

_____

Firma Jurado

Neiva,

(Dedicatoria)

A mi madre Luz Alba Samboni de Goschl, su esposo Eddy Sarrasin, mis hermanos Hugo Fernando Erazo y Gilder Erazo, porque gracias al apoyo de ellos fue posible la culminación de este proyecto. Por último, pero no menos importante, a Lina Bohusch por haberme acompañado y darme fuerzas para seguir adelante ante las adversidades. ¡Muchas gracias a todos!

Agradecimientos

Primeramente, doy gracias a Dios por haberme iluminado y dado salud. Agradezco enormemente a mi madre y su esposo por haberme apoyado incondicionalmente, porque sin ellos nada de esto sería posible, de igual forma a las demás personas que de alguna manera con sus ideas me ayudaron a culmina mi proyecto de grado.

Agradezco también a mi codirector del proyecto, el Doctor Francisco Maximiliano Fernández Periche por su apoyo incondicional, sus valiosas asesorías y recomendaciones para el desarrollo del proyecto.

Contenido

Pág.

Lista de Figuras

Pág.

Lista de tablas

Pág.

Lista de ecuaciones

Lista de Símbolos y Abreviaturas

Abreviaturas

Abreviatura	Término
DL	Deep Learning
ML	Machine Learning
XML	Extensible Markup Languaje

Resumen

En el departamento del Huila, en los últimos años se ha registrado un aumento en cultivos de aguacate Hass, por lo cual, se evidencia también la presencia de plagas como el Monalonion, considerada como la principal según los agricultores, la aparición de esta puede generar pérdidas económicas entre el 50 Y 100% (Eugenia & Zuluaga, 2020). El propósito de este proyecto fue desarrollar una aplicación Android móvil utilizando Visión Artificial y Deep Learning para la identificación del fruto con Monalonion. El conjunto de datos consolidado por el autor contiene imágenes reales de los aguacates, se empleó la técnica de aprendizaje por transferencia de Deep Learning y el modelo EfficientDet-Lite3, se evaluó en el entorno de Google Colab con la métrica mAP (mean Average Precision) que es muy popular entre diferentes competiciones y se obtuvo una precisión promedio de 90%.

Palabras Clave: Aguacate Hass, Plaga Monalonion, Aplicación Android móvil, Visión Artificial, Deep Learning.

Abstract

In the department of Huila, in recent years there has been an increase in Hass avocado crops, which is why there is also evidence of the presence of pests such as Monalonion, considered the main one according to farmers, the appearance of this can generate economic losses between 50 and 100% (Eugenia & Zuluaga, 2020). The purpose of this project was to develop a mobile Android application using Artificial Vision and Deep Learning for the identification of the fruit with Monalonion. The data set consolidated by the author contains real images of avocados, the Deep Learning transfer learning technique and the EfficientDet-Lite3 model are used, it was evaluated in the Google Colab environment with the mAP (mean Average Precision) metric which is very popular among different competitions and an average accuracy of 90% was obtained.

Key words: Hass avocado, Monalonion pest, Android mobile application, Artificial Vision, Deep Learning.

Introducción

En la actualidad la Visión Artificial y el Deep Learning son dos conceptos que se encuentran en varios campos de las ciencias, por ejemplo, en el área de la salud siendo utilizados para hacer diagnósticos médicos, en marketing para dividir o clasificar a sus clientes potenciales o hacer una correcta clasificación, también se encuentran estos conceptos en las redes sociales y la industria.

El Aprendizaje Profundo aprovecha dos sistemas con estructuras diferentes como la velocidad de procesamiento de la máquina y el reconocimiento de patrones, con ello haciendo imitación al comportamiento neuronal humano, implementando la incorporación de Redes Neuronales Artificiales (RNA) que funciona mediante varias capas que a su vez hacen una fragmentación de la información que se esté manipulando, convirtiéndose así en una metodología de procesamiento más eficiente y que a su vez otorga la posibilidad de manejar varios temas de manera paralela (Erik Leonel Otero & Yago Graña De Brasi, 2019).

De acuerdo a los conceptos mencionados, este trabajo de grado pretende obtener como resultado final un modelo de detección de objetos incorporado en una aplicación móvil para el Sistema Operativo Android, con el cual se va a poder hacer detección de la plaga Monalonion.

Los agricultores consideran esta plaga como la principal de la región (MinAgricultura, 2021). La aplicación final funcional será capaz de hacer detecciones dentro de una imagen capturada con la cámara del teléfono, se hizo detección de tres clases, “Monalonion” que es la plaga objetivo, “Sano” que hace referencia a un aguacate sin ninguna afectación y la clase “Enfermo” para lograr diferenciar al fruto correctamente. La construcción del algoritmo se da

mediante las bibliotecas de tareas de Tensorflow creadas por el equipo de Google Brain Team (*TensorFlow*, s. f.) que permite el uso de modelos de detección de objetos en un dispositivo móvil.

Además, se acude al uso de la plataforma Google Colaboratory (Google, 2022), que da acceso gratuito a GPU que es indispensable para la mejora de tiempos de entrenamiento.

Planteamiento del problema

Para tener un buen proceso en el cultivo del aguacate son necesarias cuatro condiciones las cuales son: La primera; es hacer un buen análisis de suelos donde se va a sembrar el aguacate, eso es indispensable debido a que hay tierras las cuales unas son mejores que otras, por otra parte, hay unas que definitivamente no sirven para sembrar el aguacate. La segunda; es contar con un buen material vegetal, es decir tener una semilla certificada, o un buen injerto, o una buena yema proveniente de una copa que pueda ser verificada en cuanto a la productividad. La tercera; es contar con un buen plan de fertilización. Y la cuarta condición; es la asociatividad, es decir asociarse con el vecino, o el amigo que tiene el cultivo de aguacate para llevar un plan de control de plagas y no se afecte ninguna de las partes.

Otro punto a tener en cuenta en el proceso de cultivo y comercialización del producto, es el tema del cuidado al medio ambiente, debido a que una de las primeras condiciones cuando se va a exportar el producto, es tener un certificado de trazabilidad del producto, es decir mostrar que un producto no tiene trazas de los insumos o de los productos con los cuales se ha preparado el fertilizante o producto con el cual se ha fumigado. Adicional a ello, uno de los aspectos que se tienen en cuenta en el tema de la cosecha y de poscosecha es en cómo se recoge el aguacate directamente del árbol, el cual se debe hacer cogiendo el aguacate y con unas tijeras cortarle el pecíolo. Nunca se debe arrancar de raíz, solo se debe cortar poca longitud del mismo, siempre y cuando no se le desprenda del todo, porque comienza a entrar en descomposición desde ese punto de donde se arrancó del árbol. Tampoco se puede dejar golpear el aguacate, se tiene que recoger y manipular de forma delicada y cuidadosa, porque si se deja golpear, cuando el aguacate madura y se destapa no va a estar en buenas condiciones de calidad para el consumo, en

esta etapa el fruto va a generar un mal sabor para el consumidor final y patologías en el mismo. Finalmente, hay que tener en cuenta tener un buen programa de fertilización, ya que también es un pilar fundamental básico en el cultivo del aguacate. Adicional a ello, en cuanto a los desarrollos que trae el cultivo del aguacate a la comunidad donde se presenta el mismo están: primero se encuentra el empleo que se genera con ello; segundo y consecuentemente a eso, unos mejores ingresos; en tercer lugar, lógicamente las prácticas ambientales adecuadas que se generan con todo el tema de la cosecha y postcosecha; Cuarto y último, un comercio justo.

De acuerdo a lo referido en todo el proceso de cosecha y post cosecha del aguacate Hass, se podría evidenciar que durante el proceso post cosecha, se puede llegar a dar la presencia de aguacates con algún inconveniente fitosanitario, principalmente la plaga Monalonion que es la principal en aparecer en el Departamento del Huila (MinAgricultura, 2021). De esta manera, teniendo en cuenta lo planteado anteriormente, surge el siguiente interrogante de investigación: ¿Cómo identificar de forma rápida y precisa aguacates tipo Hass con Monalonion?

Antecedentes

El total de la producción en un determinado cultivo puede verse afectada si no se hace una detección temprana de alguna plaga que se encuentre presente, para ello es posible utilizar técnicas de Inteligencia Artificial (IA) como lo es el Deep learning en conjunto con las Redes Neuronales Convolucionales (CNN), ya que estas ofrecen la posibilidad de automatizar la detección de plagas en cultivos y de esta manera dar una gran ayuda al agricultor y mejorar su productividad.

En la actualidad hay una serie de investigaciones en donde usan diferentes estrategias para cumplir sus objetivos encaminados al apoyo de la agricultura, entre las cuales se mencionan a continuación:

Detector para enfermedades de las plantas de tomate en tiempo real y reconocimientos de plagas En esta investigación utilizan Deep Learning como detector robusto para detectar enfermedades en el tomate, en donde los autores hacen selección de tres familias de detectores (Red neuronal convolucional basada en regiones más rápida (Faster R-CNN), Red totalmente convolucional basada en regiones (RFCNN) Y detector Multicaja (SSD)) y mediante la evaluación seleccionan el más conveniente el cual fue el primero (Faster R-CNN), lograron una precisión del 83%, para hacer evaluación y reconocimiento de la composición de alimentos utilizan visión por computador y para conseguir una eficiente detección hicieron uso de Redes Neuronales Convolucionales (CNN)(Fuentes et al., 2017).

Reconocimiento de alimentos basado en aprendizaje profundo En este trabajo utilizaron aprendizaje profundo y aprendizaje de transferencia, haciendo uso de la librería Tensorflow, además, escogieron el modelo Inception-Resnet-Modelov2 para realizar la

clasificación de frutos de Cacao donde el objetivo principal fue detectar si se encontraba Maduro o no, al finalizar su proyecto se obtuvo una precisión del 90% (Qian Yu et al., 2019).

Sistema de monitorización y detección de plagas en cultivos aplicando algoritmos de Deep Learning "En este proyecto se presenta un diseño global de un sistema de monitorización de un campo de cultivo, en el que, mediante sensores y diferentes tipos de dispositivos de recolección de datos, UAV y cámaras para obtención de imágenes, se monitorizan constantemente datos ambientales, de la tierra y de las plantas" (Ferrer Martínez, 2021).

Prototipo web para predicción y detección de incendios forestales en los cerros orientales de Bogotá, mediante una red de sensores e inteligencia artificial "El objetivo de este proyecto es el darle un uso asertivo a la información suministrada por los sensores, como lo es sus características, datos, localización, la creación de una inteligencia artificial entrenada con los datos suministrados, y la administración de dichos datos, la creación de dicha tecnología e integración de la inteligencia artificial se realizara mediante una metodología cualitativa en la que se reunirá la información necesaria para la construcción de una plataforma web" (Castillo Sarasty & Presencial, 2021)(Castillo Sarasty & Presencial, 2021).

Identificación de síntomas de Huanglongbing en hojas de cítricos mediante técnicas de Deep Learning "En este trabajo se describe el desarrollo de una aplicación móvil que utiliza técnicas de Deep Learning para identificar síntomas de Huanglongbing y carencias nutricionales en hojas de árboles cítricos" (Berger et al., 2019).

Aplicación móvil para detección y tratamiento de daños de los cultivos de la parroquia Taura del cantón Durán, mediante el uso de software de análisis de imagen basado en técnicas de Machine Learning "El proyecto se basa en identificar las plagas y

enfermedades más conocidas de la parroquia Taura del cantón Durán, obtener una base de datos de imágenes de cada plaga, enfermedad o daño para entrenar una red neuronal con algoritmos de redes convolucionales usados en machine Learning y generar un modelo que sea capaz identificar las plagas, enfermedades y daños que estén presentes en la planta capturado por el sensor de imagen de los dispositivos móviles" (Calderon & Cortes, 2020).

Diagnóstico automatizado para la clasificación de café trillado con broca mediante procesamiento de imágenes con Deep Learning "En este trabajo, se plantea diagnosticar automáticamente la clasificación en los granos de café trillado que están infectados por la enfermedad de la broca" (de Ingeniería et al., 2019).

Sistema de visión artificial para apoyar en la identificación de plagas y enfermedades del cultivo de sandía en el distrito de Ferreñafe: "Este proyecto desarrolló una aplicación móvil para identificar y brindar el agroquímico correcto para las plagas o enfermedades del cultivo de Sandía en el distrito de Ferreñafe. Utilizando características consecuentes del agente hospedante en el cultivo de sandía y los algoritmos de la inteligencia artificial, se pretende hallar patrones relevantes en las imágenes de forma rápida y confiable (Piscoya Ferreñan, 2019).

Objetivos

Objetivo general

Desarrollar una aplicación Android móvil utilizando Visión Artificial y Deep Learning para la identificación de Aguacates Hass con la plaga Monalonion en la finca “Las Palmas”, ubicada en el municipio de San Agustín, Huila.

Objetivos específicos

- Diseñar un algoritmo empleando el lenguaje de programación Python para la implementación de Visión Artificial y Aprendizaje profundo dedicado a la identificación aguacate Hass con la plaga objetivo.

- Consolidar un conjunto de datos que permitan calibrar una máquina de aprendizaje para la identificación de aguacates enfermos mediante la toma de imágenes.

- Diseñar la interface gráfica (Front-End) teniendo en cuenta el pre-entrenamiento del algoritmo dedicado al propósito expuesto anteriormente.

- Construir una aplicación funcional que integre un algoritmo de aprendizaje de máquina para la identificación de los frutos de aguacate con la plaga.

Justificación

Desde hace aproximadamente dos décadas empezó la producción de aguacate Hass en Colombia (Garzón, 2020), es decir se empezó a realizar el proceso de cosecha, post cosecha y comercialización de mismo, con cerca de 1200 hectáreas sembradas. En el departamento del Huila, el 96% de los productores se dedican a la industria del café, pero actualmente ven la producción de aguacate en la variedad Hass como una buena alternativa para diversificar sus cultivos (caracol.com, 2021).

Con la introducción al cultivo de este fruto, los agricultores se han visto involucrados con la aparición de plagas y enfermedades, entre ellas la principal, el Monalonion (MinAgricultura, 2021).

En contraste, se deben tener en cuenta condiciones fitosanitarias, que es parte fundamental para la comercialización local y aceptación para la exportación del fruto.

Con esta información se identifica una problemática y se procede a implementar una estrategia para brindar ayuda al productor, es así que este proyecto, tiene como finalidad la identificación de aguacates tipos Hass con Monalonion. Los frutos crecen con diversas plagas, haciendo que los agricultores que cuentan con estos cultivos tengan grandes pérdidas económicas, además de tiempo y esfuerzo, así mismo se busca optimizar el proceso de clasificación de la fruta, ya que esto requiere de conocimientos y concentración al momento de identificar los aguacates que no serían aptos para ser comercializados, destacando que para las personas nuevas en el tema o que no cuenten con mucho conocimiento se les facilite mucho más esta tarea. Adicional a ello, otra de las motivaciones en la realización de la investigación y diseño

de este tipo de proyectos en la industria del aguacate, es que con la aparición del virus del COVID-19 a finales del año 2019, se han generado cambios o modificaciones en los procesos que se hacían antes de forma totalmente manual y visual por las personas. Por esta razón, como se menciona en el artículo desarrollado por (Fuentes Pérez, 2020) “, se debe realizar un breve análisis de ciertos aspectos de la industria alimentaria en general, que se cree que deberán cambiar en función de la adaptación a la realidad que se tendrá que vivir luego de la pandemia”, resultado de la aparición del COVID-19.

