UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

DISEÑO DE UN SISTEMA EMBEBIDO DE MONITOREO POR VISIÓN ARTIFICIAL QUE PERMITA MEDIR EL GRADO DE MADUREZ DE LAS FRUTAS

TRABAJO DE GRADO PREVIO A LA OBTENCION DEL TITULO DE INGENIERO EN TELECOMUNICACIONES

AUTOR: CARLOS ARTURO ERAZO NARVÁEZ

DIRECTOR: MSC. EDGAR ALBERTO MAYA OLALLA

IBARRA-ECUADOR

2023

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1004561948		
APELLIDOS Y NOMBRES:	Erazo Narváez Carlos Arturo		
DIRECCIÓN:	Atuntaqui- Calle Bolívar y Los Geranios		
EMAIL:	caerazon@utn.edu.ec		
TELÉFONO FIJO:	062909425	**TELÉFONO MÓVIL:**	0981326556

DATOS DE LA OBRA	
TÍTULO:	DISEÑO DE UN SISTEMA EMBEBIDO DE MONITOREO POR VISIÓN ARTIFICIAL QUE PERMITA MEDIR EL GRADO DE MADUREZ DE LAS FRUTAS
AUTOR (ES):	Erazo Narváez Carlos Arturo
FECHA DE APROBACIÓN: DD/MM/AAAA	03/08/2022
PROGRAMA:	■ **PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	Ingeniero en Telecomunicaciones
ASESOR /DIRECTOR:	MSc. Edgar Maya

2. CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 18 días del mes de octubre de 2023

EL AUTOR:

.....

Carlos Arturo Erazo Narváez

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCAS APLICADAS

CERTIFICACIÓN:

MAGÍSTER EDGAR MAYA, DIRECTOR DEL PRESENTE TRABAJO DE TITULACIÓN CERTIFICA:

Que el presente trabajo de Titulación DISEÑO DE UN SISTEMA EMBEBIDO DE MONITOREO POR VISIÓN ARTIFICIAL QUE PERMITA MEDIR EL GRADO DE MADUREZ DE LAS FRUTAS, ha sido desarrollado por el señor Erazo Narváez Carlos Arturo bajo mi supervisión.

Es todo cuanto puedo certificar en honor a la verdad.

MSc. Edgar Maya Olalla
DIRECTOR

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

DEDICATORIA

Dedico este logro principalmente a mi familia quienes siempre me brindan un gran apoyo en las metas que me propongo, son personas muy especiales en mi vida, ya que me han dado soporte incluso en las circunstancias no tan favorables

A mi padre y mi madre quienes me ha sabido guiar de una manera adecuada durante todo este proceso de crecimiento profesional, además inculcándome valores mediante las enseñanzas a lo largo de este tiempo.

También quiero dedicar este logro a mi hermana, quien ha sido como una amiga incondicional que me apoya y me ayuda a seguir adelante.

Amigos y compañeros quienes que me han acompañado casi toda la carrera, apoyándome, ayudándome, formando un gran equipo el cual ha superado varias situaciones, procurando que siempre todos avancemos a la par.

Finalmente, mi novia que me ha regalado varias horas de su tiempo para escucharme, para darme el aliento para seguir adelante.

Erazo Narváez Carlos Arturo

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

AGRADECIMIENTO

A Dios, a la carrera de ingeniería en Telecomunicaciones, a la Universidad por permitirme estudiar y desarrollarme como un profesional competente y responsable.

A mis profesores quienes me han guiado durante este proceso de formación académica, compartiendo sus conocimientos y experiencias.

A mis padres quienes han sido un pilar importante en mi vida, brindándome un apoyo incondicional durante todo este proceso de formación personal y académica, otorgándome su confianza y a su vez siendo parte importante de cada obstáculo que supero

A mi director de tesis Msc. Edgar Maya y mi asesor de tesis Msc. Jaime Michilena, quienes me han sabido orientar y guiar de una manera adecuada para el desarrollo de este trabajo de titulación, obteniendo un logro inmenso en mi vida.

Erazo Narváez Carlos Arturo

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

RESUMEN

En el trabajo de titulación que se ha desarrollado se describe el diseño y la construcción de un sistema de monitoreo por visión artificial para medir el grado de madurez de plátanos y fresas, con el fin de evitar el desperdicio de estas frutas o alguna enfermedad relacionada con el consumo de alimentos en mal estado. Cabe recalcar que este sistema de monitoreo consta con una aplicación móvil para dispositivos Android para visualizar los resultados de las frutas monitoreadas, además esta aplicación móvil tiene la característica de emitir una notificación de alerta cuando una o más frutas se encuentran en un estado de maduración excesiva o podrido.

El proyecto fue realizado mediante la utilizando de la metodología en cascada la cual consta de 5 fases. La primera fase se indica el levantamiento de requerimientos, es decir, se indaga de forma teórica acerca de los elementos necesarios para el diseño y la construcción del sistema; para la siguiente fase, en base a los requerimientos del sistema, se plantea cada uno de los componentes de hardware y software que se van a utilizar. En la tercera fase se realiza la construcción del sistema utilizando cada uno de los componentes elegidos anteriormente, tomando en cuenta la integración de todas las partes para el funcionamiento adecuado del sistema de monitoreo, siguiente con el proceso en la fase de verificación se realizaron pruebas de funcionamiento enfocadas

principalmente a medir la precisión del sistema, estas pruebas se efectuaron con frutas con el mismo nivel de madurez, así como también con diferentes estados de madurez, para la fase final se definen parámetros extra para que el sistema sea capaz de funcionar de forma autónoma. Como resultado de la construcción de este sistema de monitoreo, se ha obtenido un nivel de confianza aproximado a 80%, este porcentaje tiene un rendimiento aceptable, debido a que la mayor parte de la fruta es detectada.

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

ABSTRACT

The degree work that has been developed describes the design and construction of a monitoring system by artificial vision to measure the degree of maturity of bananas and strawberries, in order to avoid the waste of these fruits or any disease related to the consumption of food in poor condition. It should be emphasized that this monitoring system consists of a mobile application for Android devices to visualize the results of the monitored fruits, in addition this mobile application has the feature of issuing an alert notification when one or more fruits are in a state of overripe or rotten.

The project was carried out using the cascade methodology, which consists of 5 phases. The first phase indicates the requirements gathering, i.e., a theoretical investigation of the elements necessary for the design and construction of the system; for the next phase, based on the system requirements, each of the hardware and software components to be used is proposed. In the third phase the construction of the system is carried out using each of the components previously chosen, taking into account the integration of all the parts for the proper functioning of the monitoring system, following with the process in the verification phase, performance tests were performed mainly focused on measuring the accuracy of the system, these tests were performed with fruits with the same level of maturity, as well as with different states of maturity, for the final phase extra parameters are defined so that the system is able to function autonomously. As a result of the

construction of this monitoring system, a confidence level of approximately 80% has been obtained; this percentage has an acceptable performance, since most of the fruit is detected.

INDICE DE CONTENIDOS

INDICE DE TABLAS

INDICE DE FIGURAS

Capítulo I

Antecedentes

En este capítulo se detallada los requerimientos necesarios para el desarrollo del presente trabajo de titulación, siendo estos: el tema elegido, la problemática, los objetivos, el alcance, la justificación con la finalidad de concluir este proyecto de una manera exitosa.

1.1. Tema

DISEÑO DE UN SISTEMA EMBEBIDO DE MONITOREO POR VISIÓN ARTIFICIAL QUE PERMITA MEDIR EL GRADO DE MADUREZ DE LAS FRUTAS.

1.2. Problema

El sector alimenticio es uno de los ejes principales de la sociedad. Dentro de este sector se debe considerar un control de calidad y estado de los mismos, especialmente en productos agrícolas como frutas y verduras, las cuales son más susceptibles a hongos y bacterias, lo que a su vez contribuye al deterioro de los mismos productos y posteriormente a ser desechados (Trigos, Ramírez y Salinas, 2008). Las hifas fúngicas (filamentos que constituyen el cuerpo de los hongos multicelulares) cuando infectan una fruta se propagan en espacios intercelulares y dentro del parénquima (tejido esponjoso de las frutas que cumplen diversas funciones), lo cual causa la maceración del tejido de la fruta, que posteriormente condice a la descomposición de la fruta (Soto, Tramón, Aqueveque, & De Bruijn, 2018).

Actualmente, el problema radica en los hogares, en donde el desperdicio de alimentos de productos agrícolas principalmente son frutas, debido a causas relacionadas como falta de tiempo

de tiempo para la revisión de estos productos alimenticios, ambiente poco adecuado para su almacenamiento y desconocimiento del estado de madurez; esto puede afectar de forma parcial el nivel socioeconómico de una familia por el desperdicio de estos productos alimenticios.

En base a los planteamientos mencionados con anterioridad, se manifiesta una necesidad de la generación de nuevas tecnologías que permitan conocer el estado de madurez de los alimentos de producción agrícola como frutas, donde los usuarios tengan la capacidad de obtener información de estos productos alimenticios, dando paso a tomar medidas ante el estado de madurez de las frutas para evitar el desperdicio.

1.3. Objetivos

1.3.1. Objetivo General

Diseñar un sistema embebido de monitoreo inteligente por visión artificial que permita medir el grado de madurez de las frutas para evitar el desperdicio y enfermedades causadas por intoxicaciones alimentarias bacterianas.

1.3.2. Objetivos Específicos

- Analizar el estado del arte acerca de visión artificial, redes neuronales, Deep Learning, tipo de frutas y algoritmos de reconocimiento para el levantamiento de requerimientos necesarios que utilizará el sistema de monitoreo embebido.
- Esquematizar el diseño de un sistema basado en una arquitectura IoT, que permita la obtención de datos acerca de las frutas que se va a monitorear y procesamiento de los datos mediante redes neuronales y Deep Learning.
- Construir un prototipo mediante la implementación de hardware y software que posibiliten

el cumplimiento de requerimientos del sistema embebido para la medición del grado de madurez de las frutas por medio de visión artificial.

- Verificar los resultados obtenidos para el establecimiento de rangos de precisión acerca funcionamiento del sistema embebido de monitoreo.

1.4. Alcance

El desarrollo del proyecto se basa en la metodología en cascada, en donde, para el sistema de embebido de monitoreo del grado de madurez de las frutas como: plátanos y fresas, es necesario realizar el análisis del estado del arte acerca de visión artificial, redes neuronales, el tipo de frutas, algoritmos de identificación, definición de umbrales del estado de madurez de las frutas a estudiar, además, de los recursos de software que se va utilizar para este proyecto; integrando de esta manera todos los requerimientos que conforman la etapa de inicial de este sistema.

Para la etapa de diseño (Tabla 1), de forma inicial se especifica que se requiere de un dataset (conjunto de datos tabulados) acerca de las frutas como: plátanos y fresas, en donde, como se indica en la tabla 1 para la sección de creación del dataset, esta base de datos se construye, a partir de la captura de varias imágenes de las frutas durante el proceso de maduración utilizando una cámara, estas mismas imágenes son etiquetadas de acuerdo a su tonalidad y saturación, posteriormente estos datos son almacenados y tabulados en el dataset. En la sección de arquitectura de Deep Learning, a partir del dataset construido anteriormente se realiza el entrenamiento de un modelo mediante la utilización de redes neuronales y Deep Learning, a continuación, se realiza la exportación de las métricas del modelo entrenado. Y finalmente, para la última sección, se realiza la importación de las métricas del modelo entrenado dentro del sistema embebido, además, mediante la implementación de una plataforma se permite la incorporación de funcionamiento del algoritmo de clasificación y las métricas del modelo entrenado en el sistema embebido de

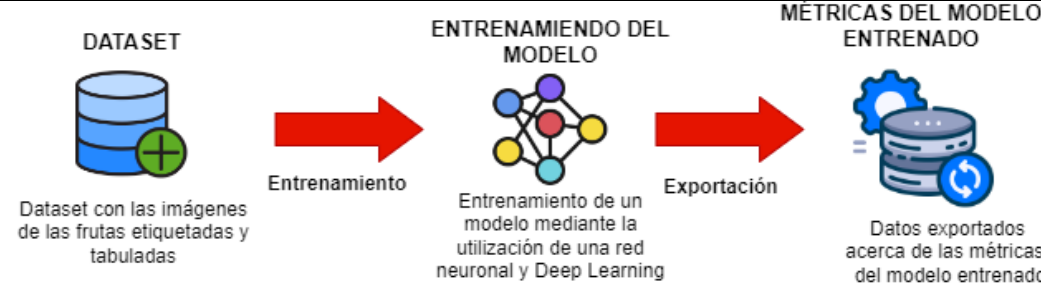
monitoreo, de esta forma se puede realizar el procesamiento de la información adquirida por medio de la cámara para establecer el grado de madurez que tiene la fruta monitoreada, obteniendo como último paso, la visualización de los resultados.

En la etapa construcción del sistema embebido se implementa hardware y software como: una cámara, un dispositivo procesamiento, un algoritmo de clasificación, el modelo entrenado por Deep Learning y una aplicación móvil para Android, en donde a través de la formación de este sistema posibilita la medición del grado de madurez de los plátanos y fresas, considerando todos los requerimientos y la estructuración del diseño que se ha realizado en la etapa anterior.

Finalmente, para la etapa verificación y mantenimiento, se establece pruebas de funcionamiento para el sistema embebido de monitoreo, para analizar la precisión del sistema y comprobar la eficiencia para determinar del grado de madurez de las frutas.