CAPITULO 1: Marco teórico

En este capítulo del presente documento se centra en la definición de los conceptos que fueron necesarios para el desarrollo, así como también las diferentes herramientas.

1.1 Estado del Arte

El aprendizaje profundo (DL), cada vez va adquiriendo mayor fuerza en el ámbito de la Inteligencia Artificial (IA), esta es una subcategoría del aprendizaje automático, emplea redes neuronales artificiales para un mejor reconocimiento ya sea de la voz, la visión por computadora o el lenguaje natural (Rostros & Algoritmos De Reconocimiento De Objetos De La Biblioteca Opencv Edison Rene Caballero Barriga, 2017).

Microempresas como AGROCOTOPAXI crearon un modelo de predicción con el fin de dar a conocer las bondades de las plantas, lograron la implementación de esta inteligencia artificial utilizando herramientas como el software Django, Tensorflow y Xamarin, además del lenguaje de programación Python. Para el desarrollo de la aplicación móvil finalmente utilizaron la metodología SCRUM y para el modelo CRISP-DM (Edison et al., 2020).

Adicional a los proyectos que se mencionaron, el Deep Learning de la mano con la Visión Artificial en la industria 4.0, tienen infinidad de aplicaciones, en cuanto al ámbito automovilístico permiten la incorporación de modelos autómatas con múltiples propósitos tales como detectar el cansancio del conductor de tal manera que le hace un llamado para así evitar un accidente, como también brindarle una seguridad extra al estar monitoreando las variables del entorno mientras se conduce.

En el ámbito de la investigación y la educación, un grupo de estudiantes de la Universidad Señor de Sipán, en el año 2021 implementaron un proyecto para la selección de aguacate Hass aceptables por el mercado en cuestión de calidad utilizando técnicas de visión por computadora el cual consiste en identificar frutos con queresas o plagas y defectos menores como lo es el color. Dicha técnica consistió en la utilización de 1260 imágenes, 210 frutos distribuidos en 3 niveles, de ahí se extraen seis imágenes de cada fruto de aguacate Hass siendo esta la etapa de pre-procesamiento y filtrado en la cual se usaron descriptores llamados (K-Means), por último, se procede a la etapa de clasificación mediante Segment Color, Segment Rose y K-Means. Estos clasificadores dan como resultado el clasificador llamado KNN convirtiéndose en el más óptimo para la clasificación con un 77% de precisión (Quiroz et al., 2021).

Por otra parte, va ligado en aplicaciones de dispositivos interconectados como es el caso de IoT (Internet Of Things), con ello mejorando la productividad en modelos de negocios y permitiendo a la vez la transformación digital (Nexusintegra, 2022).

1.1.1 El departamento del Huila y Aguacate hass

Según Diario del Huila, 2020), huila es el departamento con mayor potencial para convertirse en el principal productor de aguacate en Colombia, superando a grandes productores como Antioquia y Tolima, cuenta además con un área de 21.849 hectáreas aptas para la siembra de este cultivo.

En contraste con otros departamentos del país, Huila es el que tiene el área más adecuada para la producción de aguacate, cumple con todas las condiciones climáticas necesarias. En la

actualidad presenta un crecimiento del 26 % en áreas sembradas, lo cual representa una cercanía a 3.4000 hectáreas (caracol.com, 2021).

Adicionalmente, cabe resaltar que San Agustín e Isnos son los municipios con mayor área sembrada de aguacate hass y se estima que se produce alrededor de 4.000 kilos por hectárea (ICA, 2017).

1.1.2 Monalonion

Su nombre científico es Monalonion velezangeli perteneciente al orden Hemíptera de descendencia Miridae y también conocido como chinche del aguacate o coclillo (Pérez et al., 2014), genera pérdidas en la producción del mismo entre el 50% y el 100% (Eugenia & Zuluaga, 2018). Se caracteriza por tener manchas dentro y fuera de la areola de las alas anteriores, y por tener sitios de ovoposición o incubación en los tallos de ramas jóvenes, debajo de las hojas, en lugares de poca incidencia de la luz solar. La identificación de daños frescos como manchas pequeñas y grandes negras, que evidencian la presencia de este insecto en el fruto, son una herramienta útil para el monitoreo y evaluación de incidencia de esta plaga, debido a la dificultad de observar estos insectos de manera directa (Juliana Moraes Boldini, 2020).

Figura 1-1: Insecto Monalonion

Fuente: Adaptado de (Marisol Giraldo Jaramillo, 2010)

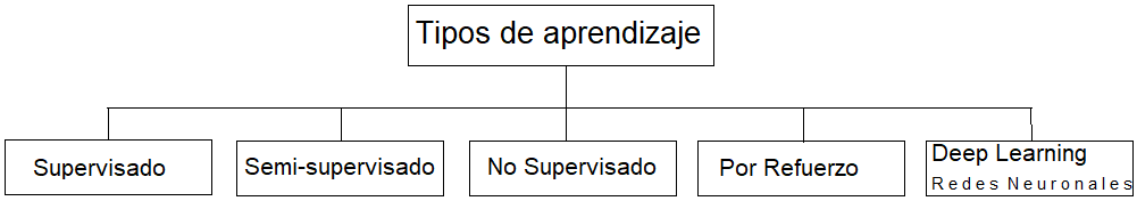
1.2 Machine Learning

Es un subconjunto de la Inteligencia Artificial el cual utiliza técnicas de (Deep Learning) en donde la máquina o sistema de acuerdo a su experiencia mejora sus tareas (Microsoft Learn, 2022a). Es capaz de reconocer patrones dentro de los datos para arrojar predicciones, esta habilidad la utilizan plataformas como Netflix a hacer las sugerencias a sus usuarios o Spotify, también inteligencias como Alexa de la empresa Amazon y Siri de Apple (BBVA, 2021).

1.2.1 *Tipos de aprendizaje*

En el campo del Machine Learning se encuentran diferentes tipos de aprendizaje, entre los cuales está el aprendizaje supervisado, semi-supervisado, no supervisado, por refuerzo y deep learning que permite hacer uso de redes neuronales.

Figura 1-2:Tipos de aprendizaje.



Fuente: Figura hecha por el autor.

- *Aprendizaje Supervisado*

Esta es una de las maneras en el machine learning donde el sistema requiere conocer las entradas y salidas, consiste en dar etiquetas al conjunto de datos y con ello se consiguen predicciones, donde el sistema deberá cuantificar su precisión (TIBCO Software, 2020)(TIBCO Software, 2020).

- *Aprendizaje semi-supervisado*

Esta es una técnica que hace una combinación del Aprendizaje supervisado con el No supervisado, usualmente se le da uso, cuando se tiene una cantidad muy grande de datos para etiquetar, este método genera la posibilidad de obtener un modelo de predicción que sea capaz de funcionar mejor en comparación con uno que contenga únicamente datos etiquetados, reduciendo así costos computacionales (Ignacio Herrera & Alejandro Figueroa, 2016).

- *Aprendizaje no supervisado*

En este tipo de aprendizaje a diferencia del aprendizaje supervisado sucede lo contrario, ya que aquí no es necesario conocer las salidas del sistema, pero si las entradas, es importante precisar, que este tipo de aprendizaje es más utilizado en el área de clasificación de objetos; se tiene una definición de las etiquetas en los datos, entonces se deberá inferir las estructuras de los datos que se emplean en determinado caso. Para lograr esto, se puede hacer por medio de procesos de matemática para contrarrestar la redundancia o también haciendo una agrupación de elementos que coincidan (Russo et al., 2019).

- *Aprendizaje por refuerzo*

El Aprendizaje por refuerzo (Reinforcement learning), es una técnica que tiene como propósito, la optimización de un resultado en un problema, es decir se centra en la técnica prueba y error, aquí se mapea un canal de situaciones de acuerdo a la experiencia del sistema en donde se maximiza un escalar al cual se le llama “Señal de refuerzo o recompensa” (Leslie Pack Kaelbling et al., 2015).

- *Deep learning (Redes Neuronales)*

El deep learning es un método especifico que se encuentra dentro del machine learning, permite la incorporación de redes neuronales. Caracterizado por poseer redes neuronales complejas que emulan como el ser humano piensa, son técnicas muy utilizadas en detección de objetos, reconocimiento de voz y visión artificial. Es capaz de aprender de datos no estructurados y aprende de ellos de manera iterativa (*¿Qué Es Deep Learning? - Colombia | IBM*, 2021).

1.3 Visión Artificial

Es la ciencia que utiliza y combina cámaras, informática perimetral (Salazar Gualoto, 2020), software, e Inteligencia Artificial (Corvalán, 2018) para facultar que los sistemas computacionales puedan "ver" e identificar cualquier tipo de objeto. Es un campo de la Inteligencia Artificial la cual les posibilita a las computadoras la observación, el análisis y la comprensión según la información que previamente se haya tomado de una imagen digital, un video u entrada visual. En comparación con la visión humana, la Visión Artificial tiene como propósito aprender mediante un entrenamiento que el humano hace por medio de Datos y Algoritmos. Es también conocida como visión computarizada o visión técnica (*¿Qué Es La Visión Artificial? | IBM*, s. f.).

1.3.1 *Detección de Objetos*

Es una salida clave para el algoritmo de Deep learning, en contraste con los humanos que son capaces de mirar determinado objeto y detectar a qué clase pertenece. De manera similar, se enseña a la computadora a que adquiera un grado de conocimiento sobre un objeto y lo detecte

ya sea dentro de una imagen digital o un video en donde se va a arrojar el resultado por medio de un cuadro delimitador o bounding box (*Detección de Objetos - The Machine Learners*, 2021).

1.4 Redes Neuronales

Son un tipo de sistema artificial y simplificado, basado en el reconocimiento de patrones del cerebro humano. Es decir, 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 (Kai fu lee, 2020).

1.4.1 *Aprendizaje por transferencia en Deep Learning*

Esta técnica de Deep Learning es comúnmente ahora empleada, por error se cree que entrenar un modelo de Deep Learning requiere millones de cantidad de datos mientras que con esta técnica no será necesario porque el modelo de partida ya se encuentra entrenado con un conjunto de gran tamaño.

Es un método que a menudo es utilizado en aplicaciones de detección de objetos, el aprendizaje por transferencia básicamente consiste en la toma de un modelo para entrenar otro, pero considerando que el modelo de partida realiza una tarea similar a la que se desea con el nuevo entrenamiento deseado, otro punto importante es la reducción en requerimiento de costos computacionales y tiempos de entrenamiento entre otros como el aprovechamiento de arquitecturas de modelos desarrollados por los investigadores de este campo de AI (*Transfer Learning - MATLAB & Simulink*, 2022).

1.4.2 Redes Neuronales Convolucionales (CNN).

Una Red Neuronal Convolucional (CNN o ConvNet), es un tipo de arquitectura perteneciente a la técnica Deep Learning de Inteligencia Artificia, con ducha arquitectura no hay necesidad de extraer las características de forma manual, sino, ella aprende directamente de los datos, son también capaces de clasificar datos sin imágenes como es el caso de sonido, series temporales y señales. Por lo general las aplicaciones que utilizan técnicas de detección de objetos y visión artificial dependen de manera sustancial de una Red Neuronal Convolucional (MATLAB & Simulink, 2022).

Las CNN al ser un tipo de Red Neuronal Artificial con Aprendizaje Supervisado, su funcionamiento se centra en la imitación del córtex visual del ojo humano para identificar rasgos de entrada (Cifuentes et al., 2019).

Las 3 capas más frecuentes en esta área son: convolución, activación o ReLu, y Max Pooling. Esta operación llega a repetirse por docenas o cientos de veces según parámetros para así cada capa poder cumplir el objetivo de identificar diferentes características de una imagen.

Figura 1-3:Ejemplo de una red neuronal desarrollada con varias capas convolucionales

Fuente: Adaptado de (Calvo, 2017).

1.4.3 Capa convolucional

Este layer es el encargado de aplicar filtros de tres dimensiones en tamaños reducidos, hace un recorrido por la imagen de izquierda a derecha y de arriba a abajo para obtener la salida de la capa (Torralba, 2022).

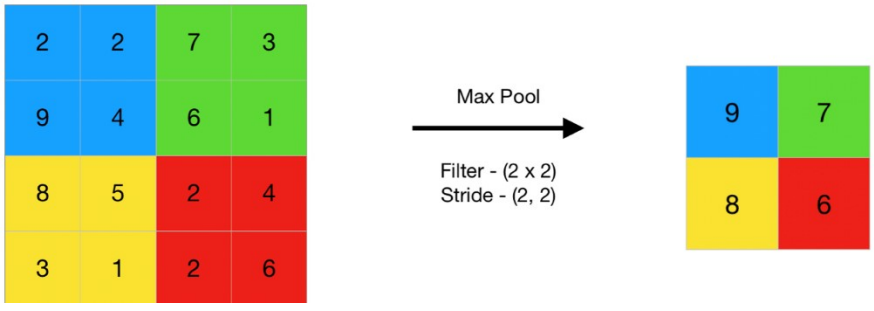
1.4.4 Capa de activación o ReLu

En la actualidad es la capa más usada debido a que ofrece mayor precisión y mayor velocidad, entrando a reemplazar a otras que usaban funciones como sigmoide. La capa de Activación (ReLu) normalmente sigue después de que se cumpla el proceso de la capa convolucional (Felipe et al., 2018).

1.4.5 Max Pooling

Es una operación muy importante y se encuentra dividida en dos partes, por un lado, está la zona de la imagen en donde se van a agregar las características y segundo es la operación de agregación que decide el método de mezclar las características locales previamente obtenidas (Juan I. Forcen et al., 2018). Generalmente esta capa es usada para reducir el sobreajuste en la red (Overfitting) y también varianza en la translación (Politécnica et al., 2018).

Figura 1-4: Ejemplo de Max Pooling



Fuente: Adaptado de (Kaggle, 2022)

Como se observa en la **Figura 1-4,** al realizarse el proceso de discretización mediante Max Pooling, la unidad de la capa toma la característica más representativa de la región 2x2 y la representa en de una forma más condensada como se observa en la imagen donde se obtuvo dicha representación proveniente de una matriz 4x4. Hay otros métodos en los que involucra usar la norma l^2o un promedio que se distingue por ser pesado que se enfoca en el análisis de la distancia al pixel central.

En el proceso de convolución se aplican varió filtros donde se originan nuevos mapas de características, entonces Max-pooling lo que hace es aplicar su principio a cada mapa que originó la capa convolucional de tal manera que si la capa produjo 3 mapas de características entonces automáticamente van a haber un mapa de pooling de tres canales.

1.5 EfficientDet

En este apartado se explicará cómo está compuesto y la funcionalidad de este modelo.

Primeramente, pertenece a la familia de modelos enfocada a la detección de objetos, fue desarrollado en el año 2020 por Google Brain Team (Brain, 2021), es un modelo que consigue una consistencia en su eficiencia con menor cantidad de parámetros y sobresale considerablemente frente a otros detectores como Yolo y Resnet. Presta grandes beneficios frente a escases de recursos computacionales.

El modelo es el sucesor de EfficientNet y propone el empleo de una red piramidal de características bidireccional ponderada (BiFPN), permite realizar la combinación de características de manera más rápida y con mayor brevedad, también propone utilizar el método de escalado compuesto el cual hace un escalamiento compuesto frente a la resolución,

profundidad y el ancho para el total de las redes de detección de objetos que tienen también las características y la caja/clase (Google Brain Team, 2020).

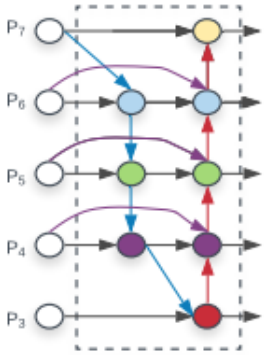
1.5.1 Definición de red piramidal

Las redes piramidales son aplicables a las redes neuronales convolucionales, estas redes trabajan de manera independiente a la red que se emplea, es decir que opera en compañía de la red neuronal convolucional y es capaz de brindar una mayor precisión a la hora de hacer la detección de un objeto dentro de una imagen. Para este proyecto se centró principalmente en la red BiFPN que es la que describe la función del modelo EfficientDet.

1.5.2 Red Piramidal de Características Bidireccional Ponderada (BiFPN)

Es una red piramidal de características bidireccional ponderada la cual da la posibilidad de fusionar características multiescala de manera fácil y rápida, tiene la idea de combinar funciones multinivel, hacen que fluya la información en dirección de arriba hacia abajo y caso contrario (Google Brain Team, 2020). Estas funciones son: FPN, PANet y NAS-FPN y se explican a más adelante.

Figura 1-5: Arquitectura de red BiFPN



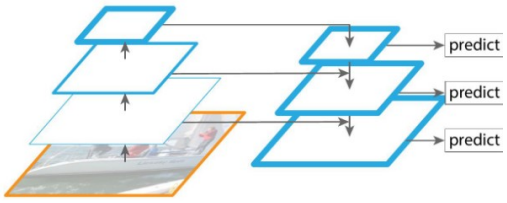
Fuente: Adaptado de(Google Brain Team, 2020)

Las BiFPN también utilizan una función breve de normalización rápida, por lo regular los otros métodos suelen ir encaminados a tomar las funciones que entran a la FPN todos de la misma forma y además de entradas que tienes resoluciones diferentes, entonces la red piramidal de características bidireccional ponderada lo que hace es agregar unos pesos más significativos de tal manera que la red aprenda la importancia de cada entrada. Esta nueva red de última generación presenta una optimización suprimiendo nodos que poseen un solo borde de entrada cosa que no sucede con PANet y PANet quienes generan una mayor exigencia de recursos informáticos, si la red detecta que se encuentra en un mismo nivel, ella agrega un nodo que va desde la entrada del nodo genuino hasta la salida del mismo. Ella trata cada red bidireccional como una capa de características (Robert et al., 2019).

1.5.3 Red piramidal de funciones (FPN)

FPN (Feaute Pyramid Network), esta red es usada en redes Convolucionales de Deep Learning para dar una solución genérica en la construcción de pirámides de características en tareas de detección de objetos. FPN toma como entrada una imagen con un tamaño determinado y arroja como resultado un mapa de características (Lin et al., 2017).

Figura 1-6: Red de función FPN

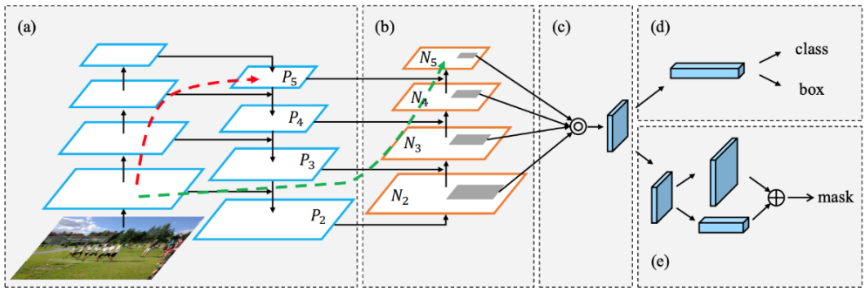


Fuente: Adaptado de (Lin et al., 2017)

1.5.4 Red de agrupación de rutas PANet

PANet (Path Aggregation Network) es una red que se encarga de la agregación de rutas con el objetivo de agilizar el flujo de la información. Según la **Figura 1-7**, en el paso (c) se puede observar la llegada de las características de FPN para hacer la agrupación según PANet (Shu Liu et al., 2018).

Figura 1-7: Representación de agregación de rutas.



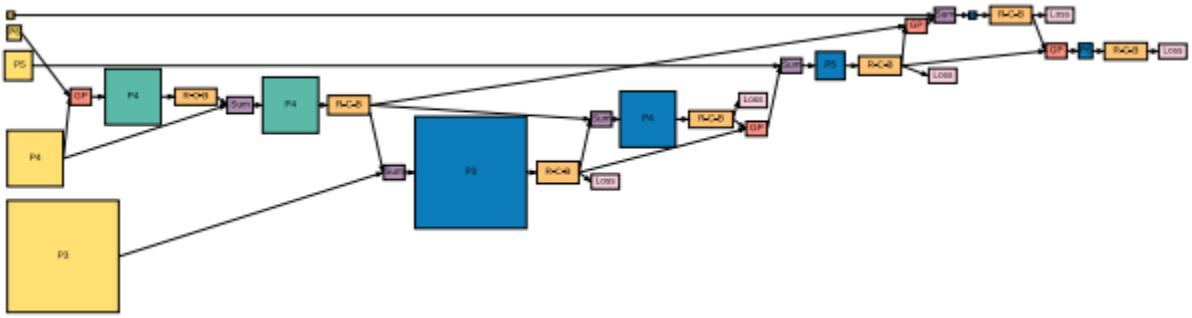
Fuente: Adaptado de (Robert et al., 2019)

1.5.5 NAS-FPN (Aprendizaje de arquitectura piramidal de características escalables para la detección de objetos)

También es una red piramidal en donde su funcionamiento consiste en la fusión de características por medio de escalas, tarea que cumple al hacer combinaciones de las conexiones que van de arriba abajo y de abajo hacia arriba (Ghaisi et al., 2019).

La **Figura 1-8** muestra una entrada de cinco capas (Cajas amarillas) desde arriba hacia abajo y cinco salidas que son representadas con las cajas de color azul.

Figura 1-8: Representación estructura NAS-FPN.

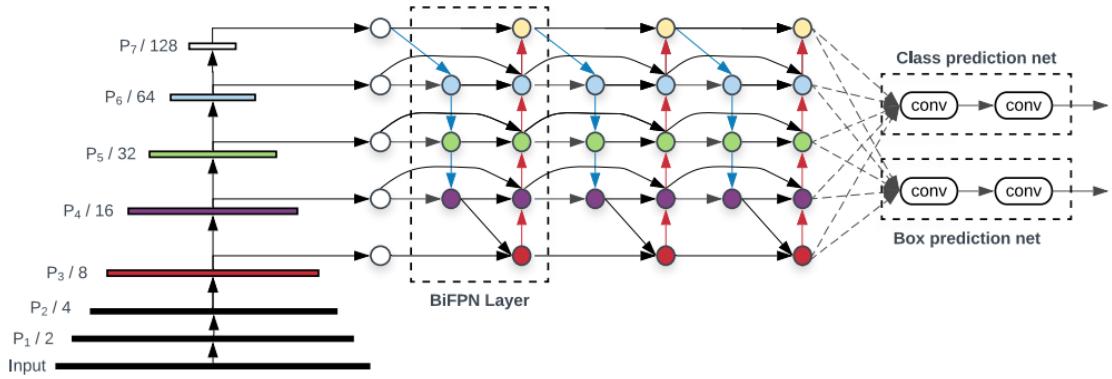


Fuente: Adaptado de (Xiaoshu, 2020)

1.5.6 EfficientDet

EfficientDet propone una diferente forma de realizar tareas de detección de objetos mediante una combinación de redes piramidales bidireccionales las cuales brindan una mejor optimización y mayor precisión. Además, la posibilidad de poder incorporar estos modelos de detección a dispositivos móviles(Google Brain Team, 2020).

Figura 1-9: Arquitectura general de EfficientDet



Fuente: Adaptado de (Google Brain Team, 2020)

Como se observa en la **Figura 1-9**, la arquitectura de EfficientDet se basa en EfficientNet como red troncal y BiFPN para poder hacer la detección de las características como la clase y la

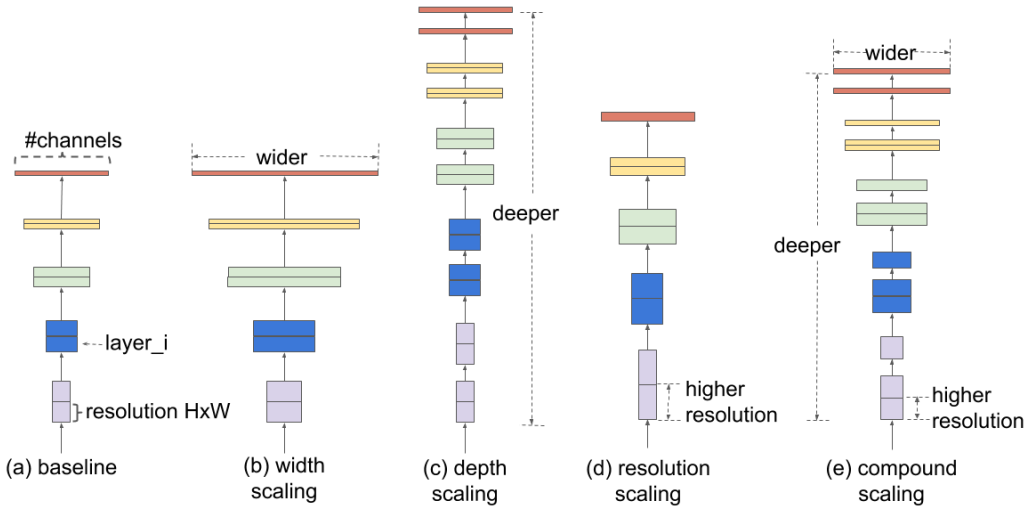
colocación del recuadro delimitador, también se observa que en la estructura hay tres bloques principales, los cuales son:

- *Bloque de columna*

La tarea que aquí se cumple es que, al ser asignada una imagen, se extraen las características para posteriormente pasar por un bloque de fusión.

En la **Figura 1-10** se observa el método de escalado que se aplica uniformemente a todas las dimensiones de profundidad, ancho y resolución que tenga la red con un conjunto de coeficientes de escala que son fijos, estos pasos anteriores son los originalmente utilizados en EfficientNet del equipo de Google el cual fue publicado años atrás, pero que EfficientDet se basa en él y presenta una mayor eficiencia antes sus antecesores.

Figura 1-10: Método de escalado

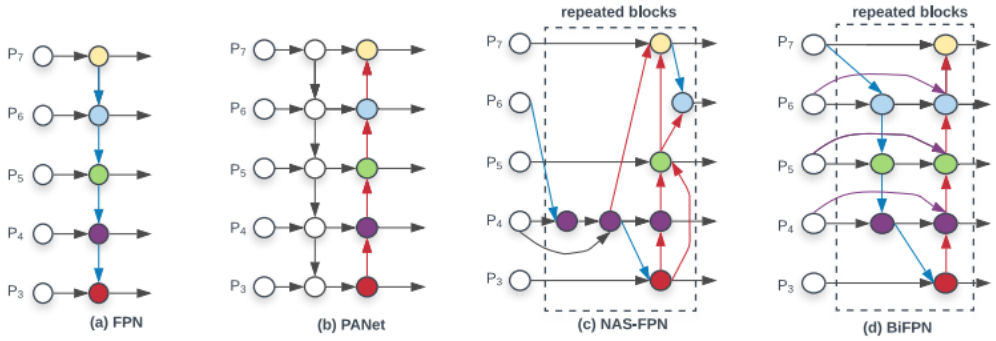


Fuente: Adaptado de (Google Research, 2019)

- *Bloque de fusión de Características Multiescala*

Se puede describir el funcionamiento de este bloque observando que primeramente FPN genera una combinación desde la parte superior hacia la inferior lo cual mezcla las entidades multiescala que va desde el nivel 3 al nivel7, seguido PANet si se observa se evidencia la agregación de una nueva ruta que va en dirección ascendente, luego NAS-FPN se encarga de hallar una arquitectura neuronal donde requiere hallar una topología de red que posee características irregulares para luego de cumplido lo dicho lo que hace es repetir este ciclo y por último BiFPN se encarga de brindar mejores compensaciones de eficiencia y precisión a lo que le entregan todas las etapas anteriormente expuestas (*EfficientDet: Scalable and Efficient Object Detection*, 2020).

Figura 1-11: Bloque de fusión de características multiescala (BiFPN)

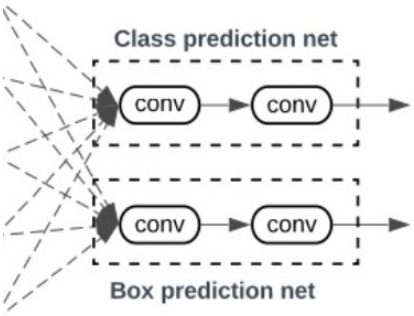


Fuente: Adaptado de (*EfficientDet: Scalable and Efficient Object Detection*, 2020)

- *Bloque de red de clase y cuadro delimitador*

Este bloque es el encargado de mostrar las predicciones de un objeto en una imagen, recibe las características que se obtuvieron en las capas anteriores de BiFPN y emplea dos capas completamente conectadas que van a realizar dicha tarea, una capa es la encargada de localizar las coordenadas donde estará el cuadro delimitador del objeto en la imagen y la otra capa va a definir a que clase pertenece el objeto que se detectó al igual que un puntaje de que hace referencia a la confianza con que se definió determinada clase.

Figura 1-12: Bloque de predicción



Fuente: Adaptado de (*EfficientDet: Scalable and Efficient Object Detection*, 2020)

1.5.7 EfficientDet-Lite

Esta familia de detección de objetos parte de la misma arquitectura EfficientDet, se encuentra adaptada para dispositivos móviles y también IoT (Apache et al., 2020). Por esta razón, con estos modelos se pudo lograr la tarea de detección de la plaga objetivo en este proyecto de grado.