Tabla 1

Arquitectura del sistema embebido de monitoreo

Sección	Representación gráfica de la sección
Creación del dataset	
Arquitectura de Deep Learning	

Importación de las métricas del modelo entrenado en el sistema embebido	MÉTRICAS DEL MODELO ENTRENADO  Datos exportados acerca de las métricas del modelo entrenado	SISTEMA EMBEBIDO Importación  Sistema embebido con las métricas del modelo entrenado para medición del grado de madurez de las frutas	RESULTADOS Se obtiene  Resultados del proceso de medición del grado de madurez de las frutas

Nota. La tabla indica las diferentes secciones de la arquitectura del sistema embebido de monitoreo. Fuente: Autoría propia

1.5. Justificación

El sector alimenticio es de suma importancia en la sociedad, ya que, debido a que este factor está relacionado directamente con la salud de los consumidores. Los productos alimenticios como: carnes, frutas, vegetales, etc., deben tener un control de calidad y de estado minuciosos, puesto que, estos productos en mal estado o un excesivo estado de madurez contribuyen al desarrollo de bacterias y microorganismos, los mismos que a su vez pueden conducir a enfermades relacionadas con los alimentos o incluso puede provocar una intoxicación alimentaria.

Las enfermedades transmitidas por alimentos (ETAS), forman parte de uno de los problemas de la salud pública a nivel mundial, debido a que la contaminación de los productos alimenticios puede ocurrir durante el proceso de producción de los mismos alimentos hasta el consumo final, a causa de factores como: almacenamiento inadecuando de estos productos, contaminación por bacterias, virus, entre otros agentes externos.

El consumo de estos alimentos contaminados puede generar síntomas gastrointestinales, sin embargo, también este tipo de enfermedades pueden producir síntomas neurológicos, ginecológicos, inmunológicos y de otro tipo (Ministerio de Salud Pública, 2021).

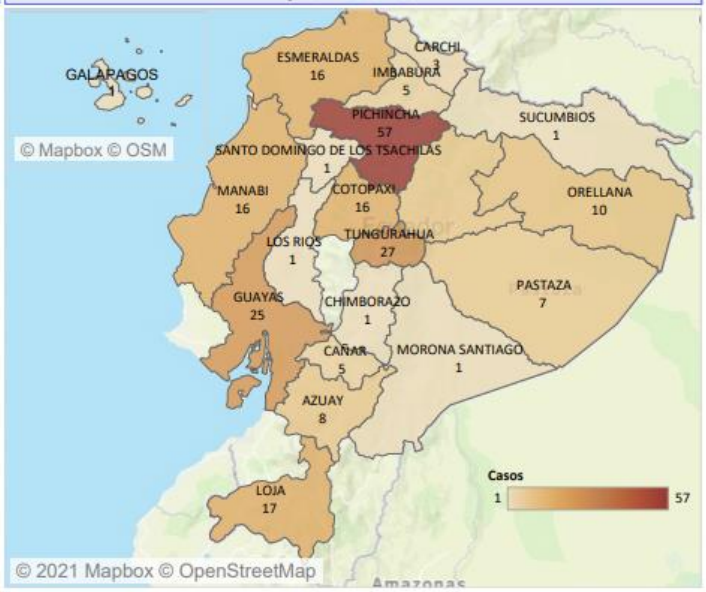
En Ecuador según el Ministerio de Salud Pública (Ministerio de Salud Pública, 2021), indica que los alimentos pueden ser contaminados por bacterias en cualquier momento de la producción o del procesamiento, los síntomas principales de intoxicaciones alimentarias bacterianas son: nauseas, diarrea, dolor y calambres abdominales y fiebre; en el año 2020 existieron 5890 casos por esta enfermedad y hasta enero del 2021 como se indica en la Figura 1, en todo el país se presentan 226 casos por intoxicaciones alimentarias bacterianas.

Figura 1

Casos notificados de A040-A049 Otras intoxicaciones alimentarias bacterianas por provincia.

Casos notificados de A040-A049 Otras intoxicaciones alimentarias bacterianas por provincia SE 03

PROVINCIA	SE 01-02	SE 03	Total
PICHINCHA	40	17	57
TUNGURAHUA	11	16	27
GUAYAS	20	5	25
LOJA	10	7	17
MANABI	8	8	16
ESMERALDAS	10	6	16
COTOPAXI	13	3	16
ORELLANA	6	4	10
NAPO	3	5	8
AZUAY	8		8
PASTAZA	5	2	7
IMBABURA	5		5
CAÑAR	4	1	5
CARCHI	2	1	3
SUCUMBIOS	1		1
SANTO DOMINGO DE L..	1		1
MORONA SANTIAGO		1	1
LOS RIOS		1	1
GALAPAGOS		1	1
CHIMBORAZO	1		1
Total	148	78	226



Nota. En la figura se indica los casos por intoxicaciones alimentarias bacterianas. Tomado de SUBSISTEMA DE VIGILANCIA SIVE - ALERTA ENFERMEDADES TRANSMITIDAS POR AGUA Y ALIMENTOS ECUADOR Ministerio de Salud Pública (2021). https://www.salud.gob.ec/wp-content/uploads/2021/01/Etas-SE-03.pdf

Aproximadamente el 14% de los alimentos a nivel mundial se desperdician desde el proceso de postcosecha hasta el nivel minorista, el cual conlleva una pérdida valorada en 400 000 millones de dólares cada año y, además, referente al impacto ambiental, esta pérdida y desperdicio de alimentos producen el 8% de las emisiones de gases de efecto invernadero (GEI) (FAO, 2020). Los alimentos de origen agrícola en mal estado pueden generarse a por factores como: manipulación deficiente, ambiente poco adecuado para el almacenamiento, falta de capacidad de capacidad de frío, entre otras causas. La reducción del desperdicio de los alimentos conduce a un mejor aprovechamiento de la explotación agrícola, generando mayor cantidad de alimentos y contribuyendo a la disminución de las emisiones de GEI.

Capitulo II

2.1. Maduración de la fruta

La maduración de la fruta consiste en un conjunto de procesos que suceden desde las etapas posteriores al crecimiento y desarrollo, hasta cuándo la fruta va a ser consumida. Los principales cambios que se dan en las frutas son: el ablandamiento de la pulpa, el aumento del contenido de azúcar y los niveles de acidez se reducen (Capino & Farcuh, 2022). Además, para determinar los patrones o indicadores de maduración de una fruta se debe considerar la división entre: frutas climatéricas (pueden madurar después de la cosecha) y no climatéricas (no pueden madurar después de la cosecha). También como se indica en la siguiente imagen.

2.1.1. Indicadores de maduración

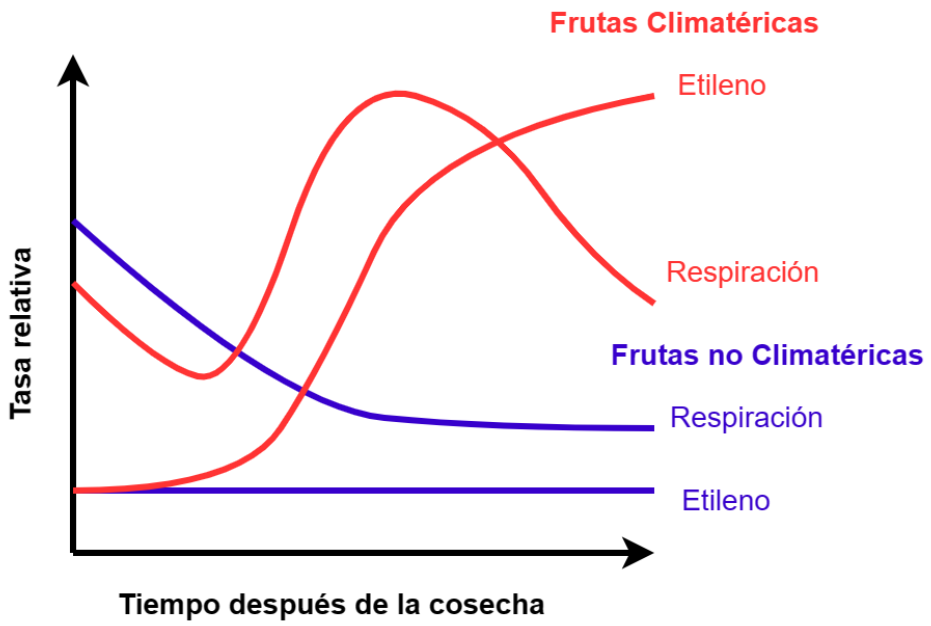
En el proceso de maduración de un fruto climatérico, se indica una mayor tasa de respiración y aumento de los niveles de etileno, esto debido la concentración inicial de etileno que posee el fruto. A medida que la producción del etileno crece la maduración de la fruta se acelera; cabe recalcar que este proceso sucede cuando se cosecha la fruta. Algunos ejemplos de frutos climatéricos son: plátanos, manzanas, mangos, aguacates, entre otras.

Para los frutos no climatéricos los niveles de etileno no aumentan cuando se cosecha la fruta, por lo cual, este tipo de productos deben ser cosechado cuando estén completamente maduros. Algunos ejemplos de frutas no climatéricas son: fresas, piñas, uvas, naranja, entre otras.

Como se indica en la Figura 2, se puede visualizar los procesos antes descritos para las frutas climatéricas y no climatéricas, en donde se realiza la representación de dos parámetros fundamentales durante el proceso de maduración de estas frutas, los cuales son: el valor del etileno y la tasa de respiración de la fruta.

Figura 2

Patrones de maduración de frutos climatéricos y no climatéricos



Nota. En la figura se indica la tendencia de los niveles etileno y respiración es frutas climatéricas y no climatéricas. Adaptado de *Ethylene and the Regulation of Fruit Ripening* (Gao et al., 2020).

2.1.1.1. Biosíntesis del etileno.

En la maduración climatérica de una fruta sufre un pico de producción de etileno cuando se cosecha el producto, disminuyendo la vida útil de la fruta y aumentando cierta susceptibilidad a los ataques de patógenos o al crecimiento de hongos y bacterias dentro del mismo fruto.

2.1.1.2. Color.

Entre los factores más notables para determinar la maduración de una fruta es el cambio de color que sufren durante este proceso. En varias frutas los distintos colores son resultado de la pérdida de clorofila y la acumulación de antocianinas (tipo de antioxidantes naturales) relacionados

con colores como: purpura, azul y rojo; o con carotenoides (tipo de antioxidantes naturales) relacionado con los colores: rojo, naranja o amarillo (Gao et al., 2020).

2.1.1.3. Sabor.

El cambio de sabores dulces y ácidos son indicadores importantes en la maduración de una fruta. Aunque las frutas climatéricas son dependientes del etileno aún no existe la suficiente evidencia que indique la relación entre la metabolización de azucares solubles y ácidos orgánicos con la producción del etileno en la fruta.

2.1.2. Tiempo de maduración

Para el tiempo de maduración de las frutas se debe considerar factores como: el tipo de fruta (climatérica o no climatérica), temperatura de almacenamiento, tipo de ambiente de almacenamiento, entre otros. En este caso, a continuación, se indica el tiempo de maduración para dos frutas: plátano y fresas.

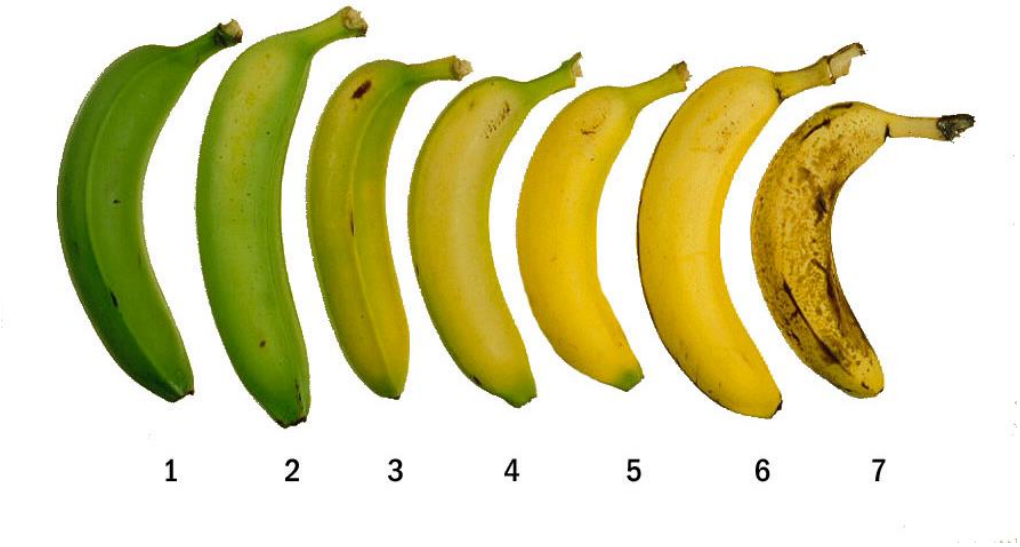
2.1.2.1. Tiempo de maduración de plátanos.

El plátano es una fruta que crece en zonas climáticas tropicales y subtropicales, y además son económicamente importantes en países como: Ecuador y Filipinas; los plátanos tienen una maduración aproximada de 7 días a partir de la cosecha, sin embargo, si se conservan a una temperatura ambiente la maduración se completa en dos días (Abobakr et al., 2019).

El indicador de maduración más notables es el color de la cáscara de esta fruta, la cual se torna de un color amarillo durante este proceso, esto puede ser visualizado de mejor forma en la en la Figura 3.

Figura 3

Estados de maduración del plátano



Nota. En la figura se presenta los diferentes estados de madurez del plátano. Tomado de *Sistema de visión artificial para gestión de calidad del Banano Cavendish en etapa de postcosecha* (Nieto & Rangel, 2022). https://revistas.utp.ac.pa/index.php/ric/article/view/3670/4248

2.1.2.2. Tiempo de maduración de fresas.

La fresa es una fruta que puede ser cultivada en diversas regiones del mundo, su crecimiento puede producirse en zonas desde climas templados hasta tropicales. Esta fruta al ser no climatérica no va a tener un mayor nivel de maduración después de su cosecha, la cual se realiza entre 4 a 6 semanas a partir de su floración, después de este proceso se debe conservar en una ambiente fresco y sombreado a una temperatura de 32 a 35 grados Fahrenheit y con una humedad relativa del 90% al 95%; bajo estas condiciones las fresas pueden soportar 7 días antes de empezar con la descomposición de la fruta (Klodd et al., 2021).

En la actualidad existen diversos sistemas de monitoreo, que permiten la medición del estado de madurez de las frutas, uno de estos sistemas tiene como base de funcionamiento la utilización de sensores electroquímicos, los cuales permite la medición de etileno en las frutas durante el proceso de maduración. Sin embargo, los sistemas que se basan en este tipo de sensores tienen un funcionamiento adecuado en frutas climatéricas, en donde el índice de emisión etileno crece durante la maduración; en frutas no climatéricas, la emisión de etileno se mantiene constante durante proceso de maduración, por lo cual, existen otros tipos de sistemas más avanzados, una opción entre estos son los sistemas de monitoreo que funcionan a través del procesamiento de imágenes utilizando redes neuronales artificiales.

2.2. Redes Neuronales Artificiales

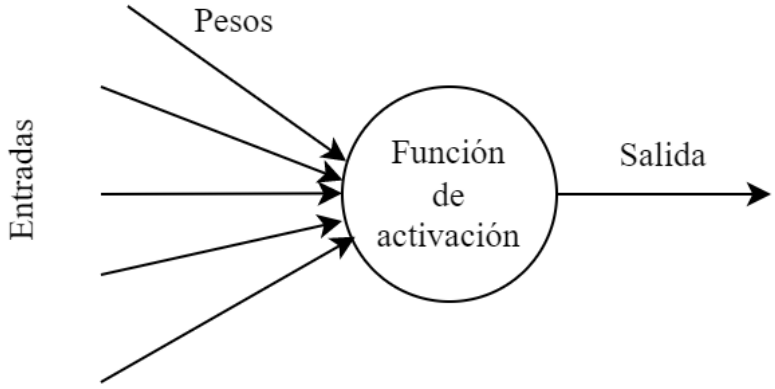
Las redes neuronales artificiales se basan en el funcionamiento del cerebro humano para resolver problemas e identificar patrones, esto por medio de programas informáticos, los cuales están enfocados a la inteligencia artificial, el aprendizaje automático y el aprendizaje profundo.

2.2.1. Neuronas Artificiales

Básicamente una neurona artificial es un modelo computacional, el cual, está basado en las neuronas humanas. Esto tipo de neuronas creadas se componen de entradas que son multiplicadas por “pesos” para cada una de las señales ingresadas, posteriormente se realiza el procesamiento computacionalmente de acuerdo a una función matemática que define la activación de la neurona, la representación de esto se observa en la Figura 4; otra función ejecuta el cálculo de salida de esta neurona, que a veces está definida por un umbral (Gershenson, 2003).

Figura 4

Estructura de una neurona artificial



Nota. En esta imagen se indica la estructura básica de una neurona artificial. Adaptado de *Artificial Neural Networks for Beginners,* (Gershenson, 2003).

2.2.2. Estructuración de una red neuronal artificial

Una red neuronal básicamente se estructura en función de su organización, la cual puede contener: número de niveles o capas, número de neuronas por nivel, patrones de conexión y flujo de información.

2.2.3. Características de las redes neuronales artificiales

Los factores más relevantes que caracterizan a las redes neuronales: son la arquitectura que poseen, el mecanismo de aprendizaje, el tipo de asociación entre la información de entrada y salida, y la representación de la información de entrada y salida (González & Martínez, 2001).

2.2.3.1. Arquitectura.

La arquitectura también puede ser definida como la topología de la red neuronal, la cual describe la estructuración, organización y arreglo que forman las neuronas en capas o en agrupaciones, distinguiendo de forma clara las entradas y salidas de la red; en base a este concepto

se puede establecer que los principales parámetros para una red de este tipo son: número de capas, número de neuronas por capa, grado de conectividad y tipo de conexión entre neuronas (González & Martínez, 2001).

2.2.3.2. Mecanismos de aprendizaje.

El método que tiene para realizar el proceso de aprendizaje es mediante la variación del valor de los pesos cuando se tiene información entrante a la red neuronal; cuando ocurre esta fase de aprendizaje existen diversos cambios como: modificación, creación y eliminación de las conexiones de las neuronas.

En redes neuronales artificiales el proceso de aprendizaje implica que los pesos de las conexiones de la red van a presentar modificaciones, por lo cual cuando los valores de estos pesos se mantengan en un rango estable se puede indicar que el modelo ha sido entrenado o a la red ha aprendido. Cabe destacar, que en base al patrón de modificación que siguen los pesos para aprender nueva información son considerados como criterios para establecer dos tipos de aprendizaje: aprendizaje supervisado y aprendizaje no supervisado.

2.2.3.3. Tipo de asociación entre la información de entrada y salida.

Existen dos maneras primarias de establecer una asociación de entre la entrada y salida de una red neuronal. La heteroasociación hace referencia a pares de datos $[(A_1, B_1), (A_2, B_2), \dots, (A_n, B_n)]$, en donde para cada información de entrada A_i, la red neuronal tendrá una salida asociada B_i. La autoasociación es determinada cuando la red aprende una serie de información $A_1, A_2, \dots, A_n$ y realiza una autocorrelación con la información de entrada, es decir, dando como respuesta uno de los datos almacenados, el cual posea una mayor similitud con la entrada.

Debido a estos criterios de asociación de entrada y salida de información se tienen dos tipos de redes neuronales: las redes heteroasociativas y autoasociativas. Las redes heteroasociativas pueden considerarse como aquellas que son capaces de procesar cierta función, la cual no podrá ser expresada analíticamente, entre conjuntos de entradas y salidas existe una relación de correspondencia para cada posible entrada una determinada salida; para una red neuronal autoasociativa se interpreta su funcionamiento como un proceso en el cual se reconstruye una determinada información de entrada incompleta o distorsionada con el dato almacenado más similar (González & Martínez, 2001).

2.2.3.4. Representación de la información de entrada y salida.

Otra de las características de las redes de neuronales artificiales es la forma de representación de los datos tanto de salida como de entrada, esta clasificación se obtiene en base a la naturaleza de los datos y su procesamiento en la red. En el tipo analógico, los datos tienen valores reales continuos, generalmente normalizados y con un valor absoluto menor a la mitad. Por otro parte, también existen redes que permiten únicamente valores discretos o binarios en su entrada y además tienen como salida el mismo tipo de datos binarios.

2.2.4. Hiperparámetros de una red neuronal

Los hiperparámetros son configuraciones modificables que se definen antes del proceso de entrenamiento de un modelo de aprendizaje; cabe recalcar que estas configuraciones inciden directamente en el resultado del entrenamiento del modelo, por lo cual, se debe ajustar los valores de estos hiperparámetros de acuerdo al entorno donde se vaya a implementar el modelo (Nyuytiymbiy, 2020). A continuación, se indica los principales hiperparámetros:

- Tasa de aprendizaje: Controla la cantidad de pasos que se toman para ajustar los pesos durante el entrenamiento.

- Tamaño de lote: Se refiere a la cantidad de ejemplos de entrenamiento que se utilizan en cada iteración.
- Épocas: Se refiere al número de iteraciones que hará por cada lote durante el proceso de entrenamiento.

2.2.5. Tipo de redes neuronales

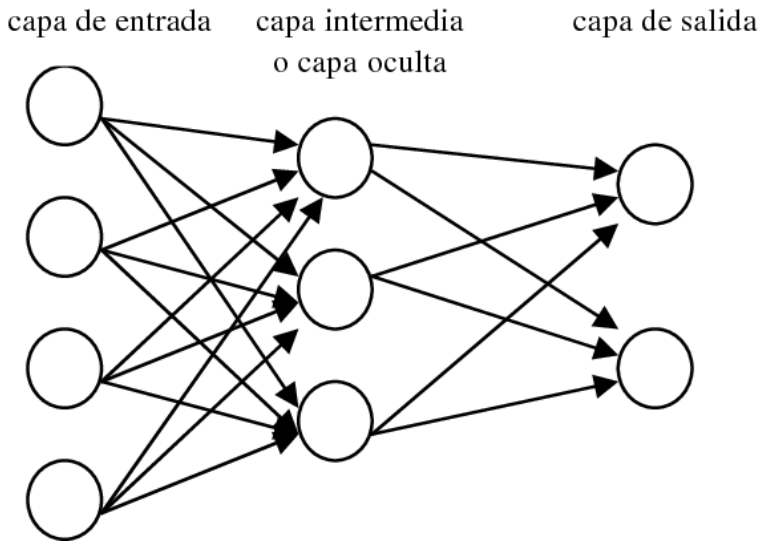
Las redes neuronales se pueden clasificar en varios tipos, dependiendo el propósito en el cual se va a utilizar, sin embargo, las clasificaciones más comunes son: red neuronal multicapa, red neuronal convolucional y red neuronal recurrente.

2.2.5.1. Red neuronal multicapa.

Este tipo de red está compuesto principalmente por una entrada, una o varias capas ocultas y una capa de salida, estas capas están constituidas de neuronas sigmoides, la cuales, por medio de esta función sigmoide convierten las entradas introducidas a una escala de (0,1), lo que permite resolución o el trabajo con problemas no lineales.(IBM Cloud Education, 2020), este modelo antes mencionado puede ser descrito en la Figura 5, en donde, se indica su estructura.

Figura 5

Red neuronal Multicapa



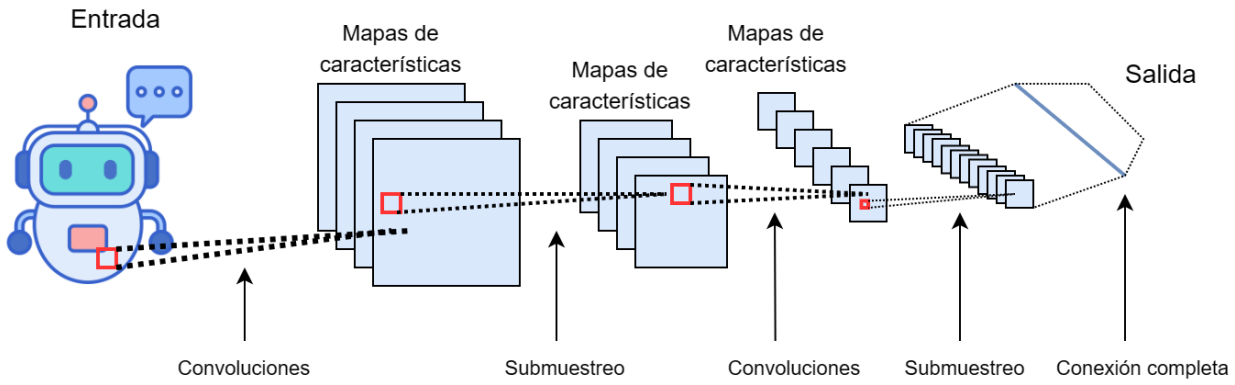
Nota. En esta imagen se indica la de una red neuronal multicapa. Adaptado de *Aplicación de una red neuronal para la predicción de la reacción catalítica isomerización del n-Octano* (Chica et al., 2001).

2.2.5.2. Red neuronal convolucional.

Las redes neuronales convolucionales son similares a las redes neuronales multicapa, sin embargo, este tipo de redes utilizan principios de álgebra lineal y operación de matrices para distinguir un patrón; son usadas generalmente para el reconocimiento de imágenes, es decir, en visión artificial. En la Figura 6 se indica el proceso que realiza este tipo de red neuronal.

Figura 6

Proceso de detección de una red neuronal convolucional



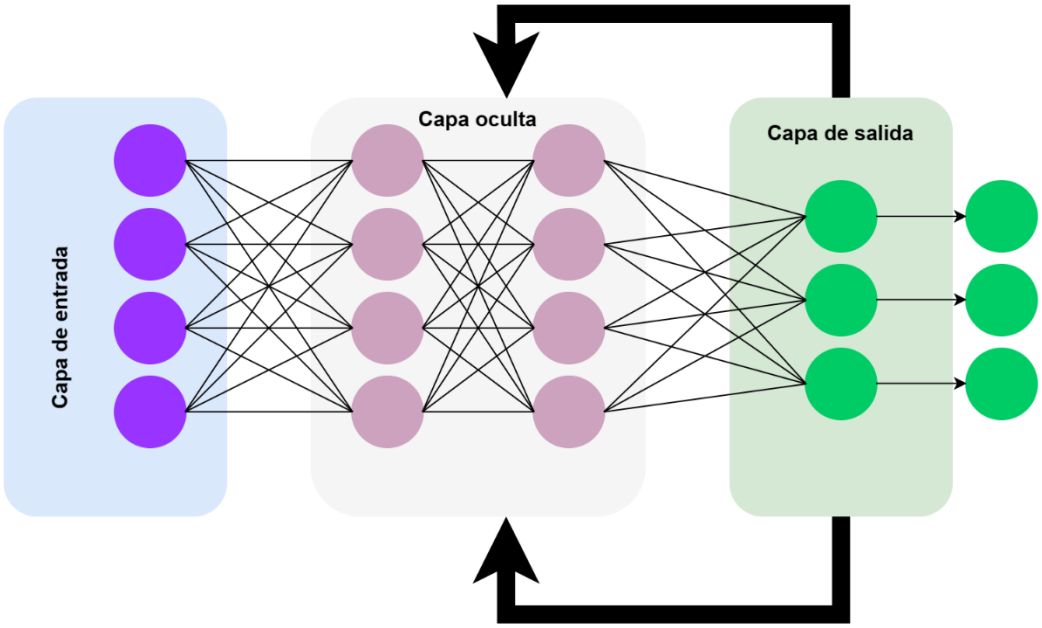
Nota. En esta figura se indica de una forma didácticas las capas y funciones de una red neuronal convolucional. Adaptado de *Detection and Tracking of Pallets using a Laser Rangefinder and Machine Learning Techniques* (Ihab, 2017).