La **Tabla 1-1** muestra las características principales de los 5 modelos disponibles actualmente:

Tabla 1-1: Familia de modelos EfficientDet-lite

Model architecture	Size(MB)*	Latency(ms)**	Average Precision***
EfficientDet-Lite0	4.4	37	25.69%
EfficientDet-Lite1	5.8	49	30.55%
EfficientDet-Lite2	7.2	69	33.97%
EfficientDet-Lite3	11.4	116	37.70%
EfficientDet-Lite4	19.9	260	41.96%

Fuente: Adaptado de (Apache et al., 2020)

1.6 Métrica de evaluación mAP

mAP (mean Average Precision), es una métrica popular entre competiciones de detección de objetos, esta técnica se usa con el fin de evaluar los resultados de una tarea de detección de objetos y su valor de precisión promedio se calcula en un rango de 0 a 1, donde 1 hace referencia al 100 % de acierto en una predicción. Dichos valores se encuentran calculados sobre valores de recuperación a recall (Shah, 2022).

Por lo general, en un conjunto de datos pueden encontrarse diferentes tipos de clases, eso quiere decir que no se tiene un conjunto de datos uniforme y si se emplea cualquier métrica simple, esta va a introducir sesgos en la evaluación. Por esta razón, nace la necesidad de obtener un “puntaje de confianza” ideal para cada cuadro delimitador y se introdujo la precisión promedio (AP) que en el caso de este proyecto AP Y mAP son los mismos términos (C Arlen, 2018) ya que el modelo empleado se encuentra entrenado mediante la base de datos Microsoft COCO (Microsoft, 2021).

Para poder comprender más a fondo la métrica mAP, es necesario comprender algunos términos como la recuperación de un clasificador (Recall), la precisión, Intersección sobre unión IoU y la matriz de confusión, pero antes cabe definir los siguientes:

- **Falsos Positivos (FP)** Cuando se hace una predicción y no corresponde a los datos etiquetados.
- **Falsos Negativos (FN)** El modelo dice que la predicción no corresponde a una clase la cual si es equivalente a los datos etiquetados.
- **Verdaderos Positivos (TP)** El modelo hace una predicción correcta con respecto los datos etiquetados.

1.6.1 Precisión

Para definir este término, cabe decir que hace referencia a la proporción de detecciones positivas correctas que se hicieron definiéndose con la **ecuación 1-1**

Ecuación 1-1: Cálculo de precisión

$$Precisión = \frac{TP}{TP + FP}$$

1.6.2 Recall

La recuperación de un clasificador se define como la cantidad de detecciones positivas reales identificadas en la tarea de detección de objetos y matemáticamente se puede calcular con la **ecuación 1-2**

Ecuación 1-2: Cálculo de recall

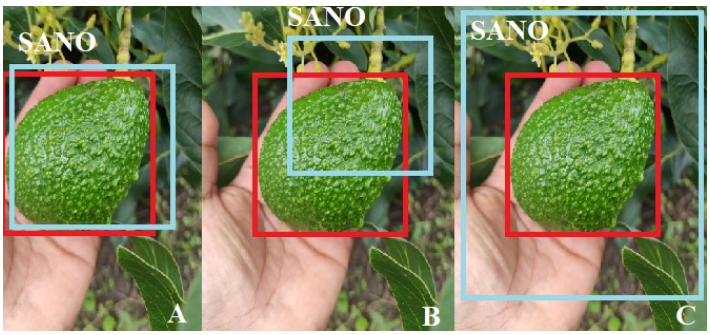
$$Recall = \frac{TP}{TP + FN}$$

1.6.3 Intersección sobre unión IoU

Este término hace referencia a un valor que cuantifica el grado de superposición entre dos recuadros que en este caso sería de detección de objetos. Hace evaluación de la superposición entre la región Ground True que se refiere al recuadro original etiquetado y a la Predicción que es la ubicación del nuevo recuadro que genera el detector, dicho de otra menara, es una métrica que posibilita medir la exactitud de una predicción (Kukil, 2022).

De acuerdo a la **Figura 1-13**, si se analiza los otros modelos diferentes de A, llega la necesidad de penalizar estas métricas ya que C tiene una mayor superposición que A, pero involucra el fondo y no se acerca al recuadro rojo y B no tiene buena precisión en cuanto a la localización.

Figura 1-13: Cálculo de IoU

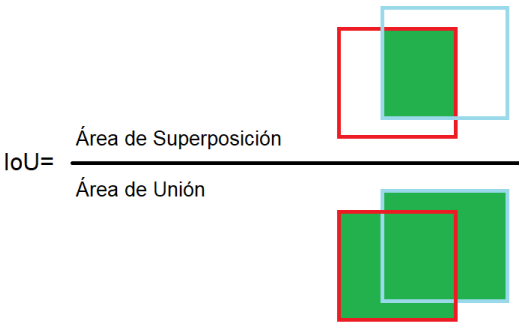


Fuente: Figura hecha por el autor.

1.6.4 Diseño de la métrica IoU

Es el grado de superposición entre la predicción y el recuadro original. El valor a calcular va desde 0 a 1, donde 0 quiere decir que no hay superposición y 1 indica una superposición perfecta. La operación IoU se puede observar en la **Figura 1-14**.

Figura 1-14: Diseño de IoU



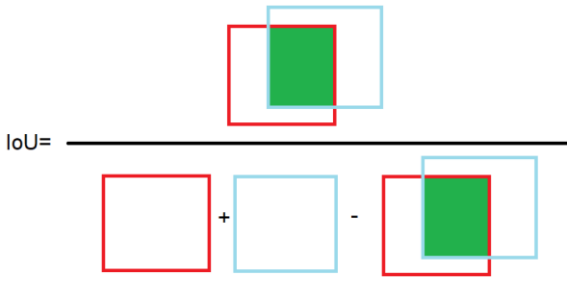
Fuente: Adaptado de (Kukil, 2022)

Si el área del recuadro que se detecta es mayor, el denominador se hace más grande, haciendo que IoU sea menor. En la **Figura 1-15** muestra la suma de dos veces el área de la intersección de acuerdo a la ecuación

Ecuación 1-3: Cálculo de IoU

$$IoU = \frac{Área\ de\ Intersección}{Área\ de\ verdad\ fundamental + Área\ de\ recuadro\ de\ predicción - Área\ de\ intersección}$$

Figura 1-15: Operación IoU

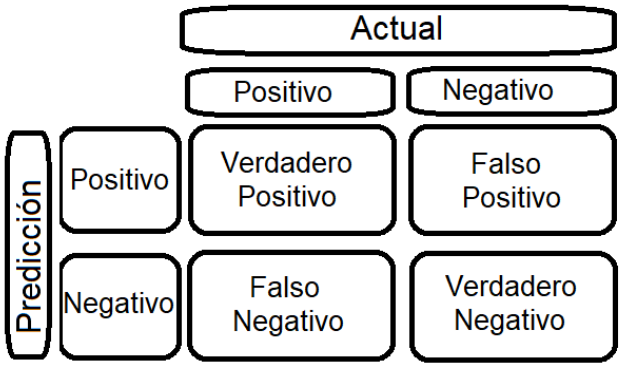


Fuente: Adaptado de (Kukil, 2022)

1.6.5 Matriz de confusión

En la detección de objetos, la matriz de confusión es una herramienta útil para analizar las predicciones hechas por un modelo de aprendizaje supervisado dentro de una imagen, para hacer su construcción es necesario contar con los valores de TP, TN Y FP, donde finalmente los valores generales por la predicción están definidos previamente por el umbral IoU.

Figura 1-16: Matriz de Confusión y sus valores



Fuente: Adaptada de (Shah, 2022)

CAPITULO 2: Diseño metodológico

En este capítulo se presenta la metodología que se llevó a cabo para lograr el cumplimiento de cada uno de los objetivos planteados.

2.1 Diseño

Para cumplir la detección de la plaga objetivo, la aplicación propuesta en este trabajo de grado hace uso de un modelo de Machine Learning con las librerías y códigos de Tensorflow Lite que son propiedad de (Brain, 2021). La aplicación hace la detección por medio de la cámara nativa del teléfono móvil y además presenta las características siguientes:

Toda la paquetería que se utilizó es de código libre y cualquier persona puede hacer uso de ellas sin necesidad de inversión monetaria.

- La aplicación puede ser utilizada para detectar cualquier objeto de interés que la persona desee. La tarea en este proyecto es detectar la plaga de Monalonion en aguacate hass, pero es posible pasarle otro tipo de datos etiquetados de interés y hacer el entrenamiento para que la ampliación haga detección de ellos.
- El entrenamiento del modelo de este proyecto se hizo mediante la técnica de Aprendizaje por Transferencia. La aplicación aquí propone un modelo ya entrenado, pero si se desea el usuario puede elegir en el repositorio de Tensorflow Lite entre 5 modelos diferentes con diferentes características entre precisión y velocidad que ahí describen.
- En cuanto a la movilidad, la aplicación no requiere de conexión a internet Wi-Fi o red de datos ya que una vez compilada la aplicación, esta tiene incorporado todos los paquetes y

metadatos necesarios para la tarea de detección, lo que permite que el usuario pueda hacer uso de ella en zonas aledañas sin depender de cualquier tipo de conectividad móvil.

- La instalación puede hacerse en cualquier dispositivo con sistema operativo Android con versión igual o superior a 9 y que cuente con cámara.
- La interfaz de usuario de la aplicación es caracterizada por su sencillez, cuenta con únicamente con una pantalla principal donde muestra un mensaje de descripción de su funcionamiento, tres imágenes de ejemplo de Aguacate hass y un botón en la parte inferior con el que el usuario ingresa a la cámara nativa del móvil.

2.1.1 Recopilación de herramientas

Para lograr obtener toda la información expuesta en este proyecto, se acudió a la búsqueda de documentos e investigaciones previa mediante recursos como Scielo y Google Académico donde se encontró además repositorios de diferentes universidades los cuales sirvieron para la aclaración de duda y recolecta de información vital para el desarrollo de esta investigación. Además, se acudió también a la página oficial de Tensorflow que es propiedad de Google Brain Team; en este sitio se encuentra documentación de cómo usar su biblioteca y los usos que se pueden llevar a cabo con ella.

2.1.2 Toma de fotografías

Para capturar las fotografías de los frutos de Aguacate hass se utilizó un dispositivo móvil de la marca Xiaomi con referencia Redmi Note 9 Pro, cuyas características principales se describen en el siguiente apartado.

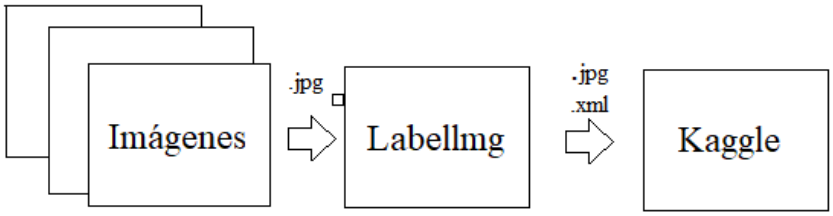
Las tomas se hicieron bajo condiciones de luz ambiente, fue un día en condición soleado y despejado, se hizo captura conservando un rango de distancia entre los 15 y 20 centímetros de distancia entre la cámara y el fruto en el árbol, se procuró que el objetivo este en posición vertical donde se evidencie la clase a detectar y en ningún momento haciendo uso del zoom para evitar distorsión en la imagen. En la **Figura 2-1** se puede evidenciar una captura hecha por el desarrollador de este proyecto.

Figura 2-1:Toma de fotografías de los frutos de Aguacate Hass

Fuente: Figura hecha por el autor

2.1.3 Proceso de etiquetado

Figura 2-2: Proceso de obtención del dataset en Kaggle



Fuente: Figura hecha por el autor

Input: Fotografías de los frutos.

Output: Archivos jpg y xml en formato PASCAL VOC alojados en la plataforma Kaggle.

El proceso de etiquetado de los datos de frutos de aguacate hass inicia con la apertura de las fotografías mediante el software de etiquetado, la correspondiente asignación de etiquetas (Sano, Monalonion y Enfermo) y la generación lo los archivos xml que contienen las coordenadas de la ubicación del fruto y la clase a la cual pertenece, luego, con todo el conjunto de datos obtenido se sube a la nube de la plataforma Kaggle donde se quedará para ser usado posteriormente en el proceso de entrenamiento.

Para construir el conjunto de datos se utilizó el software de código libre Labellmg, sus características están especificadas en el siguiente apartado. Para este paso es recomendable hacer reducción del tamaño de las imágenes con el fin de que en el entrenamiento se requiera menor costo computacional y un menor tamaño del conjunto de datos, esto se traduce en una reducción de tiempos y posibles pérdidas de características al poseer imágenes en grandes tamaños de pixeles. Existen diferentes herramientas para dicha redimensión, en este proyecto se utilizó Bulk Resize (*Bulk Resize Photos - Resize Images*, 2022), que se puede utilizar accediendo por medio

de un navegador. El conjunto de imágenes sirve como entrada al software para posteriormente quedar acompañadas de un archivo XML que contiene información de la ubicación del objeto (coordenadas), clase y el tamaño del mismo.

Cabe resaltar que, es indispensable hacer selección del formato PASCAL VOC, ya que es el compatible con las herramientas de TensorFLow Lite.

Figura 2-3: Estructura de archivo XML en formato PASCAL VOC.

```
<annotation>
        <folder>all</folder>
        <filename>IMG_20220315_144805.jpg</filename>
        <path>C:\Users\Ricardo\Desktop\A FINAL FINAL\all\IMG_20220315_144805.jpg</path>
        <source>
                <database>Unknown</database>
        </source>
        <size>
                <width>699</width>
                <height>935</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>Monalonion</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>214</xmin>
                        <ymin>385</ymin>
                        <xmax>424</xmax>
                        <ymax>627</ymax>
                </bndbox>
        </object>
</annotation>
```

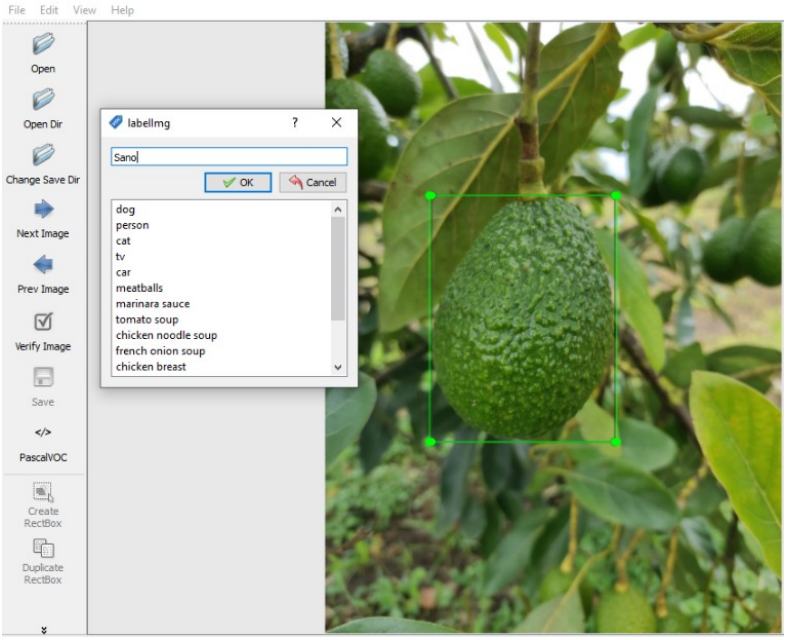
Fuente: Figura hecha por el autor

Respecto a las clases, como prioridad se hace detección de la clase “Monalonion” y “Sano”. La tercera clase “Enfermo” se agregó con el fin de hacer diferenciación de otras enfermedades que pueden estar presentes. Se etiquetaron con la clase mencionada para así poder diferenciar los aguacates hass sanos y con presencia de la plaga objetivo a detectar.

Como se observa en la **Figura 2-4,** la interfaz contiene una barra de herramientas donde permite el acceso a la carpeta contenedor de las imágenes a etiquetar y guardado de las mismas,

también permite la selección del formato de salida deseado y la definición de las clases deseadas por el usuario.

Figura 2-4: Etiquetado el Labellmg



Fuente: Figura hecha por el autor

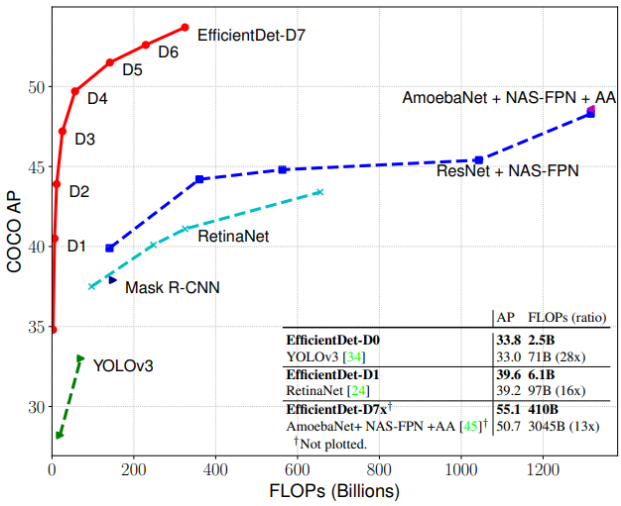
Con el archivo que se genera en los anteriores pasos, se obtiene cada fotografía con su respectivo archivo XML ambos en una misma carpeta donde se hizo la división normalmente empleada 80-20% según la literatura de (Amazon, 2022), perteneciéndole el 80 % a los datos de entrenamiento y el restante a los datos de evaluación procurando distribuir las clases en proporciones equilibradas; esta división es comúnmente utilizada con el fin de tener una pequeña parte de los datos para ir evaluando el modelo y saber en qué momento del entrenamiento se consigue una precisión aceptable. La carpeta con dicha división se guardó en un archivo comprimido en formato .zip con el nombre de “Dataset_Aguacate_Hass” y se sube a la nube,

para ello se empleó la plataforma Kaggle que da acceso gratuito para almacenar Dataset y posteriormente poder importarlos en la plataforma de entrenamiento.

2.1.4 Definición del modelo

En la **Figura 2-5** se observa la precisión superior vs parámetros en comparación con otras redes, se decide optar por la familia EfficientDet ya que obtiene una precisión de 83% con tan solo 40 millones de parámetros en comparación del modelo más próximo AmobaNet-A que logra la misma precisión, pero con 90 millones de parámetros, de esta manera el modelo escogido requiere de menos tiempo de entrenamiento y como resultado una mayor precisión.

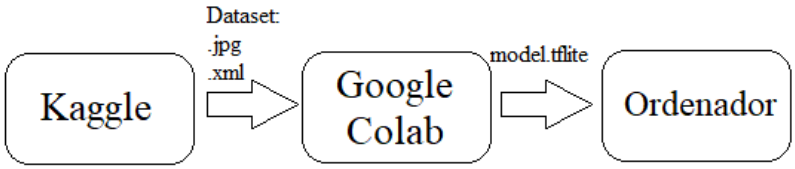
Figura 2-5: Precisión en comparación a cantidad de parámetros



Fuente: Adatado de (Google Brain Team, 2020)

2.1.5 Entrenamiento y ajuste del modelo TensorFlow Lite

Figura 2-6: Secuencia de pasos para el entrenamiento



Fuente: Figura hecha por el autor

Input: Dataset con archivos jpg y xml desde la plataforma Kaggle.

Output: Modelo entrenado en la plataforma de Google Colab en formato tflite.

Esta actividad corresponde al entrenamiento que se realizó en el entorno de Google Colaboratory (Google, 2022). Aquí mediante la ejecución de los comandos en Python, se importa el conjunto de datos generado previamente alojado en Kaggle el cual contiene los archivos requeridos para que el modelo de TensorFlow Lite aprenda las características asignadas, una vez se completa el proceso de entrenamiento se procede a guardar el archivo (model.tflite) en el almacenamiento local del ordenador para luego poder ser usado en la aplicación móvil.

Para llevar a cabo las actividades generales en Colab, se procede creando un cuaderno en donde se alojan todos los códigos para importar las herramientas y librerías indispensables para la tarea de detección de objetos, aquí se llamó al cuaderno “ENTRENAMIENTO FINAL.ipynb” y se lo puede encontrar en el anexo de códigos fuente de este proyecto de grado.

El entrenamiento del modelo inicia con activación de la aceleración por hardware desde la barra de herramientas de Colab, esto se hace con el objetivo de reducir los tiempos de entrenamiento. Cabe resaltar que la plataforma brinda de manera gratuita una GPU Tesla-T4 con 16 Gibabytes de Ram y un almacenamiento de 80 Gb.

Las librerías que se emplearon están descritas en la **Tabla 2-1**, allí se hace una breve descripción sobre la tarea que las caracteriza.

Tabla 2-1: Librerías empleadas para desarrollar el modelo

LENGUAJE DE PROGRAMACIÓN	LIBRERÍA	DESCRIPCIÓN
Python	Tensorflow	Para aprendizaje automático, desarrollada por Google Brain Team para la creación y entrenamiento de modelos redes neuronales convolucionales.
Python	Tflite-model-maker	Creación de un modelo con conjunto de datos personalizado, utiliza la técnica de aprendizaje por transferencia.
Python	pycocotools	Paquete de datos oficial de Microsoft COCO.
Python	Numpy	Manejo de vectores y matrices por medio de matemáticas avanzadas.
Python	Os	Manera versátil de usar funcionalidades dependientes del sistema operativo.

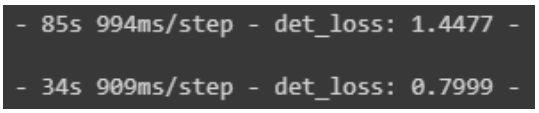
Fuente: Tabla hecha por el autor

Una vez que se tiene los paquetes mencionados en la **Tabla 2-1**, de forma secuencial, el siguiente paso consiste en la importación del Dataset que previamente se había subido a la nube de Kaggle, se importa el modelo EfficientDet-Lite que se encarga de recibir todo el conjunto de metadatos provenientes de las convoluciones y por último y muy importante la configuración de las iteraciones que se llevan a cabo en el entrenamiento y el tamaño del lote.

Para saber si se estaba haciendo un correcto entrenamiento, se utilizó la técnica de detección temprana, que consiste en observar de acuerdo al número de iteraciones la mejora de la precisión. Además, recordar que si se deja en aumento demasiadas iteraciones el modelo va a presentar lo que se conoce como sobreajuste (Overfitting) y cuando se lo compare con los datos de entrenamiento su certeza será muy alta o muy baja en contraste con los datos de evaluación, en pocas palabras se puede decir que no se está generalizando la detección. Con múltiples pruebas se evidenciaba que luego de 1000 iteraciones las pérdidas (loss) iban disminuyendo, lo

que indica que el modelo a medida que transcurre las épocas va mejorando su precisión, con esto aclarado se procedió a establecer los hiperparámetros, dejando las iteraciones (epochs) en 1000, que hace referencia a la cantidad de veces que se pasa por el conjunto de datos de entrenamiento y un tamaño de lote (batch size) de 16, referente a la cantidad de datos por iteración. Durante ese proceso, el entorno muestra valores de pérdida de datos (loss), que al empezar el entrenamiento el valor es bastante alto, pero a medida que avanzan las iteraciones este valor va disminuyendo lo cual se hace bueno, si este valor no decrementa quiere decir que puede haber un inconveniente con las imágenes del dataset o si por el contrario este valor aumenta de manera rápida es evidencia de que se está dando sobreajuste.

Figura 2-7:Valores de loss durante el entrenamiento en Colab

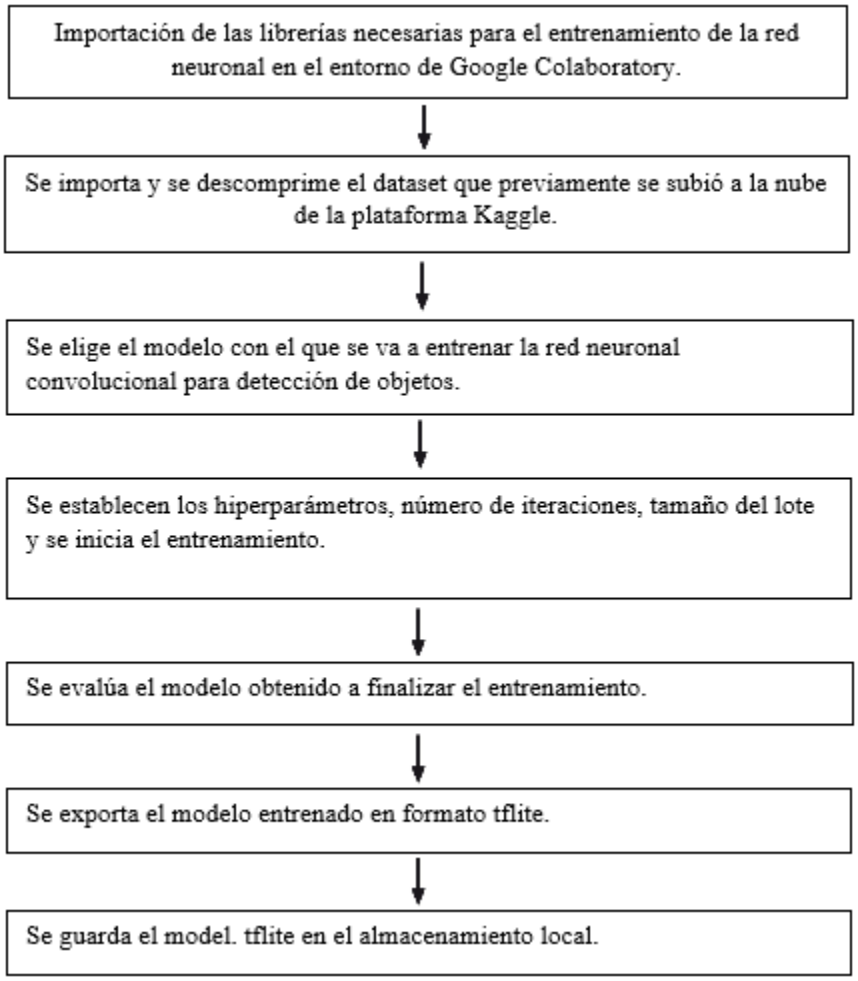


Fuente: Figura hecha por el autor

Una vez que termina el entrenamiento, se evalúa el modelo obtenido para poder observar si se cumple con una precisión aceptable que está dado entre 0 y 1 referente al 100 % de precisión y se guarda el archivo en el almacenamiento local del ordenador.

La **Figura 2-8** muestra la serie de pasos de manera secuencial en el entorno de Colab.

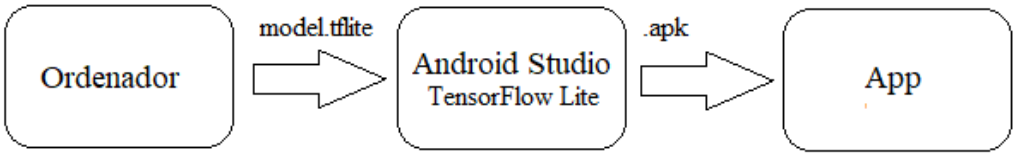
Figura 2-8:Secuencia de entrenamiento en Google Colaboratory



Fuente: Figura hecha por el autor

2.1.6 Desarrollo de la aplicación Android móvil

Figura 2-9: Incorporación del modelo en la aplicación Android móvil



Fuente: Figura hecha por el autor

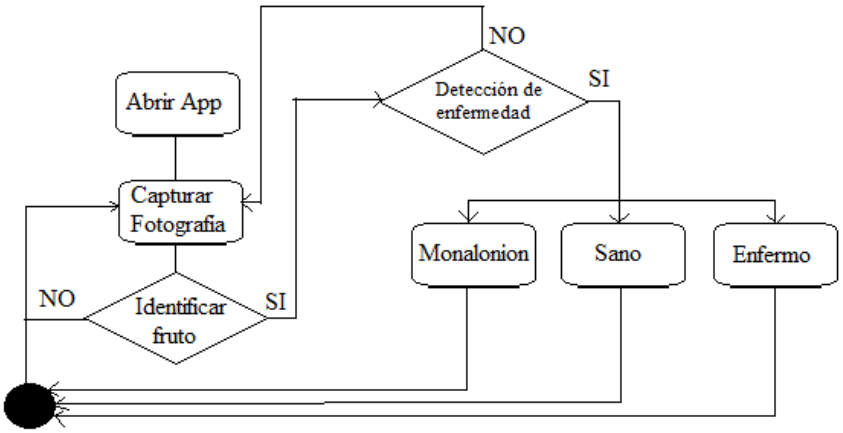
Input: Librerías TensorFlow Lite, model.tflite.

Output: App detección de objetos.

Uno de las etapas importantes de la aplicación es la incorporación del modelo entrenado mediante Google Colaboratory, como se mencionó inicialmente, la aplicación hace uso de la biblioteca de tareas de TensorFlow Lite, que contiene un conjunto de bibliotecas que permitió al desarrollador la creación de esta experiencia de detección de objetos, se enlaza el modelo por medio de la herramienta Android Studio, alojándolo en una carpeta dedicada para este archivo (Assets), dentro de la herramienta también se verifica que no falte ninguna de las librerías necesarias para la app, así como también la definición y alienación correcta de los botones y demás contenedores que harán parte de la interfaz de usuario con que se da manejo a la app para luego mediante la opción “Build” de Android Studio hacer compilación de toda la paquetería y así obtener un archivo final en formato .apk instalable en el teléfono móvil Android.