2.2.5.3. Red neuronal recurrente.

Para este tipo de redes se utiliza bucles de retroalimentación, lo cual permite el aprendizaje por medio de series temporales para realizar una predicción acerca de posibles resultados futuros, una de sus aplicaciones puede ser el pronóstico en el mercado de valores o para otro tipo de ventas. Para la figura 7 se muestra la representación de una red neuronal recurrente

Figura 7

Representación de una red neuronal recurrente



Nota. La red neuronal recurrente tiene una retroalimentación de acuerdo a la capa de salida. Adaptado de *Simple Explanation of Recurrent Neural Network (RNN)* (Boufeloussen, 2020).

Como se ha indicado anteriormente las redes neuronales artificiales buscan replicar el funcionamiento del cerebro humano para obtener la capacitad de tomar decisiones mediante la utilización de recursos informáticos, estas redes neuronales están compuestos por características como: una arquitectura definida, un mecanismo de aprendizaje, la asociación de la información y la representación de la misma. Los parámetros otorgados a las redes neuronales artificiales para su entrenamiento tienen como resultado, el desarrollo de un modelo matemático, el cual generalmente es empleado para Machine Learning, y posteriormente implementado en un sistema informático.

2.3. Machine Learning

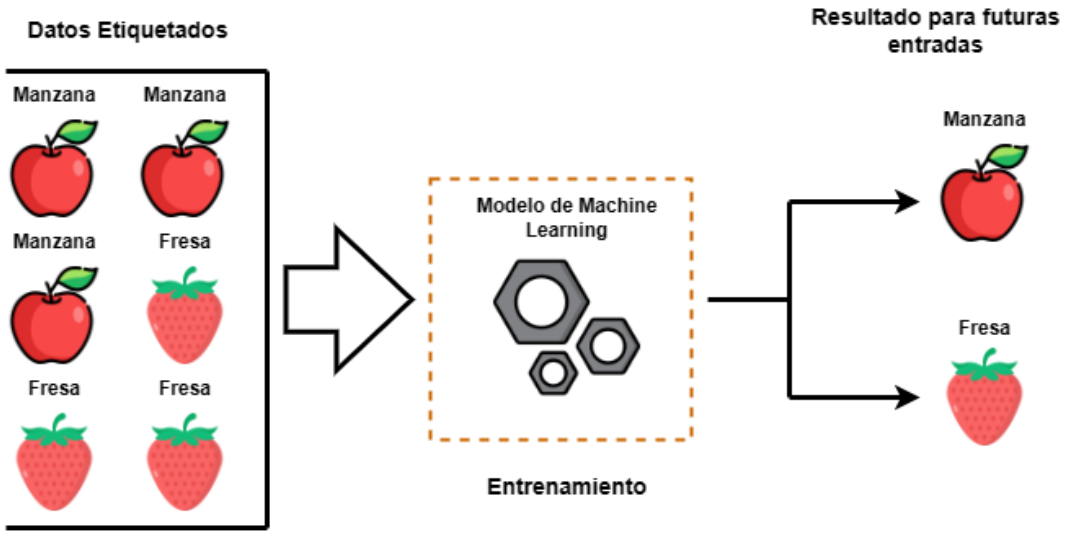
Machine Learning es denominado como un tipo de técnica que, a partir del análisis de datos, permite la construcción y uso de modelos estadísticos, esto posibilita a una máquina establecer la toma de decisiones e identificar de patrones.

2.3.1. Aprendizaje Supervisado

Este tipo de aprendizaje hace referencia a un modelo de Machine Learning, el cual obtiene esta capacidad aprendizaje a partir del entrenamiento con un conjunto de ejemplos, que tienen como resultado una variable conocida. Básicamente, estos modelos entrenan con datos etiquetados, procurando realizar la mayor cantidad de pruebas posibles, para alcanzar una relación adecuada entre la respuesta y sus predictores, con el fin predecir datos nuevos que no han sido procesados (Romero, 2020). Una representación de este modelo se visualiza en la Figura 8.

Figura 8

Representación del aprendizaje supervisado



Nota. En esta imagen se indica la representación del aprendizaje supervisado interpretado con manzanas y fresas. Adaptado de *Una introducción a los modelos de Machine Learning* (Romero, 2020).

La regresión y la clasificación son las pruebas principales para el aprendizaje supervisado, las cuales, mediante la utilización de variables en estos modelos, se puede caracterizar de forma cuantitativa (puede tomar un valor numérico) o cualitativa (puede tomar un valor de k diferentes clases o categorías).

2.3.1.1. **Deep Learning.**

El Deep Learning es una subclasificación del Machine Learning, este aprendizaje se compone principalmente de tres o más capas neuronales, las mismas que a su vez pretenden simular la capacidad del pensamiento humano para tomar decisiones. Está compuesto por una combinación de entradas, pesos y sesgos, los cuales, permiten realizar funciones como reconocimiento y clasificación.

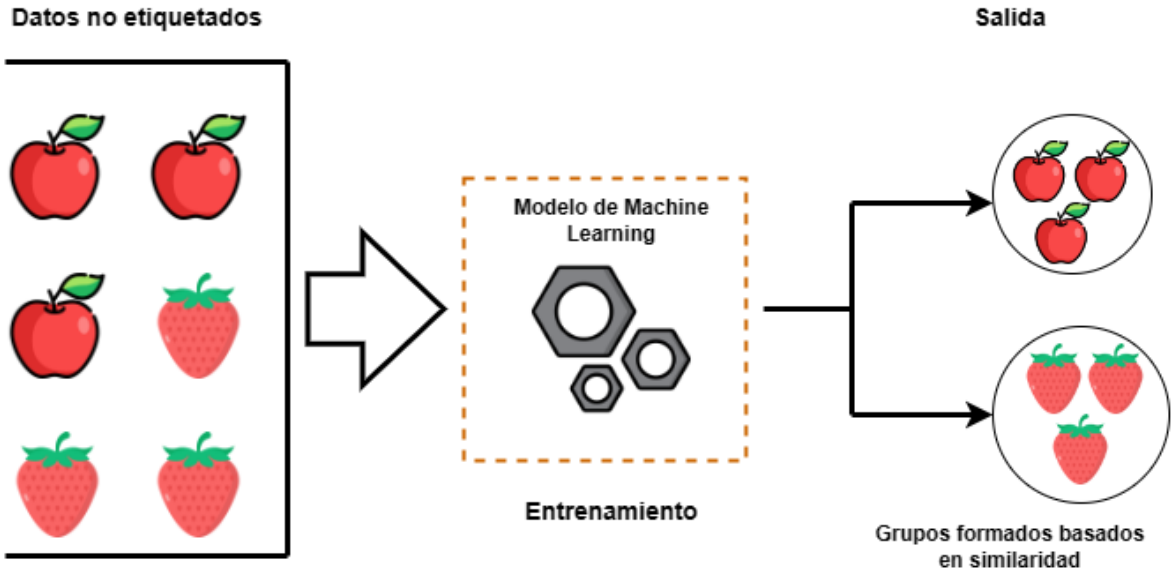
Básicamente, las redes neuronales de Deep Learning utilizan múltiples capas de nodos interconectados, de tal forma que la capa anterior permite la mejora y optimización en el modelo de predicción; el proceso de este cálculo jerárquico a través de la red es denominado propagación directa (IBM, 2020).

2.3.2. Aprendizaje no supervisado

Para este tipo de aprendizaje los datos de entrenamiento no se encuentran etiquetados, además, se pretende realizar este aprendizaje sin un punto central de aprendizaje. La meta de esta clasificación de Machine Learning es la obtención de información relevante acerca de un conjunto de datos, sin basarse en variables de salida conocidas y por medio de la examinación de los datos no etiquetados, visualizar Figura 9. Estas técnicas agregan etiquetas a los datos y se convierten en un tipo de lenguaje supervisado; existen dos categorías principales del aprendizaje no supervisado: agrupamiento y reducción dimensional.

Figura 9

Representación del aprendizaje no supervisado



Nota. En esta imagen se indica la representación del aprendizaje no supervisado interpretado con manzanas y fresas. Adaptado de *Una introducción a los modelos de Machine Learning* (Romero, 2020).

2.3.2.1. Agrupamiento.

Esta técnica exploratoria se basa la organización de información en grupos, sin poseer ningún tipo de modelo previo para su análisis. El objetivo de este proceso es agrupar los datos con características similares en un subconjunto, el cual, a su vez se diferencia de datos de otros grupos.

2.3.2.2. Reducción Dimensional.

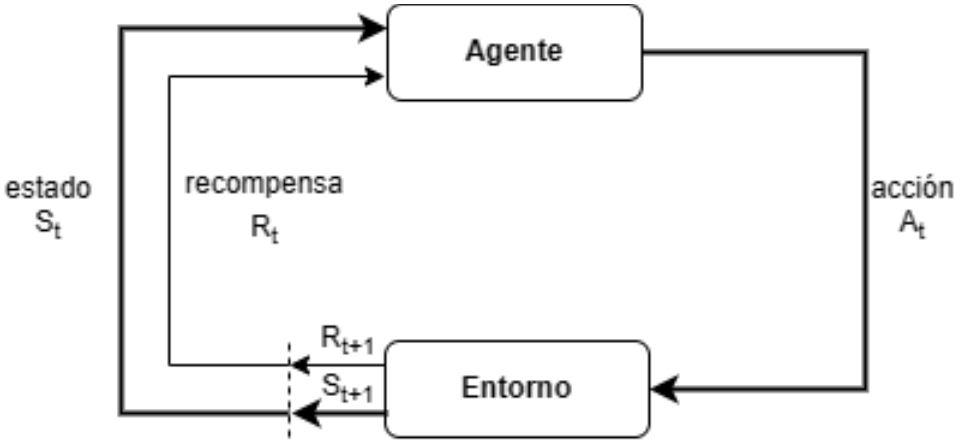
El funcionamiento por medio de reducción lineal se centra en la búsqueda de correlaciones en las características de cada uno datos u objetos del conjunto, estableciendo de esta manera una redundancia de información; este método elimina el "ruido" de los datos, para luego comprimir los mismos datos en un subespacio menor, lo que permite conservar la información más relevante (Romero, 2020).

2.3.3. Aprendizaje por refuerzo

El aprendizaje por refuerzo se construye mediante la guía de un objetivo específico, en donde, un agente realiza la interacción en un entorno desconocido utilizando evaluaciones de prueba y error para obtener una retroalimentación adecuada dentro del entorno, de esta manera se consigue el entrenamiento de los datos que se encuentren en dicho entorno (Naeem et al., 2020), este proceso se puede identificar en la Figura 10.

Figura 10

Representación del aprendizaje por refuerzo



Nota. En figura se indica la representación del aprendizaje por refuerzo, interpretado mediante un simple diagrama. Adaptado de *Reinforcement Learning: An Introduction Second edition, in progress* (Sutton & Barto, 2015).

Los inconvenientes relacionados a este tipo de Machine Learning se centran principalmente en el nivel de retroalimentación que se consigue por cada interacción realizada con cada uno de los datos del entorno, esto debido a que, no todos valores del conjunto tendrán características similares.

El Machine Learning permite el procesamiento de conjuntos de datos para la formación y la utilización de modelos matemáticos o estadísticos en una máquina, esto ofrece la capacidad de reconocer e identificar patrones para la toma de decisiones, este tipo de técnicas son utilizadas generalmente para la clasificación de datos y la automatización de procesos. De forma específica mediante la implementación de algoritmos de Machine Learning se puede utilizar para la categorización de datos como: audio, video, imágenes, entre otros tipos de datos. La visión artificial o visión por computadora utiliza esta tecnología como base para su funcionamiento.

2.4. Visión Artificial

La visión artificial se define como un campo de estudio, el cual tiene como objetivo el desarrollo de métodos para comprender, interpretar e identificar objetos que son visualmente perceptibles por medio de una unidad de procesamiento de datos; este proceso se lleva cabo mediante inteligencia artificial y por medio de modelos de aprendizaje profundo (Deep Learning), los cuales están basados en la capacidad de humana para la toma de decisiones (Lawaniya, 2020).

Uno de los problemas de visión artificial son el reconocimiento de objetos, ya que por medio del entrenamiento a partir de datos preexistentes se logra obtener un modelo que permite el reconocimiento de objetos, sin embargo, se presentan situaciones como: deformaciones, variación de la apariencia del objeto, variación de escala, borrosidad, etc. que alteran la eficiencia del reconocimiento.

En general la visión artificial básicamente es utilizada para el procesamiento de imágenes y la interpretación de información visual, de esta manera automatizando ciertos procesos, además, este tipo de tecnología tiene diversas aplicaciones como: reconocimiento facial, identificación de objetos, seguridad, entre otros. Con el fin de implementar este tipo de procesos en dispositivos con baja capacidad de procesamiento computacional, se ha desarrollado alternativas como TinyML, la

cual permite la ejecución de modelos estadísticos de predicción sobre dispositivos de bajo consumo, microcontroladores y dispositivos portátiles, esto mediante la optimización de los mismos modelos.