La tarea de detección del fruto se cumple con los siguientes pasos:

Figura 2-10: Diagrama de flujo que describe la tarea de detección



Fuente: Figura hecha por el autor

Al ejecutar la aplicación se hace un llamado hacia la etapa de detección de objetos, donde se llama al método de detección de objetos (*private fun runObjectDetection(bitmap: Bitmap))* y queda a la espera del ingreso de una imagen capturada por la cámara del teléfono móvil. Al realizarse una captura, se hacen ciertas transformaciones, entre ellas convertir la imagen a RBG, crear una imagen de tensor que contiene información de la ubicación del objeto interés para la ubicación del cuadro delimitador o BoundingBox.

Una vez se retorna un objeto detectado, se desprecia los que no tienen un puntaje de confianza sobre el umbral establecido, para este proyecto se lo estableció sobre un 80 %, pero puede ser cambiado por el usuario en la opción (setScoreThreshold(0.8f)), igualmente para la cantidad de detecciones por captura que se estableció en 2 (.setMaxResults(2)).

Figura 2-11: Actividad de detección de objeto y cuadro delimitador

Fuente: Figura hecha por el autor

2.1.7 Acceso a la cámara

El permiso para el acceso ya se encuentra otorgado en el código de la aplicación, en esta interfaz en la parte inferior central está ubicado el botón “Acceder a cámara”, al ingresar, la app hace un llamado a la cámara nativa del teléfono móvil y es mostrada para hacer la captura.

Figura 2-12: Botón de acceso a la cámara nativa del teléfono móvil

Fuente: Figura hecha por el autor

2.2 Herramientas de desarrollo

En este apartado se mencionan las diferentes herramientas que se utilizaron para el desarrollo de este proyecto.

2.2.1 Tensorflow

Es una novedosa biblioteca open Source creada por Google, esta librería se basa en gráficos computacionales, posee una gran capacidad para calcular de manera muy veloz los gradientes ya que ofrece herramientas de excelente potencia para cumplir sus tareas, además de brindar buenas herramientas para visualizar resultados. Tensorflow cuenta con una interfaz de programación de bajo nivel en donde ofrece un control muy interactivo con el usuario para que se lleve a cabo el desarrollo de redes neuronales (Goldsborough, 2018).

Esta herramienta se seleccionó teniendo en cuenta los aspectos siguientes

- Primeramente, fue la que arrojó un modelo el cual funcionó perfectamente al contar con los metadatos incorporados a diferencia de Yolo v5 que generó un modelo, pero no compatible para la aplicación de este proyecto.
- Está integrado para dispositivos móviles
- Es liviano, permitiendo instalación en móviles gama media.
- Permite utilizar la aceleración de hardware si se tiene un móvil con características informáticas robustas.
- Contiene bibliotecas open source como TensorFlow Model Maker que genera modelos con metadatos incorporados, lo cual es requisito para la aplicación de este proyecto.

2.2.2 Tensorflow Lite

Conjunto de herramientas de código libre que permite la creación de tareas como detección de objetos, da la posibilidad a los desarrolladores de incluir modelos en dispositivos móviles o IoT.

Unas de sus limitaciones sobresalientes es que posee un tamaño reducido, con el cual los costos computacionales no serán elevados y no necesita conectividad a internet para operar, además es compatible con el lenguaje de programación Python (Google Developers, 2020).

2.2.3 Android Studio

Es un IDE oficial para el desarrollo de aplicaciones Android, de código libre y se encuentra basado en IntelliJ IDEA. Cuenta con un editor de códigos muy robusto y excelentes herramientas para los desarrolladores (Brain Team, 2022).

De acuerdo a este proyecto se tuvo en cuenta los siguientes criterios para su selección:

- Tiene incluidas herramientas para personalizar la interfaz que este proyecto requirió.
- Es una herramienta compatible con todas librerías que se emplearon. Además, permite personalizar interfaces gráficas en el mismo entorno y no hubo necesidad de acudir a otro.
- Fue posible usar un emulador dentro de su entorno para ir observando el estado del proyecto.
- Entre sus herramientas incluye la solución de errores, lo cual fue útil cuando se presentaban inconvenientes durante el desarrollo.

2.2.4 LabelImg

Es una herramienta de notación de imágenes de manera gráfica, el cual está escrito en Python y utiliza Qt para su interfaz gráfica, este software guarda las anotaciones en un archivo. XML y un formato de salida PascalVOC, el formato utilizado por imgNet (TzuTa Lin, 2021).

Esta herramienta es esencial cuando se trata de proyectos de machine learning o Deep learning ya que para entrenar un modelo es necesario ingresarle las clases que debe aprender y posteriormente predecir.

Criterios para seleccionar el software de etiquetado (LabelImg):

- Acepta imágenes de mínimo 8x8 pixeles y máximo 1024x1024 pixeles, permitiendo un amplio rango de calidades en las fotografías que puedan ser tomadas por los diferentes móviles existentes en el mercado actual como el teléfono que se utilizó.
- Tiene compatibilidad con el formato de salida PascalVOC que requiere TensorFlow
- Tiene un costo computacional reducido, funciona en el teléfono empleado.
- Aplicación ejecutable en Windows, el OS que posee el computador empleado.
- Software gratuito.

2.2.5 Kaggle

Es una plataforma de código abierto la cual cuenta con una interfaz Jupyter Notebooks que puede ser personalizada y no requiere configuración. Esta herramienta aloja una gran comunidad (más de 536 mil usuarios provenientes de 194 países diferentes) y por ende reúne más de 15 mil publicaciones cada mes, permite almacenar Dataset de manera gratuita en donde el miembro decide si lo hace de manera pública, para la comunidad de la plataforma o de manera privada, además, ofrece el servicio de préstamo de GPU sin coste alguno, empresas prestigiosas como Walmart o Facebook hacen uso de esta ya que es ideal para tareas de Data Science (DataScientest, 2021).

Criterios para escoger la plataforma Kaggle:

- Compatibilidad con la plataforma Google Colaboratory.
- Ofrece Compatibilidad con el lenguaje de programación Python para importar mediante comandos el dataset allí alojado.

- Permite almacenar datos en la nube de manera gratuita, fue útil porque ofrece 107 Gigabytes donde se pudo alojar el conjunto de datos de este proyecto y poderlo incorporar en Colab.

2.2.6 Google Colaboratory

También conocido como Colab, es un recurso informático virtual, donde se accede por medio de un navegador Web (Chrome, Firefox, Safari, Opera, Etc.) y una conexión a internet. Esta plataforma es de gran ayuda en los proyectos que requieren gran costo computacional debido a que brinda a cualquier usuario GPU de manera gratuita, a fin de, apoyar la educación y la investigación en temas de Inteligencia Artificial, Data Science y Visión Artificial (Google, 2022).

Es importante precisar, que esta plataforma ofrece compatibilidad con diferentes plataformas, un ejemplo de ellas es Kaggle que fue imprescindible para este proyecto, también, permite el acceso a librerías como Tensorflow Lite. Además, en este proyecto fue de gran ayuda ya que se requería del uso de GPU para el entrenamiento del modelo.

Esta plataforma se escogió de acuerdo a los siguientes criterios:

- Compatibilidad con la plataforma Kaggle donde se tiene el dataset.
- Ofrece Compatibilidad con el lenguaje de programación Python que el lenguaje de las librerías que se utilizaron.
- Ofrece el servicio de préstamo gratuito de GPU, a fin de, optimizar los tiempos de entrenamiento, debido al elevado costo computacional que genera este dataset de imágenes, para entrenar un modelo en específico.
- Ofrece compatibilidad con la librería TensorFLow Lite.

- Ofrece almacenar datos en la nube de Google directamente, aquí fue posible ir guardando los avances que se hacían en el día a día.

2.2.7 Hewlett Packard dv6, Intel(R) Core (TM) i5-2450M CPU @ 2.50GHz 2.50 GHz, 6GB RAM

El uso de este equipo es con el fin de realizar las tareas en las plataformas mencionadas anteriormente, etiquetar y aplicar los parámetros ideales al dataset consolidado por el desarrollador.

Entre sus principales características encontramos:

- Arquitectura = Genuine Intel.
- Cadena de Arquitectura = Intel(R) Core (TM) i5-2450M CPU @ 2.50GHz.
- ID de Arquitectura = Intel64 Family 6 Model 42 Stepping 7.
- Número de Procesadores = 1.
- MHz = 2494.
- Tipo de Arquitectura = AMD64.
- Bits de Arquitectura = 64

CAPITULO 3: Resultados y análisis de resultados

Con los resultados que se obtuvieron en este proyecto de fin de grado, se pueden evaluar los objetivos planteados, entre ellos el objetivo general que es la detección de Aguacates Hass con la plaga Monalonion mediante Visión Artificial y Deep Learning.

4.1 Modelo de detección de objetos para dispositivos móviles

La obtención del algoritmo para la detección de la plaga Monalonion se logró obtener haciendo prueba de los siguientes recursos:

4.1.1 *Prueba de modelo obtenido en YoloV5*

En un inicio, para llevar a cabo la detección de la plaga objetivo, se utilizó el algoritmo de Yolo V5, propiedad de (Jocher & ultralytics, 2021), el cual permite el entrenamiento de modelos dedicados a la detección de objetos e incorporarlos a dispositivos móviles. A partir de esto y que en este trabajo de grado se desarrolló una aplicación android mediante las librerías de TensorFlow Lite (Google Developers, 2020) para detectar la plaga Monalonion en Aguacate hass, se hizo la primera prueba de incorporar un modelo siguiendo los criterios de entrenamiento según (Jocher & ultralytics, 2021), y posteriormente se enlazó el modelo que se generó, pero la aplicación iniciaba hasta su pantalla principal y al momento de ejecutar la cámara esta se cerraba de inmediato y el ciclo se repetía al intentar nuevamente. Para dar solución a este error, se etiquetó nuevamente todo el conjunto de fotografías para comprobar si se debía a un error de las etiquetas. Sin embargo, se evidenció que esa no era la causa del problema, indagando en la comunidad de desarrolladores de Stack Overflow (Overflow, 2021) se dio por enterado de que el

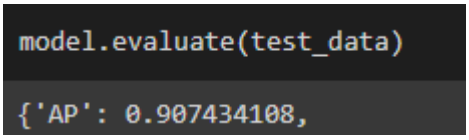
modelo que arrojaba Yolo V5 tenía por separado los nombres de las clases (clases.txt) con respecto a los demás metadatos dedicados a la detección.

4.1.2 Algoritmo de detección con TensorFlow Lite

Gracias a que Tensorflow Lite tiene gran popularidad ante la comunidad de desarrolladores se puede encontrar información relevante en diferentes plataformas virtuales como Tensorflow.org, You Tube, Stack Overflow, etc. Se pudo encontrar el algoritmo de entrenamiento para generar un modelo adecuado para la aplicación de este proyecto. En un primer entrenamiento y enlace del modelo obtenido en la app, está ya ejecutaba la cámara nativa del teléfono móvil y procedía a hacer la detección de la plaga gracias a que ahora se contaba con todos los metadatos incorporados en un solo paquete. No obstante, se notó que se confundía al diferenciar las tres clases y había casos en que no detectaba el objeto, esto se debía a que era conveniente aumentar las iteraciones a una cantidad donde el algoritmo aprendiera más características sobre los frutos y la precisión mAP aumentara ya que el primer modelo presentó una precisión inferior al 50 %.

Es importante recordar que el valor de la métrica mAP va desde 0 a 1 como se habló en la sección 1.8. En este último entrenamiento se obtuvo un valor de 0.9074 que es equivalente a 90% con un parámetro de 1000 iteraciones, lo cual lo hace ser un modelo aceptable.

Figura 3-1:Resultado de precisión



Fuente: Figura hecha por el autor

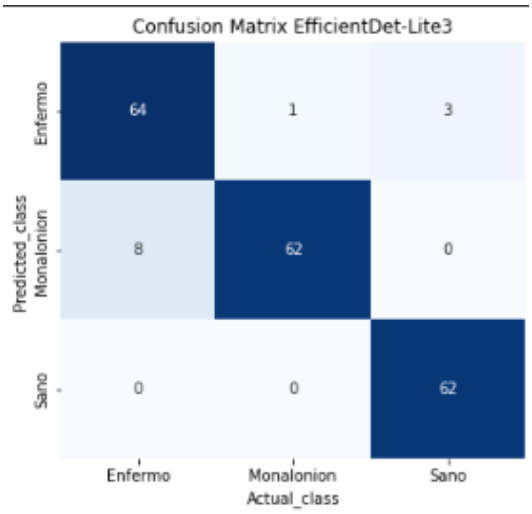
El modelo que finalmente se obtuvo tiene un formato tflite con un peso de 11.456 Kilobytes, tiene incorporado metadatos entre los cuales las clases a detectar que era necesario para la funcionalidad en la aplicación, se generó transcurridas 11 horas de entrenamiento continuo, se lo dejó desde las 8 P.m. y al día siguiente a las 7 .30 A.m. ya se encontraba listo para su descarga.

4.1.3 Matriz de Confusión

Con el fin de continuar con el proceso de evaluación del algoritmo de este proyecto, se presenta la matriz de confusión que permitió observar de forma explícita que tan bueno fue el modelo, nos permite realizar los cálculos matemáticos para saber la precisión de cada clase.

Recordar que una buena precisión del modelo depende de la adaptabilidad con respecto a los datos de entrenamiento (train), y una buena generalización se relaciona con la respuesta que tiene con los datos de evaluación (test), que son datos que la red neuronal convolucional no conoce o nunca ha visto.

Figura 3-2: Matriz de confusión



Fuente: Figura hecha por el autor

La **Tabla 3-1** muestra los valores de predicción.