2.5. TinyML

Es un campo de estudio en Machine Learning y sistemas integrados que tiene como objetivo adaptar modelos de aprendizaje en dispositivos de baja potencia para realizar inferencias, además, obteniendo un rendimiento adecuado sin desbordar el consumo de energía; generalmente suele ser utilizado en microcontroladores (Osman et al., 2021).

Las principales características que posee TinyML son:

- Baja latencia: Debido a que no necesita una conexión con un servidor, el procesamiento puede ser ejecutado de forma nativa, obteniendo una latencia baja en su salida
- Bajo consumo: Puesto que su enfoque se centra en el funcionamiento en dispositivos con capacidad de procesamiento computacional bajo, el consumo de energía no es excesivo
- Privacidad: Al realizarse el procesamiento de los datos dentro del mismo dispositivo y no tener conectividad con un servidor, los datos no son compartidos.

Los dispositivos que funcionan con este tipo de tecnología mayormente son implementados en entornos de IoT (Internet of Things); este tipo de entornos se basan principalmente en una arquitectura que está compuesta por diferentes capas o bloques, que cumplen una función específica dentro de la arquitectura.

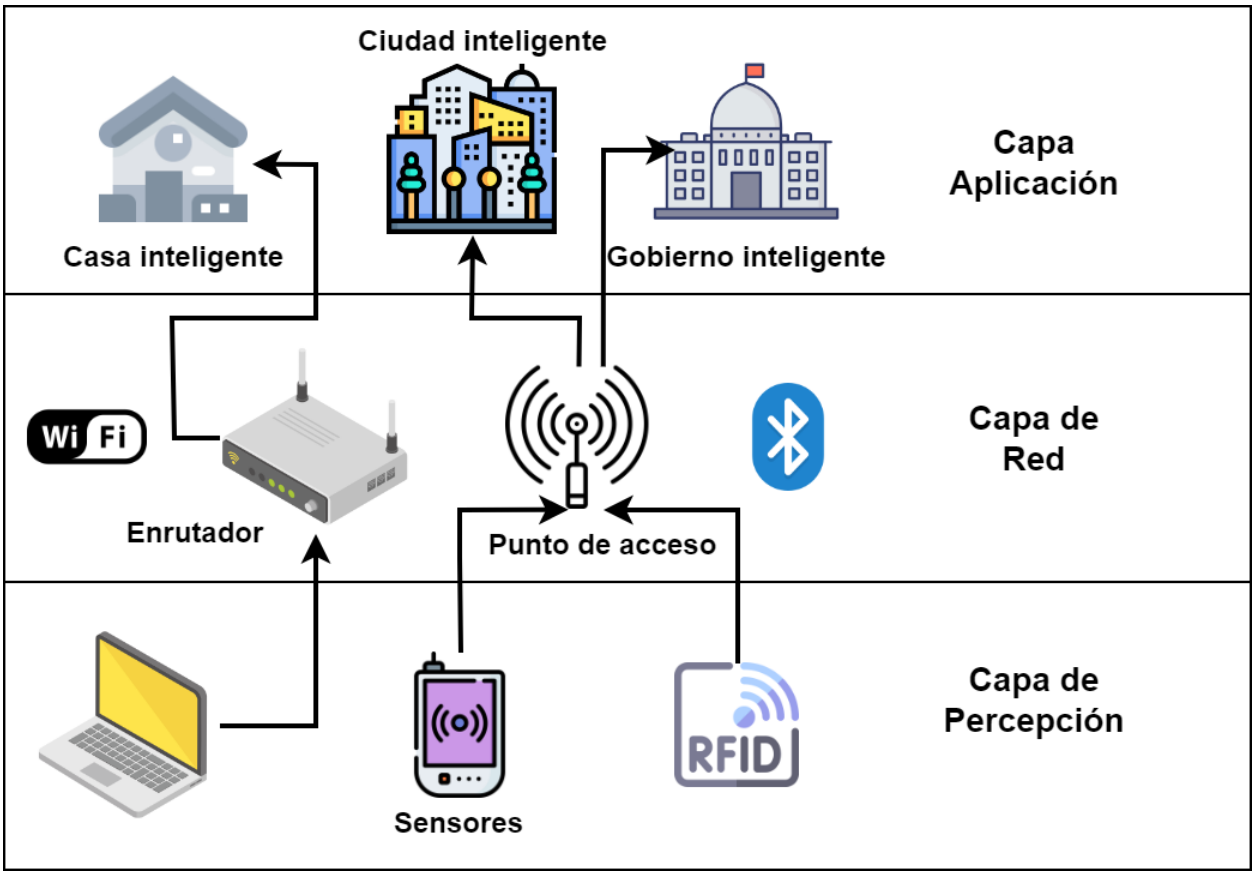
2.6. Arquitectura IoT

IoT (Internet of Things), es el conjunto de dispositivos y servicios tecnológicos interconectados entre sí, para cumplir un objetivo en común en diferentes áreas y aplicaciones. Con el fin de estructurar un sistema que se base en esta tecnología se requiere de una arquitectura,

la cual generalmente es distribuida en capas: Capa de Percepción, Capa de Red y Capa Aplicación, como se indica en la Figura 11.

Figura 11

Arquitectura IoT



Nota. En figura se indica la representación del aprendizaje por refuerzo, interpretado mediante un simple diagrama. Adaptado de *Internet of Things (IoT) Security: Current Status, Challenges and Countermeasures* (Yousuf et al., 2015).

2.6.1. Capa de percepción

En esta capa se encuentran sensores que permiten la adquisición de datos dentro de un entorno, para posteriormente procesar la información y enviar la capa superior.

2.6.2. Capa de red

La capa de red cumple la función de enrutar y transmitir los datos adquiridos en la capa de percepción a distintos dispositivos o a plataformas IoT mediante la utilización de diferentes tecnologías que permitan una conectividad, como: Wifi, Bluetooth, LTE, Zigbee, etc.

2.6.3. Capa aplicación

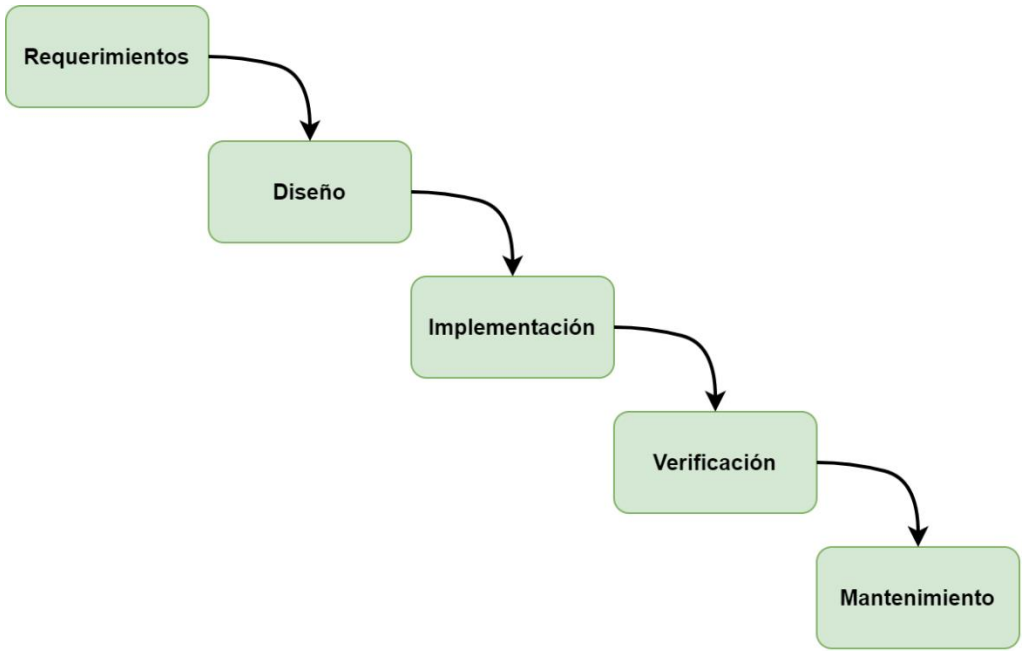
La capa aplicación tiene la función de mostrar los datos al usuario de una forma estructurada y entendible mediante el uso de interfaces o aplicaciones, además, en algunas ocasiones con la posibilidad de interactuar con los dispositivos conectados al sistema.

Capitulo III

En este capítulo se detallará acerca de la estrategia de trabajo que se llevó a cabo, para el diseño y el desarrollo de este proyecto. La metodología que se ha utilizado se divide, en una serie bloques que tiene una secuencia y flujo de trabajo lineal. Este modelo en cascada permite desarrollar proyectos de tal manera que, se puede establecer un control sobre cada una de las fases del trabajo. En la Figura 12 se puede identificar el proceso que se lleva a cabo durante esta metodología.

Figura 12

Diagrama de la metodología en cascada



Nota. Metodología en cascada para la ejecución del proyecto. Adaptado de *Metodologías y procesos de análisis de software* (Villalba, 2022).

- **Requerimientos:** En esta etapa se identifica la problemática que se va a abordar, de la misma manera se presenta la justificación, objetivos y el alcance que se obtendrá con la realización del proyecto. También, dentro de este apartado se desarrolla el estado del arte

correspondiente los temas tratados en el proyecto, también a posterior estableciendo los diferentes requerimientos necesarios para el funcionamiento del sistema.

- **Diseño:** Una vez establecidos todos los requerimientos necesarios, se plantea los componentes electrónicos y los algoritmos que formarán parte del prototipo para el monitoreo del grado de madurez de las frutas.
- **Implementación:** Cuando se finalice el diseño del sistema monitoreo, se procede con la implementación del hardware y software para la construcción, del prototipo que permite medir el estado de madurez de las frutas.
- **Verificación (Pruebas):** La construcción del sistema de monitoreo da paso a la realización de diferentes pruebas en varios escenarios que permitirán comprobar la eficiencia de este sistema bajo ciertos parámetros.
- **Mantenimiento:** Para finalizar, se ajusta todos los parámetros para un funcionamiento a largo plazo, en donde, todo el proceso sea llevado a cabo de forma autónoma, solucionando la problemática que se ha especificado anteriormente.

3.1. Diseño del sistema

En este apartado se indica los aspectos principales del sistema, en donde, se tiene como base el cumplimiento de los objetivos establecidos para la elaboración del diseño. Utilizando la metodología en cascada, la cual tiene varias fases secuenciales para el correcto desarrollo de este diseño.

3.1.1. Requerimientos

En esta sección se determina las especificaciones del sistema, en donde se han establecido tres tipos de requerimientos: requerimientos de stakeholders, requerimientos de sistema y

requerimientos de arquitectura, los mismos a los cuales se les ha asignado una abreviatura como se indica en la Tabla 2, esto con el fin de obtener una mayor facilidad del manejo y organización de cada uno de los requerimientos.

Tabla 2

Nomenclaturas

NOMENCLATURA	
Requerimientos	Abreviatura
Stakeholders	RSSt
Funcionamiento del sistema	RSFs
Arquitectura	RARa

Nota. La tabla indica las diferentes nomenclaturas que se utilizará para denominar ciertos requerimientos. Fuente: Autoría propia

3.1.1.1. Determinación de Stakeholders.

Los Stakeholders hacen referencia a personas o entidades que tengan un interés sobre el diseño e implementación del proyecto, además, pueden participar dentro del mismo proyecto ya sea de forma directa o indirecta, con la finalidad de validar y comprobar el funcionamiento adecuado del sistema de monitoreo. En la Tabla 3 se indica un listado de Stakeholders para este proyecto.

Tabla 3

Requerimientos de Stakeholders

REQUERIMIENTOS DE STAKEHOLDERS	
Universidad Técnica del Norte	Entidad de respaldo
Msc. Edgar Maya	Docente tutor

Msc. Jaime Michilena	Asesor
Sr. Carlos Erazo	Desarrollador del proyecto

Nota. La tabla indica las personas interesadas en el proyecto. Fuente: Autoría propia

3.1.1.2. Requerimientos de Stakeholders.

Tomando en cuenta algunos aspectos para el desarrollo de este proyecto se plantea la Tabla 4 en donde, se ha establecido los requerimientos que debe cumplir el sistema, también procurando solventar las necesidades del usuario.

Tabla 4

Requerimientos del sistema y del usuario

REQUERIMIENTOS DE STAKEHOLDERS (RSSt)			
N°	Requerimientos	Prioridad	
	Requerimientos del Sistema	Alta	Media
RSSt 1	Sistema de consumo energético bajo		X
RSSt 2	Adquirir imágenes en una resolución y nitidez adecuada	X	
RSSt 3	Automatización del sistema	X	
RSSt 4	Operación permanente	X	
	Requerimientos del Usuario		
RSSt 5	Sistema pequeño, que no ocupe mucho espacio		X
RSSt 6	Notificaciones de alerta	X	
RSSt 7	Sistema de fácil uso para el usuario	X	
RSSt 8	Presentación de resultados entendibles para el usuario	X	

Nota. La tabla indica los requerimientos para el sistema y el usuario, los mismos que deben ser principalmente solventados. Fuente: Autoría propia

3.1.1.3. Requerimientos de funcionamiento del sistema.

Los requerimientos de funcionamiento hacen referencia a un conjunto de características y funcionalidades el cual, el sistema debe ser capaz de cumplir para solventar las necesidades del usuario, dentro de este ámbito se pueden encontrar otros tipos de requerimientos como: requerimientos de uso, requerimientos de interfaces, requerimientos de desempeño, esto se puede visualizar de una forma más detallada en la Tabla 5.

Tabla 5

Requerimientos de funcionamiento

REQUERIMIENTOS DE FUNCIONAMIENTO (RSFt)			
N°	Requerimientos	Prioridad	
	Requerimientos de Uso	Alta	Media
RSFt 1	El sistema debe ser alimentado mediante una fuente		X
RSFt 2	Facilidad de uso del sistema para el usuario	X	
RSFt 3	Toma datos de forma rápida		X
RSFt 4	Interruptor para encendido/apagado		X
	Requerimientos de Interfaces		
RSFt 5	Puerto con conexión para cámara de alta resolución	X	
RSFt 6	Cantidad de pines digitales mayores a 4	X	
RSFt 7	Sistema con conexión a internet	X	
	Requerimientos de Desempeño		
RSFt 8	Visualización de la información resultante	X	

RSFt 9	Escalabilidad del sistema		X
RSFt 10	Capacidad para el envío de datos	X	
RSFt 11	El sistema debe tener una tasa de disponibilidad alta	X	
	Requerimientos físicos		
RSFt 12	La cámara debe tener la capacidad de obtener las imágenes de las frutas de forma nítida	X	
RSFt 13	La cámara debe tener un tamaño pequeño		X
RSFt 14	La cámara debe tener la capacidad de tener un enfoque modificable	X	

Nota. La tabla indica los requerimientos de uso, interfaces, desempeño y físicos que debe tener el sistema de monitoreo. Fuente: Autoría propia

3.1.1.4. Requerimientos de la arquitectura.

Este tipo de requerimientos consiste en el detalle acerca de los componentes de hardware y software, tomando en cuenta la funcionalidad que pueden aportar al sistema. La lista de requerimientos se presenta en la Tabla 6.

Tabla 6

Requerimientos de la arquitectura

REQUERIMIENTOS DE ARQUITECTURA (RARa)			
N°	Requerimientos	Prioridad	
	Requerimientos de Diseño	Alta	Media
RARa 1	El sistema no interfiere con el proceso de maduración	X	
RARa 2	Ubicación de la cámara en un lugar estratégico	X	

RARa 3	Escalabilidad		X
RARa 4	Sistema de bajo costo		X
	Requerimientos Lógicos		
RARa 5	Entradas y salidas digitales	X	
RARa 6	Comunicación con todo el sistema	X	
RARa 7	Sistema con acceso a internet para el envío de datos	X	
	Requerimientos de Software		
RARa 8	Leguaje de programación de código abierto	X	
RARa 9	Sistema operativo de uso libre compatible con el sistema	X	
RARa 10	Sistema operativo capaz de soportar lenguajes de programación	X	
RAR 11	Rendimiento óptimo del sistema operativo	X	
RARa 12	Compatibilidad con el manejo de cámara de alta resolución	X	
	Requerimientos de Hardware		
RARa 13	Sistema embebido con puerto de conexión para una cámara	X	
RARa 14	Sistema embebido con un procesador y memoria de alta capacidad	X	
RARa 15	La fuente de energía debe alimentar a todo el sistema	X	
RARa 16	Cantidad adecuada para pines digitales		X
RARa 17	La cámara debe ser compatible con el sistema embebido	X	

Nota. La tabla indica los requerimientos de hardware, software, diseño y lógicos que debe tener el sistema de monitoreo. Fuente: Autoría propia

3.1.2. Selección de Hardware y Software

Para la selección de hardware como de software se establecen comparativas en base a los requerimientos para el desarrollo del sistema, los mismos que fueron planteados anteriormente. Cabe resaltar que se toma en consideración los requerimientos más críticos que permitan el funcionamiento adecuado del sistema embebido. Con el fin de facilitar el análisis de cada uno de los requerimientos, se ha estructurado la Tabla 7, en donde se indica los valores que se utilizará para definir el cumplimiento del requerimiento.

Tabla 7

Descripción de valores

DESCRIPCIÓN DE VALORES	
Si cumple	1
No cumple	0

Nota. La tabla explica la nomenclatura que tomará el valor de “Si cumple” y “No cumple”.

Fuente: Autoría propia

3.1.2.1. Elección de Hardware.

Para la elección de hardware se realiza un cotejo de dispositivos físicos, en consideración de los Requerimientos de Arquitectura y de Sistema, en donde se fijan principalmente las características que deben poseer estos dispositivos, tanto para el sistema embebido como para el sensor (cámara).

3.1.2.1.1. Sistema Embebido.

En este caso para la elección de la placa que utilizará el sistema embebido se ha optado por el empleo de una computadora de placa única (SBC), debido a que, se requiere este tipo de

dispositivos que ofrecen una mayor cantidad de funcionalidad frente otros dispositivos como los microcontroladores, además de tener una mayor potencia de procesamiento y versatilidad. La comparativa de este tipo de placas en base los requerimientos necesarios para el sistema se indican en la Tabla 8.

Tabla 8

Selección de placa SBC

Selección de placa SBC							
Hardware	Requerimiento						Valoración
	RSFt 7	RSFt 10	RARa 5	RARa 6	RARa 12	RARa 14	Total
Raspberry Pi Zero	0	1	1	1	1	0	4
Raspberry Compute Module 4	1	1	1	1	1	0	5
Raspberry Pi 4 B+	1	1	1	1	1	1	6
1 Cumple							
0 No cumple							
Selección: Raspberry Pi 4 Model B+							

Nota. En la tabla se evalúa las diferentes placas SCB de acuerdo a los requerimientos que se ha establecido. Fuente: Autoría propia

Como se indica en la tabla anterior se ha realizado el análisis de las placas SBC con características similares como posibles opciones, para la utilización en el sistema embebido. Sin embargo, en la misma tabla se indica una valoración de los requerimientos evaluados de tal manera que, la placa Raspberry Pi model B+ cumple con todos los requerimientos para el sistema, de tal forma que se selecciona dicho dispositivo.

Como parte adicional se indica las características específicas del dispositivo que se ha seleccionado para el sistema embebido, esto se puede visualizar en la Tabla 9.

Tabla 9

Características técnicas del Rasberry Pi 4

Parámetros	Descripción
Procesador	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
Memoria RAM	4GB LPDDR4-3200
Puertos	2 puertos USB 3.0 2 puertos USB 2.0 2 puertos × micro-HDMI (Soporta hasta resolución 4k) 2 puertos de cámara MIPI CSI
Alimentación	5V/3A USB-C 5V/3A pines GPIO
Conectividad	IEEE 802.11ac Wireless Bluetooth 5.0 Puerto Gigabit Ethernet

Nota. En la tabla se describe las características técnicas de la Rasberry Pi 4 model B. Fuente: (Raspberry, 2021)

3.1.2.1.2. Sensor.

El sensor forma parte principal del sistema embebido, puesto que, a través de este componente electrónico se podrá capturar las imágenes acerca del estado de madurez de las frutas, por lo cual, se debe elegir un sensor que se acople y cumpla con los requisitos para este sistema. En este caso se realiza una comparativa de cámaras compatibles con la placa SBC que se ha seleccionado anteriormente, esto se detalla en la Tabla 10.

Tabla 10

Selección del sensor

Selección del Sensor					
Hardware	Requerimiento				Valoración
	RSFt 12	RSFt 13	RSFt 14	RARa 17	Total
Raspberry Pi High-Quality Camera (Waveshare)	1	1	0	1	3
Raspberry Pi Camera v2 (Adafruit)	1	1	0	1	3
Arducam Mini Raspberry Pi HQ Camera	1	1	1	1	4
1 Cumple					
0 No cumple					
Selección: Arducam Mini Raspberry Pi HQ Camera					

Nota. En la tabla se evalúa los diferentes sensores para actuar sobre el sistema embebido. Fuente: Autoría propia

En base a la tabla que se ha presentado anteriormente se selecciona la "Arducam Mini Raspberry Pi HQ Camera", como cámara para el sistema embebido de monitoreo, debido al cumplimiento de requerimientos que presenta, frente a otro tipo de cámaras, resaltando como características principales, la nitidez de la captura de imágenes, su capacidad para la modificación del enfoque y su reducido tamaño. Para mayor detalle de las características de esta cámara se presenta la Tabla 11, en donde se definen todas sus características técnicas.

Tabla 11

Características técnicas del sensor de ArduCam

Parámetros	Descripción
Sensor de cámara	Sony 12.3MP IMx477
Formato óptico	1/2.3" (diagonal 7.857mm)
Resolución	12.3MP
Pixeles activos	4056(H) x 3040(V)
Tamaño de pixel	1.55um x 1.55um
Velocidad de fotogramas	1080p30, 720p60 y 640 x 480p60/90
Tipo de obturador	Persiana enrollable
Interfaz	MIPI CSI-2
Montura de lente predeterminada	Montura M12
Parámetros de lente	F / No: 2.8, Distancia Focal: 3.9mm, lente M12 de baja distorsión

Tipo de enfoque	Manual
Campo de visión (FoV)	75°
Tamaño de la placa	24mm x 25 mm
Corriente máxima	300mA

Nota. En esta tabla se especifica las características técnicas del sensor de ArduCam. Fuente: (ArduCam, 2021)

3.1.2.2. Elección de Software.

Para la elección del software se debe tomar en consideración dos puntos: el sistema operativo con el que funcionará la placa SBC y el lenguaje de programación para el funcionamiento del sistema embebido. Cabe recalcar que, estos dos parámetros deben ser compatibles entre sí, para un óptimo funcionamiento del sistema.

3.1.2.2.1. Selección de sistema operativo.

Para definir el tipo de sistema operativo va a funcionar sobre la placa SBC, se debe tomar en cuenta requisitos como: compatibilidad con la placa SBC, capacidad para guardar y gestionar archivos y requerimientos de la arquitectura. En la Tabla 12 se realiza un contraste entre distintos sistemas operativos en base a los requerimientos de la arquitectura que debe cumplir.

Tabla 12

Selección del Sistema operativo

Selección del Sistema operativo				
Hardware	Requerimiento			Valoración
	RARa 9	RARa 10	RARa 11	Total
Arch Linux ARM	1	1	1	3
QNX	0	1	1	2

Raspberry Pi OS	1	1	1	3

1 Cumple

0 No cumple

Selección: Raspberry Pi OS

Nota. En esta tabla define el mejor sistema operativo para el sistema de monitoreo. Fuente: Autoría propia

En la tabla anterior se ha indicado el uso de Raspberry Pi OS para este proyecto, sin embargo, hay que resaltar que a pesar de que tiene el mismo puntaje de con respecto Arch Linux ARM, Raspberry Pi OS características adicionales como: una amplia compatibilidad, soporte y documentación, y mejor compatibilidad de hardware.

3.1.2.2.2. Lenguaje de programación.

Ahora para este apartado, se debe elegir un tipo de lenguaje de programación que permita el manejo de todos los componentes del sistema embebido, además, de soportar la implementación de modelos de aprendizaje, tener versatilidad sobre el uso de librerías y eficiencia a la hora del funcionamiento del mismo sistema. En la Tabla 13 se evalúa diferentes tipos de lenguaje de programación, en base a los requerimientos que debe cumplir dentro del sistema.

Tabla 13

Selección del Lenguaje de programación

Selección del Lenguaje de programación			
Hardware	Requerimiento		Valoración
	RARa 8	RARa 12	Total
JavaScript	1	1	2
Python	1	1	2

Perl	1	1	2

1 Cumple

0 No cumple

Selección: Python

Nota. En esta tabla define el mejor lenguaje de programación que se acople al sistema de monitoreo. Fuente: Autoría propia

Como se ha identificado anteriormente se ha seleccionado el lenguaje de programación Python para realizar este proyecto. A pesar de que, los otros lenguajes de programación tienen la misma valoración dentro de la tabla, se ha elegido este lenguaje programación por ciertas ventajas de usabilidad en el entorno de Raspberry, además de que, ofrece una sintaxis y legibilidad clara al momento del desarrollo del código.

3.2. Obtención de imágenes de plátanos y fresas

Una fase importante para obtener las muestras necesarias para el entrenamiento del modelo de clasificación es la adquisición de las imágenes correspondientes a plátanos y fresas, en sus diferentes fases de maduración. Este tipo de frutas de acuerdo a su grado de madurez presenta un cambio de tonalidad en su color, o también puede mostrar un cambio de color sobre la misma; las imágenes que se indican en los siguientes apartados fueron tomados de datasets previamente conformados. Además, con la finalidad de mejorar la precisión del modelo entrado se ha capturado imágenes de plátanos y fresas en sus diferentes etapas de maduración, utilizando un smartphone, estas fotografías se realizan sobre una superficie de color blanco como se muestra en la Figura 13.

Figura 13

Captura de las imágenes de plátanos y fresas

Nota. Autoría propia

3.2.1. Plátanos

La base de datos de imágenes de plátanos de se ha obtenido utilizando a la plataforma de Roboflow, la cual está enfocada al tema de visión por computadora, dentro de esta misma plataforma se pueden encontrar datasets acerca de diversos objetos, e incluso herramientas para el entrenamiento y despliegue de modelos de visión artificial. La base de datos que seleccionada que se ha extraído de esta plataforma (https://universe.roboflow.com/dark-horse/musa.ai/browse?queryText=&pageSize=50&startingIndex=0&browseQuery=true) la cual contienen 1500 muestras de imágenes de plátanos. Cabe resaltar que estas imágenes se encuentran clasificadas en seis diferentes estados de madurez: inmaduro, fresco inmaduro, fresco, maduro, maduración excesiva y podrido, dentro de este mismo grupo de imágenes se ha integrado las

nuevas fotos que se han tomado anteriormente. Con estas nuevas imágenes se ha aumentado el dataset hasta 4068 muestras.