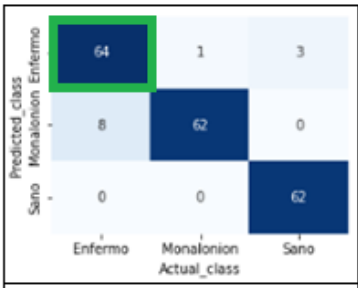
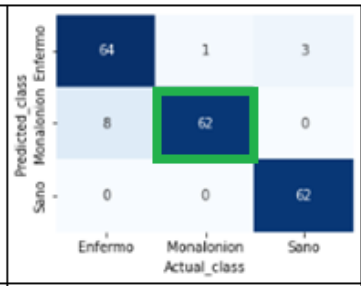
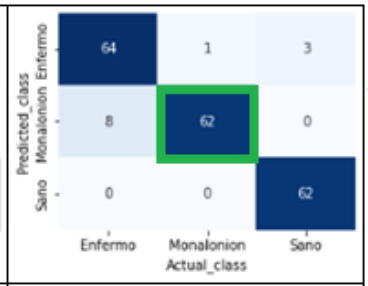
Tabla 3-1: Tabla de confusión

CLASES	VALOR DE PREDICCIONES		
	TP	FP	FN
Enfermo	64	8	4
Monalonion	62	1	8
Sano	62	3	0

Fuente: Tabla hecha por el autor

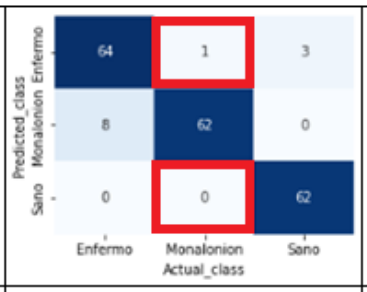
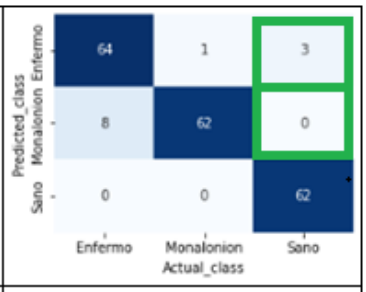
Los valores de TP, FP y TN se obtienen siguiendo las indicaciones de cada tabla a continuación:

Tabla 3-2: Obtención de los TP

		
▪ **Enfermo** Corresponde al valor de su diagonal.	▪ **Monalonion** Corresponde al valor de su diagonal.	▪ **Sano** Corresponde al valor de su diagonal.

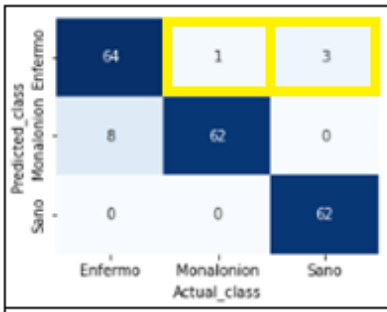
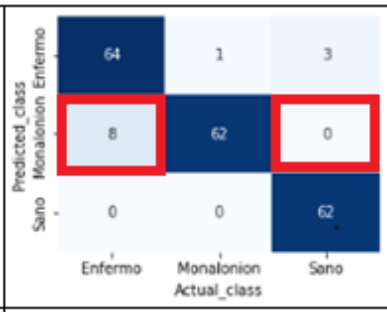
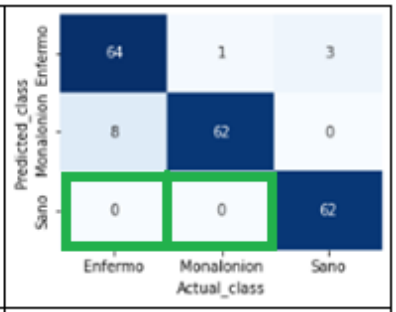
Fuente: Tabla hecha por el autor

Tabla 3-3: Obtención de los FP

		
▪ **Enfermo** Se obtiene sumando los valores con recuadro amarillo.	▪ **Monalonion** Se obtiene sumando los valores con recuadro rojo.	▪ **Sano** Se obtiene sumando los valores con recuadro verde.

Fuente: Tabla hecha por el autor

Tabla 3-4: Obtención de los FN

		
▪ **Enfermo** Se obtiene sumando los valores con recuadro amarillo.	▪ **Monalonion** Se obtiene sumando los valores con recuadro amarillo.	▪ **Sano** Se obtiene sumando los valores con recuadro amarillo.

Fuente: Tabla hecha por el autor

La matriz de confusión es una herramienta que permite demostrar de forma explícita el rendimiento de un modelo, en este caso de detección de objetos. Permite observar el número de detecciones correctas y erróneas que hace el algoritmo. Además, por medio de las ecuaciones mencionadas anteriormente, es posible calcular la precisión y recall.

- **Precisión para las tres clases** La precisión cuantifica la autenticidad de todos los objetos positivos que se detectaron dentro de la imagen

$$\text{Precisión(Enfermo)} = \frac{\text{TP}}{\text{TP+FP}} * 100 = \frac{64}{64+8} = 88$$

La precisión para la clase “Enfermo” equivale a 88 %.

$$\text{Precisión(Monalonion)} = \frac{62}{62+1} * 100 = 98$$

La precisión para la clase “Monalonion” equivale a 98 %

$$\text{Precisión(Sano)} = \frac{62}{62+3} * 100 = 95$$

La precisión para la clase “Sano” equivale a 95 %

- **Recall para las tres** El recall mide el porcentaje de las nuevas detecciones positivas en comparación con las que existen en el entrenamiento, es decir, compara los nuevos objetos con lo que el modelo recuerda y arroja un porcentaje.

$$\text{Recall(Enfermo)} = \frac{\text{TP}}{\text{TP} + \text{FN}} * 100 = \frac{64}{64 + 4} = 94$$

El Recall para la clase “Enfermo” equivale al 94 %

$$\text{Recall(Monalonion)} = \frac{62}{62 + 8} * 100 = 88$$

El Recall para la clase “Monalonion” equivale al 88 %

$$\text{Recall(Sano)} = \frac{62}{62 + 0} * 100 = 100$$

El Recall para la clase “Sano” equivale al 100 %.

De acuerdo a los resultados mostrados por la matriz de confusión, se puede decir que la precisión para cada clase mostrada en la **Tabla 3-6** hace que el modelo sea aceptable.

Tabla 3-5: Estadísticas de predicciones

CLASE	Precisión	Recall
Enfermo	88 %	94 %
Monalonion	98 %	88 %
Sano	95 %	100 %

Fuente: Tabla hecha por el autor

4.2 Conjunto de datos consolidado

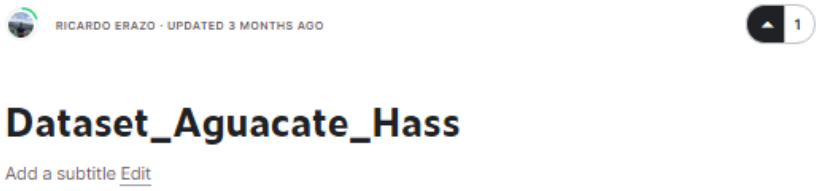
Para recolectar las fotografías se hizo vista a la finca “Las Palmas” ubicada en el municipio de San Agustín, Huila, donde se logró recolectar alrededor de 1700 fotografías en formato .jpg, de las cuales se hizo selección de 1000 imágenes que se observaron adecuadas para un correcto entrenamiento, las fotografías fueron tomadas por un celular de marca y referencia Xiaomi redmi Note 9 Pro el cual tiene las especificaciones siguientes: Cámara principal de 64 Megapíxeles, apertura focal f/2.4, tamaño del sensor 1/1.72, sensor de profundidad de 2 Megapíxeles y un campo de visión de 79,8° (Xiaomi, 2020).

Se seleccionó las imágenes procurando que se observe el fruto con claridad, un buen enfoque y buena determinación de las características, cada fotografía contiene la representación de un solo fruto de aguacate hass, se dividió el conjunto en datos de prueba y entrenamiento, perteneciendo 200 imágenes al conjunto de evaluación en las cuales están 86 de aguacates sanos, 57 con Monalonion y 57 con otras plagas, en cuanto al conjunto de entrenamiento le pertenecen

344 fotografías de aguacates sanos, 228 con Monalonion y 228 con otra plaga. De esta manera se generan tres diferentes clases (Monalonion, Enfermo y Sano).

Finalmente, el Dataset con el que se cumplió el objetivo de este trabajo de grado se caracteriza por contener 1000 fotografías de frutos de aguacate hass con resolución inferior a 1080 pixeles tomadas bajo condiciones naturales del día. EL conjunto de datos contiene un total de 430 tomas de frutos sanos, 285 de frutos con la plaga Monalonion y 285 tomas de distintas enfermedades que se capturaron en la finca “Las Palmas”. Cada fotografía se encuentra en formato jpg el cual es indispensable para los pasos correspondientes de este proyecto, archivos xml en formato PASCAL VOC, que contienen la información como coordenadas, la clase a la que se asignó en el proceso de etiquetado y el nombre de la carpeta contenedora. Este dataset se encuentra disponible en la plataforma Kaggle en un archivo comprimido en formato .zip con el nombre “Dataset_Aguacate_Hass”, en la cuenta del autor (Erazo, 2022).

Figura 3-3: Dataset en Kaggle



Fuente: Figura hecha por el autor

4.3 Interfaz gráfica de la aplicación móvil

La personalización de la interfaz de usuario para la aplicación móvil se realizó por medio de las herramientas de Android Studio, las librerías creadas por Tensorflow Lite contienen una serie de códigos que permitieron al desarrollador de este proyecto manejar a gusto la

presentación deseada, asignar colores y textos descriptivos, así como el personalizado de un botón principal muy importante que da acceso a la cámara. Se alojaron tres fotografías de aguacate hass que el usuario puede seleccionar al toque sobre alguna de ellas y la aplicación ejecuta la tarea de detección de una de las tres clases establecidas.

4.3.1 Pantalla Inicial

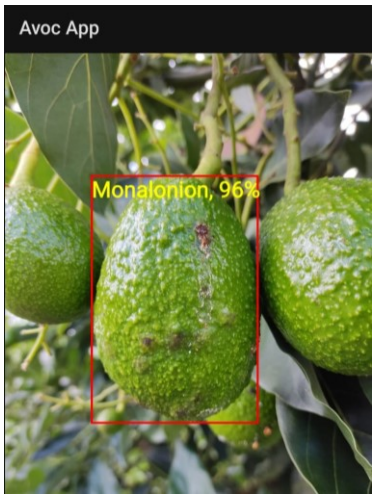
Esta pantalla inicial es la mostrada al usuario una vez que se ingrese a la aplicación "Avoc app", un contenedor importante muestra un texto con una breve descripción a cerca de la funcionalidad de la app y sugiriendo una distancia de enfoque donde se puede obtener una mejor fotografía para ser procesada, además de contener tres imágenes de ejemplo con las clases a detectar, donde el usuario puede seleccionar cualquiera de ellas y observar un resultado.

Figura 3-4: Ventana principal de la interfaz de usuario

Fuente: Figura hecha por el autor

Una vez se selecciona una de las fotografías de ejemplo o se realiza una captura con la cámara del móvil, la aplicación realiza el procesamiento de detección y muestra un resultado que obtuvo como el de la **Figura 3-5**

Figura 3-5: Detección de la plaga Monalonion en fruto de aguacate hass



Fuente: Figura hecha por el autor

4.4 Aplicación móvil funcional para la detección de frutos de aguacate hass con Monalonion

El desarrollo de la aplicación móvil funcional para la detección de frutos de aguacate hass con Monalonion también conlleva una serie de pasos con los que es importante resaltar el uso de las paqueterías y APIs propiedad de Google. Aquí se encuentra la herramienta de TensorFlow Lite que permite la incorporación de modelos de detección de objetos en un dispositivo móvil (Google Developers, 2020).

4.4.1 Paquetes TensorFlow Lite

Los repositorios de Tensorflow Lite se obtuvieron mediante los comandos ejecutados secuencialmente en el cuaderno de Google Colab, estas librerías permitieron la creación de un modelo en formato tflite para detección de objetos con el conjunto de datos de este proyecto de grado para luego poder ejecutarlo como aplicativo móvil. Es necesario contar con la herramienta Android Studio para poder hacer las modificaciones. La aplicación contiene una serie de actividades creadas por Tensorflow Lite que se encargan de hacer enlace con el modelo entrenado e invocar las funciones de detección una vez se realizó una captura.

Esta herramienta es compatible desde la versión de Android 9.0 o superior. El paquete de instalación final tiene un tamaño de 42,949 Megabytes y puede ser instalado sin necesidad de conexión a Wi-Fi o datos móviles y se tendrá una app como la representada en la **Figura 3-6**

Figura 3-6: Aplicación instalada en dispositivo móvil Android

Fuente: Figura hecha por el autor

4.4.2 Requisitos

La ejecución del aplicativo móvil para la identificación de frutos de aguacate hass con Monalonion tiene como característica funcional la verificación de los requerimientos de software para poder ejecutarse y realizar las actividades por la que se la entrenó anteriormente. Para un

correcto funcionamiento la app hace recopilación de los siguientes requisitos Funcionales y no funcionales mostrados en la **Tabla 3-8**

Tabla 3-6:Requerimientos funcionales y no funcionales

FUNCIONALES	NO FUNCIONALES
Permiso para acceder a la cámara nativa del móvil.	Compatibilidad con versión Android 9 o superior.
Hacer detección de las clases en el fruto de aguacate hass.	Latencia.

Fuente: Tabla hecha por el autor

Funcionales

- **Permiso para acceder a la cámara nativa del móvil** Es indispensable el acceso a la cámara del dispositivo móvil android, este ya se encuentra otorgado dentro de las actividades de la aplicación.
- **Hacer detección de las clases en el fruto de aguacate hass** El algoritmo debe hacer detección del fruto y señalar por medio de un recuadro una de las tres clases según corresponda.

No Funcionales

- **Compatibilidad con versión Android 9 o superior** La aplicación móvil es compatible y ejecuta todas sus actividades correctamente desde l versión de Android 9 o superior.
- **Latencia** El tiempo que tarda el modelo en dar respuesta a una solicitud de detección es de 116 milisegundos.

CAPITULO 4: Conclusiones

La aplicación Android móvil desarrollada en este proyecto permite realizar la detección de la plaga Monalonion frente a otras dos clases (sano y enfermo).

El diseño del algoritmo de detección de la plaga Monalonion se obtuvo a partir de un modelo de arquitectura Efficiendet-Lite3, con una latencia de 116 milisegundos, un peso aproximado de hasta 12 Megabytes y una precisión promedio mAP de 90%. El uso de la plataforma Google Colaboratory jugó un papel importante en la disminución de costos computacionales.

Se consolidó un conjunto de datos que contiene 1000 fotografías de aguacates hass en tres categorías, sanos, con la plaga y otra clase de enfermedades para lograr diferenciarlo, bajo condiciones reales del día, permitiendo calibrar la máquina de aprendizaje.

Con la utilización de la herramienta Android Studio se personalizó una única interface gráfica amigable, que muestra toda la información necesaria para los usuarios: (nombre de la app, descripción de funcionamiento, imágenes captadas, imágenes de referencia, toma de nueva imagen).

En la aplicación desarrollada se logró integrar un algoritmo de detección de imágenes con técnicas de Deep Learning y Visión Artificial en un dispositivo móvil con versión de Android 9 o superior sin conexión a internet con una latencia de 116 milisegundos, características suficientes para su uso en cultivos de este tipo de aguacate.

5.1 Recomendaciones

Se recomienda usar fotografías no mayores a 1080 píxeles, ya que si se excede dicha recomendación el algoritmo no va a tomar todas las características de la imagen.

Se recomienda usar imágenes en formato JPG o JPEG, sino habrá errores durante el entrenamiento en Google Colab.

También se recomienda generar el archivo de etiquetas XML en formato PascalVOC, para que TensorFlow Lite tome correctamente las clases.