3.2.1.1. Inmaduro.

En la Figura 14 se puede visualizar una de las muestras correspondientes a la etiqueta “Inmaduro”, en esta primera imagen se pueden resaltar las características del plátano en este estado de madurez. De forma general, se aprecia que la cáscara del plátano tiene un color verde intenso uniforme sobre la mayoría de la fruta, también se puede destacar que posee una textura lisa y ligeramente brillante, además, existe la presencia de pequeñas manchas marrones, los cuales son muy comunes en este tipo de frutas, estos son indicativos muy característicos de esta etapa de inmadurez del plátano.

Figura 14

Plátanos con estado de "inmaduros"

Nota. Autoría propia

3.2.1.2. Fresco inmaduro.

Para el estado de madurez “Fresco inmaduro” presentado en la Figura 15 se observa un cambio ligero de la coloración de la cáscara del plátano en comparación al estado de madurez anterior, la tonalidad de verde de esta fruta ya no es tan intensa y tiende a tener un color más claro. Algunas pequeñas manchas marrones han tomado un color más intenso y tienen un mayor tamaño sobre la fruta. En estado de madurez del plátano aun no es muy recomendable para su consumo, debido a que la fruta no tiene un nivel de maduración adecuada.

Figura 15

Plátanos con estado de "fresco inmaduro"

Nota. Autoría propia

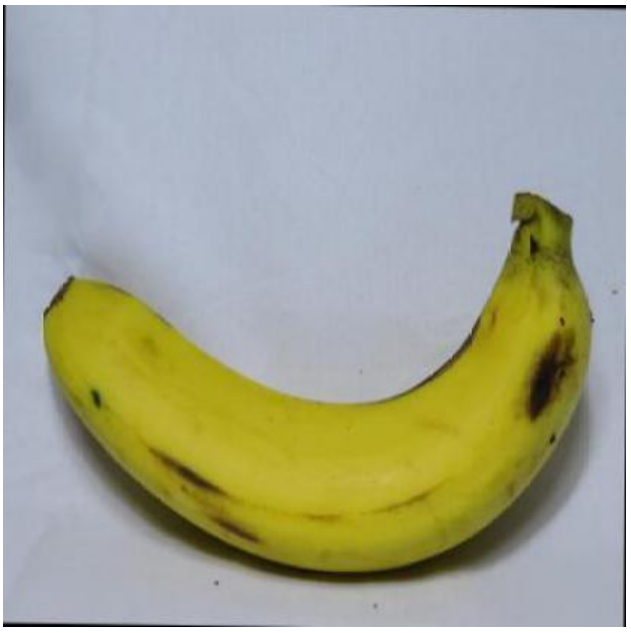
3.2.1.3. Fresco.

El estado de madurez “Fresco”, la cáscara del plátano presenta una tonalidad amarilla clara en la mayor parte de la fruta, cabe resaltar que esta tonalidad que ha cambiado de un verde más claro a un amarillo brillante, indica que el nivel de maduración se ha incrementado de forma significativa. Otro punto que se logra visualizar en la Figura 16 es el incremento de la coloración

de las manchas marrones características del plátano. En esta fase de maduración el producto ya puede ser consumido.

Figura 16

Plátanos con estado de "fresco"



Nota. Autoría propia

3.2.1.4. Maduro.

Para la representación del estado de madurez “Maduro”, se muestra la Figura 17 en donde distingue que la coloración de la cáscara del plátano pasó de un color amarillo claro a un amarrillo más intenso en la mayoría de las zonas de la fruta, igualmente, las manchas marrones presentan un aumento sobre la fruta y una coloración más intensa. Para esta etapa, los plátanos han alcanzado un estado pleno de madurez, además, siendo estas características que se han señalado un claro indicador.

Figura 17

Plátanos con estado de "maduro"

Nota. Autoría propia

3.2.1.5. Maduración excesiva.

Como se indica en la Figura 18 muestra un estado de “maduración excesiva” para los plátanos; las características principales sobre este estado de madurez, es la evolución de un color amarillo intenso a un amarillo más oscuro y opaco, también la disposición de manchas marrones sobre la fruta ha crecido considerablemente, además de que, se tiene una mayor coloración de estas manchas e incluso la aparición de manchas negras. Las cualidades de la fruta en este estado representan un grado de madurez muy avanzado, que en ciertas ocasiones deben ser revisados de forma más detallada para determinar si aún son aptos para consumo.

Figura 18

Plátanos con estado de "maduración excesiva"

Nota. Autoría propia

3.2.1.6. Podrido.

Finalmente para la fase final del plátano se tiene un estado denominado “Podrido” (Figura 19), este producto ya no es apto para consumo, debido a que la fruta ha empezado su descomposición. Las características visuales de este estado, son una tono marron oscuro intenso o incluso negro, tambien se indica la irregularidad de la cáscara con arrugas pronunciadas. En algunos casos, otro indicador significativo de este estado, es la aparición de manchas blancas de forma parcial sobre los plátanos, estas manchas de coloración blanca son hongos comunes caractísticos en una fase de descomposición.

Figura 19

Plátanos con estado de "podrido"

Nota. Autoría propia

3.2.2. Fresas

Al igual que el anterior base de datos, las imágenes se obtienen de diversos datasets de la plataforma Roboflow (https://universe.roboflow.com/object-detection-xvfhw/strawberry-d3pyi), obteniendo de esta manera 625 muestras para las fresas, y sumando un valor de 4693 imágenes para todo el dataset.

3.2.2.1. Inmaduro.

Como primera clasificación se tiene el estado de madurez “Inmaduro”, el cual de forma visual se destaca por un color verde pálido transparente, casi blanquecino como se indica en la

Figura 20, además a simple vista, se distingue los diferentes aquenios (semillas de la fresa) distribuidos sobre toda la fruta. En esta etapa de maduración la fresa mayormente tiende a tener una forma más redonda; con respecto al consumo de esta fruta durante esta fase no es muy recomendada debido no ha alcanzado un nivel adecuado de madurez.

Figura 20

Fresas con un estado de "inmaduro"

Nota. Autoría propia

3.2.2.2. Maduro.

Para el siguiente estado de madurez de las fresas “Maduro” el cual se puede observar en la Figura 21, se tiene que un cambio de coloración sobre la superficie de la fresa, pasando de un color verde pálido a un color rojo intenso, de forma característica las semillas de las fresas permanecen

con el mismo patrón de distribución sobre toda la fruta, sin presentar un cambio de coloración en la misma. Otro punto a destacar es el cambio de forma que toma la fresa en este estado de madurez más avanzado, siendo un poco más alargada en comparación con la anterior fase.

Figura 21

Fresas con un estado de "maduro"

Nota. Autoría propia

3.2.2.3. Maduración excesiva.

Como se puede visualizar en la Figura 22 se puede observar que las fresas tienen un estado de madurez, el cual se ha denominado como “Maduración excesiva”. En donde principalmente, se muestra una variación de color de las fresas, pasando de un color rojo intenso y brillante, a un color rojo oscuro y pálido sin brillo, además, se puede apreciar que la fruta a simple vista posee una menor dureza, es decir, se vuelve más blando.

Figura 22

Fresas con un estado de "maduración excesiva"

Nota. Autoría propia

3.2.2.4. Podrido.

Para esta última fase que toma la fresa representada en la Figura 23, se tiene un estado de descomposición denominado para clasificación como “Podrido”. Las principales características visuales la fruta en este estado son el cambio de colaración de un rojo intenso a un rojo más pálido, además de que, en ciertas ocaciones dependiento del grado de descomposición, en la fresa aparecen pequeñas manchas de una tonalidad marrón. Al igual que los plátanos estas frutas también durante esta etapa de descomposición pueden verse afectados por hongos como el moho, que es su mayoría toma un color blanco sobre la fruta.

Figura 23

Fresas con un estado de "podrido"

Nota. Autoría propia

3.3. Preparación de datos de entrenamiento

La preparación de los datos de entrenamiento hace referencia a la adicción de parámetros, normalización y otros ajustes necesarios antes de pasar por el entrenamiento de la red neuronal. Este tipo de proceso es muy necesario, debido a que se organiza y se estructura de una mejor manera los diferentes tipos de imágenes que se ha obtenido previamente.

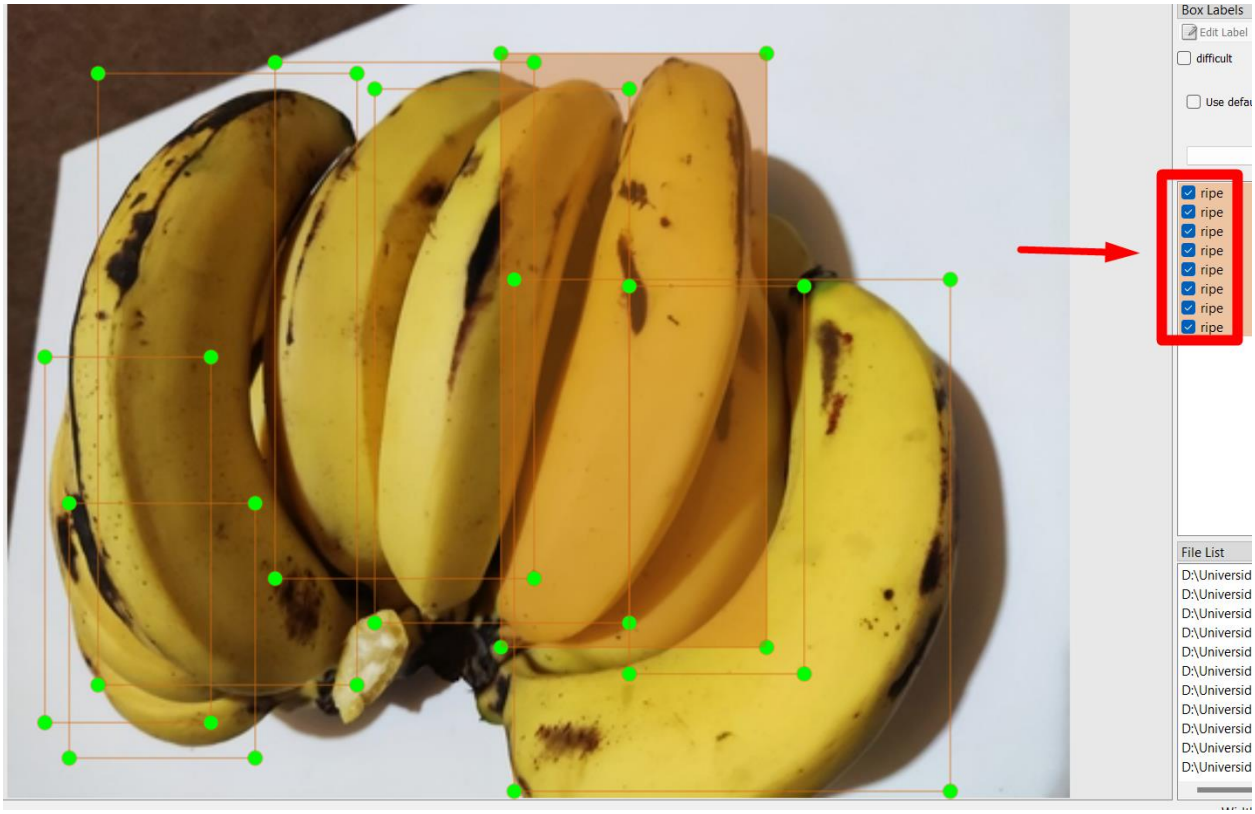
3.3.1. Etiquetado de las imágenes de plátanos

Para el etiquetado de cada una las muestras de los plátanos se ha utilizado un software de código abierto denominado “LabelImg”, el cual se ejecuta a través de Python. Este software permite delimitar varios objetos dentro de una misma imagen, como se indica en la Figura 24 se

puede apreciar que se encierra en un recuadro cada uno de los plátanos, en donde se asigna una respectiva etiqueta dependiendo el estado de madurez que posea. En este caso para los plátanos y sus diferentes estados de madurez se tienen los siguientes nombres para las etiquetas: unripe, freshunripe, freshripe, ripe, overripe y rotten, asociado respectivamente a los distintos estados de madurez: inmaduro, fresco inmaduro, fresco, maduro, maduración excesiva y podrido.

Figura 24

Etiquetado de plátanos



Nota. Autoría propia

3.3.2. Etiquetado de las imágenes de fresas

Al igual que en el apartado anterior el etiquetado de las fresas se ha realizado utilizando el mismo software “LabelImg” como se visualiza en la Figura 25, por el cual se ha realizado la

delimitación de las fresas en cada uno de los estados de madurez. Los nombres que se ha utilizado etiquetar estas frutas son: inmaduro, maduro, demasiadoM y podrido, respectivamente asociados a los grados de madurez: Inmaduro, Maduro, Maduración excesiva y Podrido.