Referencias Bibliográficas

Amazon, aws. (2022). *Capa gratuita de AWS | Cloud computing gratis |AWS*. https://aws.amazon.com/es/free/?trk=eb709b95-5dcd-4cf8-8929-6f13b8f2781f&sc_channel=ps&s_kwcid=AL!4422!3!618394789840!e!!g!!amazon%20web%20services&ef_id=CjwKCAjwkaSaBhA4EiwALBgQaHqT-yY5uCwnsKsN_byiG-f4I_9-lrxLHuQVYk6knRJnZjN8b2u6MBoC8IQQAvD_BwE:G:s&s_kwcid=AL!4422!3!618394789840!e!!g!!amazon%20web%20services&all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all

Apache, Google, & Tensorflow. (2020). *Search | TensorFlow Hub*. https://tfhub.dev/s?network-architecture=efficientdet

BBVA. (2021). *Qué es el «machine learning» y cómo funciona*. https://www.bbva.com/es/machine-learning-que-es-y-como-funciona/

Berger, J., Preussler, C., & Agostini, J. P. (2019). *Identificación de síntomas de Huanglongbing en hojas de cítricos mediante técnicas de deep learning*. http://sedici.unlp.edu.ar/bitstream/handle/10915/71000/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y

Brain, G. (2021). *Detección de objetos con Android | TensorFlow Lite*. https://www.tensorflow.org/lite/android/tutorials/object_detection

Brain Team, G. (2022). *Introducción a Android Studio | Desarrolladores de Android | Android Developers*. https://developer.android.com/studio/intro?hl=es-419

Bulk Resize Photos - Resize Images. (2022). https://bulkresizephotos.com/en

C Arlen, T. (2018). *Understanding the mAP Evaluation Metric for Object Detection | by Timothy C Arlen | Medium*. https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3

Calderon, A., & Cortes, H. D. H. (2020). Machine learning en la detección de enfermedades en plantas: Machine learning en la detección de enfermedades en plantas. *Tecnología Investigación y Academia*, *7*(2), 55-61. https://revistas.udistrital.edu.co/index.php/tia/article/view/15685

Calvo, D. (2017). *Red Neuronal Convolucional CNN - Diego Calvo*. https://www.diegocalvo.es/red-neuronal-convolucional/

caracol.com. (2021). *Noticias Huila Huila será potencia de aguacate Hass : Huila será potencia de aguacate Hass*. https://caracol.com.co/emisora/2021/08/09/neiva/1628526750_603993.html

Castillo Sarasty, J. E., & Presencial. (2021). *Prototipo web para predicción y detección de incendios forestales en los cerros orientales de Bogotá, mediante una red de sensores e inteligencia artificial*. http://repository.unipiloto.edu.co/handle/20.500.12277/9883

Cifuentes, A., Mendoza, E., Lizcano, M., Santrich, A., & Moreno-Trillos, S. (2019). Desarrollo de una red neuronal convolucional para reconocer patrones en imágenes Development of a

convolutional neural network to recognize pat-terns in images. En *Revista I+D en TIC* (Vol. 10, Issue 2). http://revistas.unisimon.edu.co/index.php/identic

Corvalán, J. G. (2018). Inteligencia artificial: retos, desafíos y oportunidades - Prometea: la primera inteligencia artificial de Latinoamérica al servicio de la Justicia. *Revista de Investigações Constitucionais*, *5*(1), 295-316. https://doi.org/10.5380/RINC.V5I1.55334

DataScientest. (2021). *Kaggle: todo lo que hay que saber sobre esta plataforma*. https://datascientest.com/es/kaggle-todo-lo-que-hay-que-saber-sobre-esta-plataforma

de Ingeniería, F., Arquitectura, Y., & Villafuerte, N. E. (2019). *Diagnóstico automatizado para la clasificación de café trillado con broca mediante procesamiento de imágenes con Deep Learning Por*. https://repositorio.upeu.edu.pe/bitstream/handle/20.500.12840/1961/Nilda_Tesis_Licenciatura_2019.pdf?sequence=5&isAllowed=y

Detección de objetos - The Machine Learners. (2021). https://www.themachinelearners.com/introduccion-deteccion-objetos/

Diario del Huila. (2020). *El Huila entra en el 'boom' del aguacate Hass*. https://diariodelhuila.com/el-huila-entra-en-el-boom-del-aguacate-hass/

Edison, I. M. E., Chango, C., Carrera, Angel, M., & Arequipa, Q. (2020). *APLICACIÓN MÓVIL PARA LA IDENTIFICACIÓN DE LAS PROPIEDADES MEDICINALES DE PLANTAS NATIVAS LOCALIZADAS EN LA MICROEMPRESA "AGROCOTOPAXI".*

EfficientDet: Scalable and Efficient Object Detection. (2020). https://github.com/google/automl/tree/

Erazo, R. (2022). *Ricardo Erazo | Novice | Kaggle*. https://www.kaggle.com/ricardoerazoo

Erik Leonel Otero, A. E. F., & Yago Graña De Brasi. (2019). *Deep Learning, la tecnología del mañana*. https://grupogemis.com.ar/wp-content/uploads/2019/05/SYO_L_DeepLearning.pdf

Eugenia, M., & Zuluaga, L. (2018). *Manejo integrado de Monalonion velezangeli en aguacate*.

Eugenia, M., & Zuluaga, L. (2020). *Manejo integrado de Monalonion velezangeli en aguacate*.

Felipe, A., Rueda, C., Dario, W., & Rios, P. (2018). *Technical analysis of cryptocurrency markets with convolutional neural networks*.

Ferrer Martínez, C. (2021). *Sistema de monitorización y detección de plagas en cultivos aplicando algoritmos de Deep Learning*. https://openaccess.uoc.edu/handle/10609/132028

Fuentes, A., Yoon, S., Kim, S. C., & Park, D. S. (2017). A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition. *Sensors (Switzerland)*, *17*(9). https://doi.org/10.3390/s17092022

Fuentes Pérez, E. M. (2020). La industria alimentaria frente a la nueva normalidad post COVID-19. *CienciAmérica: Revista de Divulgación Científica de La Universidad Tecnológica Indoamérica, ISSN-e 1390-9592, Vol. 9, Nº. Extra 2, 2020 (Ejemplar Dedicado a: ESPECIAL "Desafíos Humanos Ante El COVID-19"), Págs. 45-50*, *9*(2), 45-50. https://doi.org/10.33210/ca.v9i2.290

Garzón, A. (2020). Oportunidad de exportación de aguacate Hass a Francia. *Revista Colombiana de Ciencias Administrativas*, *2*(1), 78-102. https://doi.org/10.52948/RCCA.V2I1.164

Ghaisi, G., Lin, T.-Y., Pang Quoc, R., & le Google Brain, V. (2019). *NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection*.

Goldsborough, P. (2018). *A Tour of TensorFlow Proseminar Data Mining*. https://github.com/soumith/cudnn.torch

Google. (2022). *Te damos la bienvenida a Colaboratory - Colaboratory*. https://colab.research.google.com/?hl=es

Google Brain Team. (2020). *EfficientDet: Scalable and Efficient Object Detection*. https://github.com/google/automl/tree/

Google Developers. (2020). *TensorFlow Lite*. https://www.tensorflow.org/lite/guide?hl=es-419

Google Research. (2019). *Google AI Blog: EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling*. https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

ICA. (2017). *Instituto Colombiano Agropecuario - ICA*. https://www.ica.gov.co/noticias/ica-huila-plan-exportacion-aguacate.aspx

Ignacio Herrera, & Alejandro Figueroa. (2016). *Aprendizaje Semi-Supervisado de Múltiples Vistas para Detectar Temporalidad de Preguntas*. https://doi.org/10.13140/RG.2.1.1403.1602

Jocher, glenn, & ultralytics. (2021). *ultralytics/yolov5: YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite*. https://github.com/ultralytics/yolov5

Juan I. Forcen, Humberto Bustince, Miguel Pagola, & Edurne Barrenechea. (2018). *Adaptacion autom ´ atica del operador de ´ pooling aprendiendo pesos de medias ponderadas ordenadas en Redes Neuronales Convolucionales*. https://sci2s.ugr.es/caepia18/proceedings/docs/CAEPIA2018_paper_195.pdf

Juliana Moraes Boldini. (2020). *Manejo integrado de las principales plagas y enfermedades en aguacate Hass (Persea americana) en el departamento de Caldas.* https://repository.unad.edu.co/bitstream/handle/10596/38446/dosquebradas.pdf?sequence=3&isAllowed=y

Kaggle. (2022). *Pooling layer and its types explained! | Data Science and Machine Learning | Kaggle*. https://www.kaggle.com/general/175896

Kai fu lee. (2020). *Superpotencias de la inteligencia artificial*. https://planetadelibrosec0.cdnstatics.com/libros_contenido_extra/43/42371_Superpotencias_de_la_IA.pdf

Kukil. (2022). *Intersection Over Union IoU in Object Detection Segmentation*. https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/

Leslie Pack Kaelbling, Michael L. Littman, & Andrew W. Moore. (2015). *Reinforcement Learning: A Survey*. https://www.jair.org/index.php/jair/article/view/10166/24110

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). *Feature Pyramid Networks for Object Detection*.

Marisol Giraldo Jaramillo, P. B.-M. C. V. G. (2010). *ASPECTOS MORFOLÓGICOS Y BIOLÓGICOS DE Monalonion velezangeli Carvalho & Costa (Hemiptera: Miridae) EN CAFÉ*. https://www.cenicafe.org/es/publications/arc061(03)195-2052.pdf

MATLAB & Simulink. (2022). *Redes neuronales convolucionales* . https://la.mathworks.com/discovery/convolutional-neural-network-matlab.html

Microsoft. (2021). *COCO - Common Objects in Context*. https://cocodataset.org/#home

Microsoft Learn. (2022). *Deep learning vs. machine learning* . https://learn.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning

MinAgricultura. (2021). *Inicio*. http://aguacatehass.minagricultura.gov.co/

Nexusintegra. (2022). *Qué es la visión artificial y su aplicación en la Industria 4.0*. https://nexusintegra.io/es/vision-artificial-industria-4-0/

Overflow, S. (2021). *Stack Overflow - Where Developers Learn, Share, & Build Careers*. https://stackoverflow.com/

Pérez, J. M., Zuluaga, M. E. L., García, D. A. M., & Londoño, G. A. C. (2014). Evaluación de Insecticidas para el Manejo de Monalonion velezangeli, Carvalho & Costa (Hemiptera: Miridae) en Aguacate. *Revista Facultad Nacional de Agronomía Medellín*, *67*(1), 7141-7150. https://doi.org/10.15446/rfnam.v67n1.42604

Piscoya Ferreñan, J. E. (2019). *SISTEMA DE VISIÓN ARTIFICIAL PARA APOYAR EN LA IDENTIFICACIÓN DE PLAGAS Y ENFERMEDADES DEL CULTIVO DE SANDÍA EN EL DISTRITO DE FERREÑAFE*. https://tesis.usat.edu.pe/bitstream/20.500.12423/2356/1/TL_PiscoyaFerre%c3%b1anJesus.pdf

Politécnica, U., Madrid, D. E., Fin, T., & Máster, D. E. (2018). *ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INFORMÁTICOS REDES NEURONALES CONVOLUCIONALES PROFUNDAS PARA EL RECONOCIMIENTO DE EMOCIONES EN IMÁGENES*.

Qian Yu, Dongyuan Mao, & Jingfan Wang. (2019). *Deep Learning Based Food Recognition*. http://cs229.stanford.edu/proj2016/report/YuMaoWang-Deep%20Learning%20Based%20Food%20Recognition-report.pdf

¿Qué es deep learning? - Colombia | IBM. (2021). https://www.ibm.com/co-es/cloud/deep-learning

¿Qué es la Visión Artificial? | IBM. (s. f.). Recuperado 20 de septiembre de 2022, a partir de https://www.ibm.com/co-es/topics/computer-vision

Quiroz, B., Luis, V. A., Cabrera, M. M., & Ivan, H. (2021). *FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO*. https://repositorio.uss.edu.pe/bitstream/handle/20.500.12802/8392/Quiroz%20Valencia%20Adler%20Luis.pdf?sequence=1&isAllowed=y

Robert, R., Thomas, Guillem, Elvis, Marcin, & Andrew. (2019). *BiFPN Explained | Papers With Code*. https://paperswithcode.com/method/bifpn

Rostros, R. de, & Algoritmos De Reconocimiento De Objetos De La Biblioteca Opencv Edison Rene Caballero Barriga, N. Y. (2017). *APLICACIÓN PRÁCTICA DE LA VISIÓN ARTIFICIAL PARA EL*. https://repository.udistrital.edu.co/bitstream/handle/11349/6104/CaballeroBarrigaEdisonRene2017.pdf?sequence=1&isAllowed=y

Russo, C., Ramón, H., Alonso, N., Cicerchia, B., Esnaola, L., & Tessore, J. P. (2019). *Tratamiento Masivo de Datos Utilizando Técnicas de Machine Learning*.

Salazar Gualoto, R. C. (2020). *"ESTUDIO PARA LA IMPLEMENTACIÓN DE UN SISTEMA DE SEGURIDAD PERIMETRAL INFORMÁTICA PARA EL LABORATORIO DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN DE LA FACULTAD DE INGENIERÍA DE LA PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR"*. http://201.159.222.35/bitstream/handle/22000/18674/MTI%20TESIS%20-Salazar%20Gualoto%20Roberto%20Carlos_2020-12-01%20.pdf?sequence=1&isAllowed=y

Shah, D. (2022). *Mean Average Precision (mAP) Explained: Everything You Need to Know*. https://www.v7labs.com/blog/mean-average-precision

Shu Liu, LuQi, Haifang Qin, Jianping Shi, & Jiaya Jia. (2018). *Path Aggregation Network for Instance Segmentation*. https://arxiv.org/pdf/1803.01534v4.pdf

TensorFlow. (s. f.). Recuperado 12 de octubre de 2022, a partir de https://www.tensorflow.org/

TIBCO Software. (2020). *¿Qué es el aprendizaje supervisado?* . https://www.tibco.com/es/reference-center/what-is-unsupervised-learning

Torralba, P. P. (2022). Qué son las Redes Neuronales Convolucionales. *Thinking for Innovation*. https://www.iebschool.com/blog/redes-neuronales-convolucionales-big-data/

Transfer Learning - MATLAB & Simulink. (2022). https://la.mathworks.com/discovery/transfer-learning.html

TzuTa Lin. (2021). *labelImg · PyPI*. https://pypi.org/project/labelImg/1.4.0/

Xiaomi. (2020). *Mi Mexico*. https://www.mi.com/mx/redmi-note-9-pro/specs/

Xiaoshu, J. (2020). *论文阅读：NAS-FPN_贾小树的博客-CSDN博客*. https://blog.csdn.net/j879159541/article/details/106926689