Figura 25

Etiquetado de fresas

Nota. Autoría propia

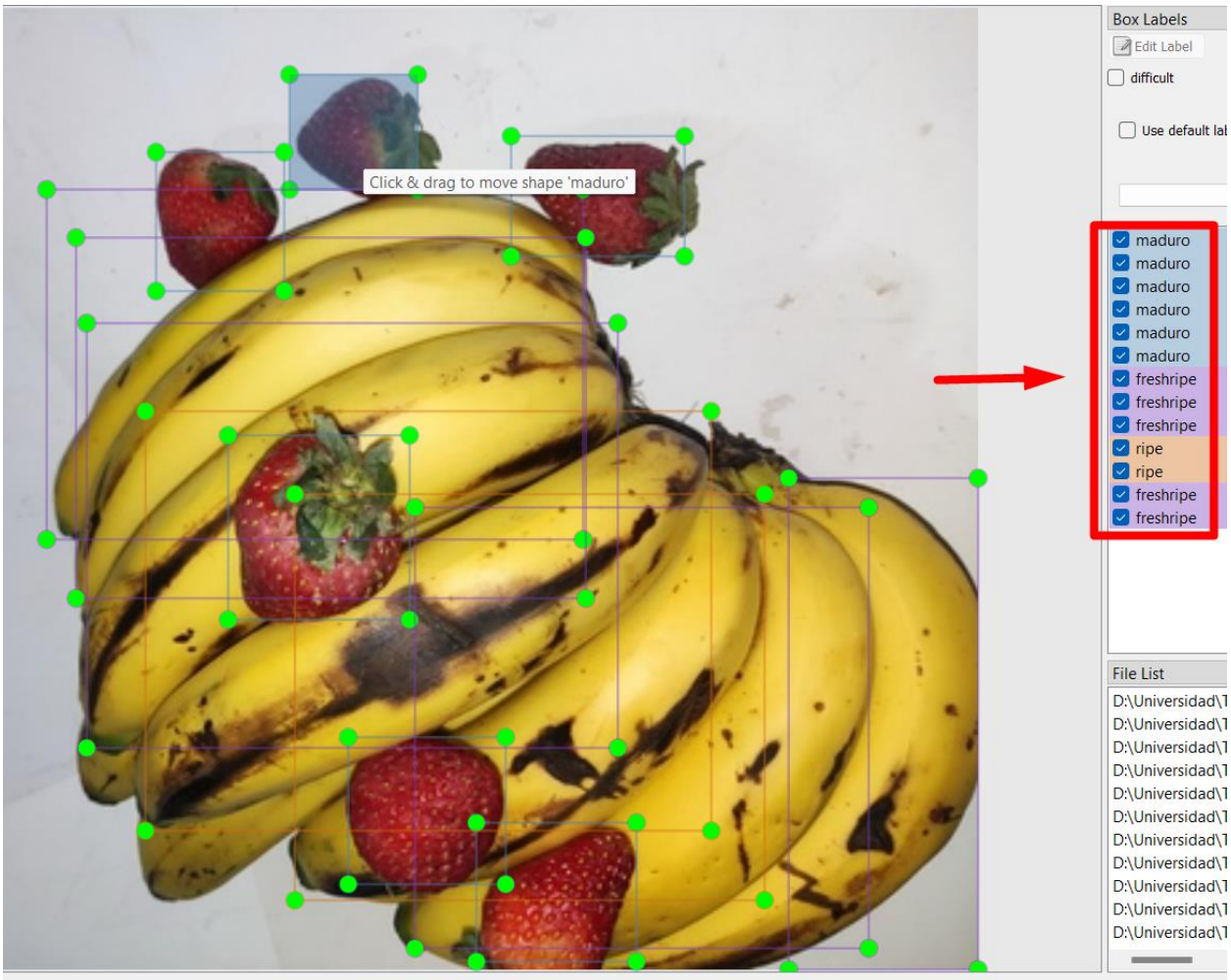
3.3.3. Etiquetado de plátanos y fresas

Además, al existir otras imágenes las cuales contienen fresas y plátanos dentro del mismo plano, se procede a etiquetar estas diferentes frutas, utilizando los nombres que se han definido

anteriormente para cada de los estados de madurez. En la Figura 26 se muestra el etiquetado de los plátanos y las fresas.

Figura 26

Etiquetado de plátanos y fresas



Nota. Autoría propia

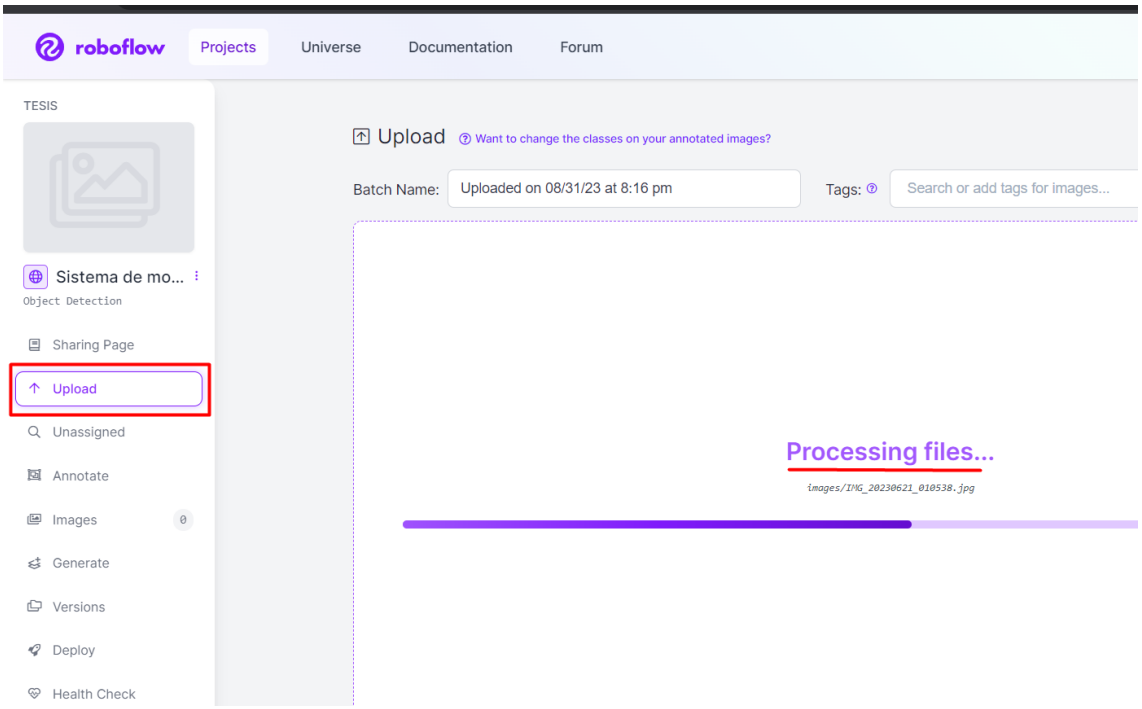
3.3.4. Preprocesamiento de imágenes

Cuando se complete proceso de etiquetado de todo el dataset, a continuación, se utiliza la plataforma de Roboflow para el preprocesamiento de imágenes, inicialmente se debe crear un espacio de trabajo y un nuevo proyecto, después se cargan todos archivos del dataset, es decir, las

imágenes y los archivos XML, los cuales contienen las coordenadas de las cajas delimitadoras, en la Figura 27, se observa este proceso.

Figura 27

Carga de las imágenes en la plataforma de Roboflow

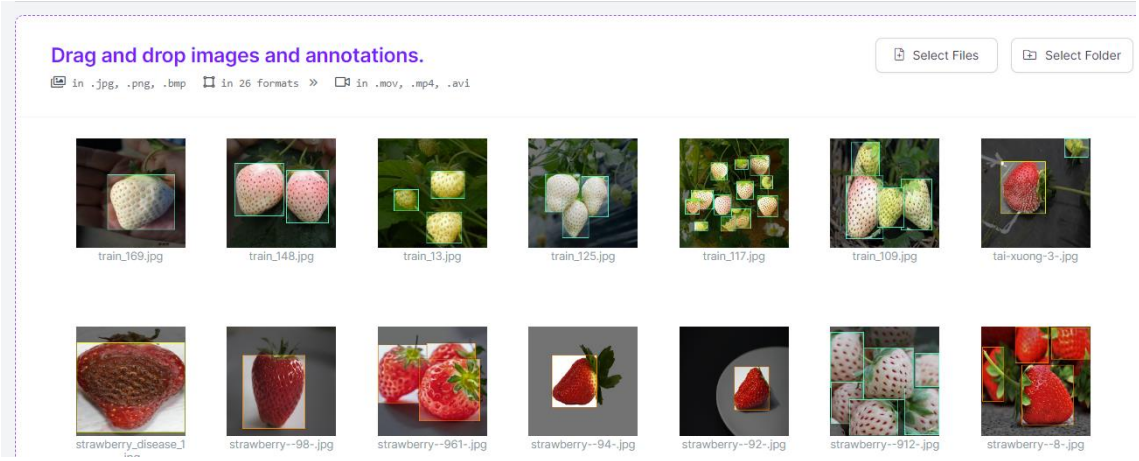


Nota. Autoría propia

Cuando se finalice la carga de los archivos seleccionados dentro del nuevo proyecto, se debe visualizar dentro de esta plataforma todas las imágenes cargadas con sus respectivas etiquetas sobre la misma imagen, observar Figura 28.

Figura 28

Imágenes etiquetadas en la plataforma de Roboflow

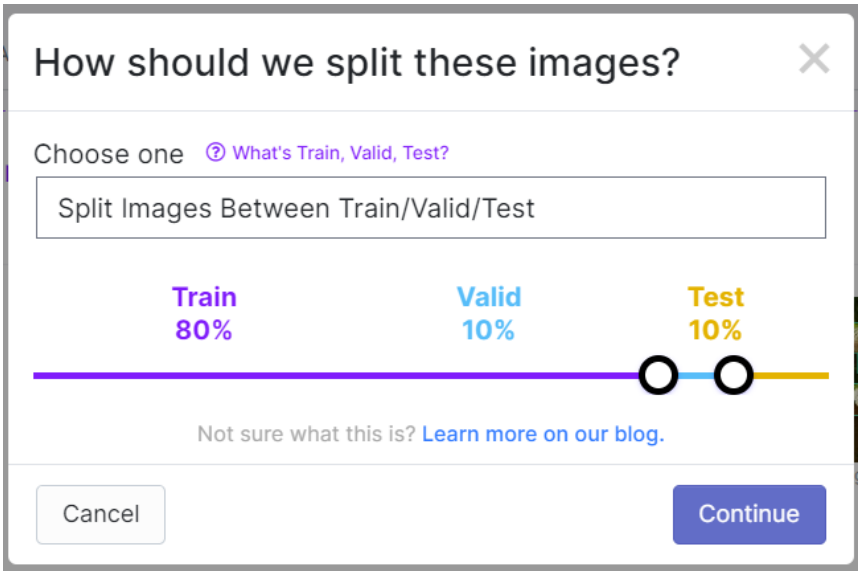


Nota. Autoría propia

Siguiendo con este proceso, se realiza la división del dataset en un 80% para entrenamiento, 10% para validación y 10% para pruebas como se indica en la Figura 29.

Figura 29

Definición del porcentaje de imágenes de entrenamiento, validación y pruebas

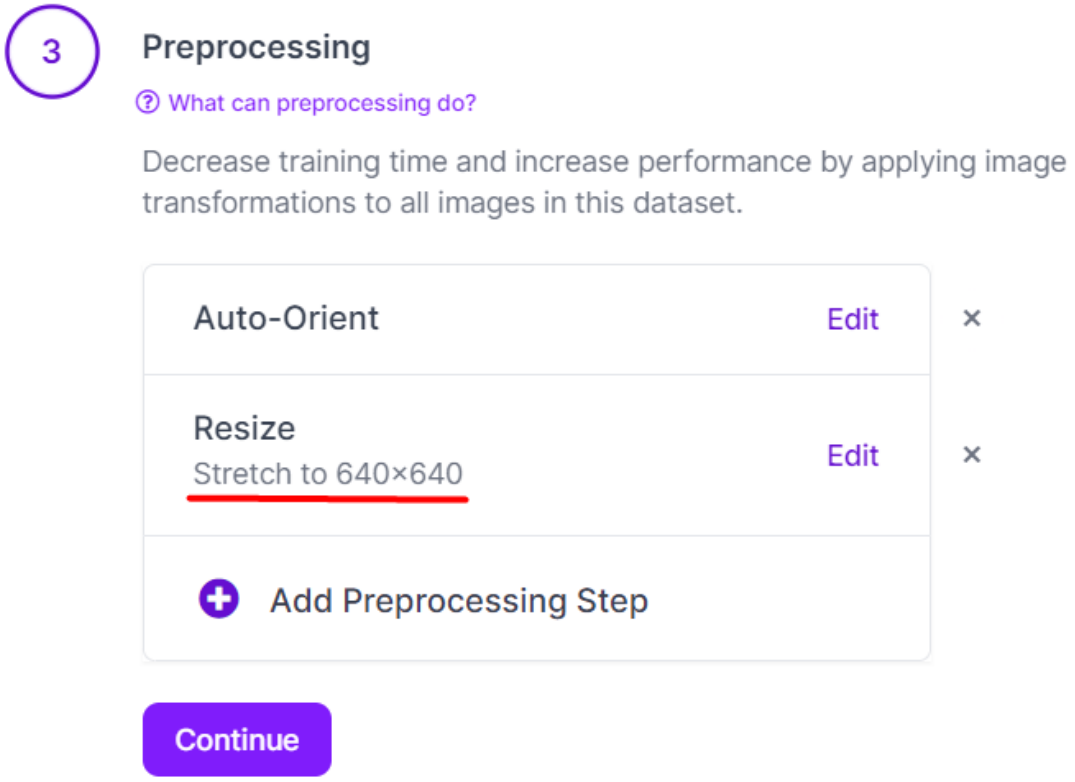


Nota. Autoría propia

Adicional, en esta plataforma se puede realizar ciertos procesos sobre las imágenes como el indicado en la Figura 30, en donde se redimensiona los archivos del dataset a 640x640.

Figura 30

Redimensionamiento de las imagenes

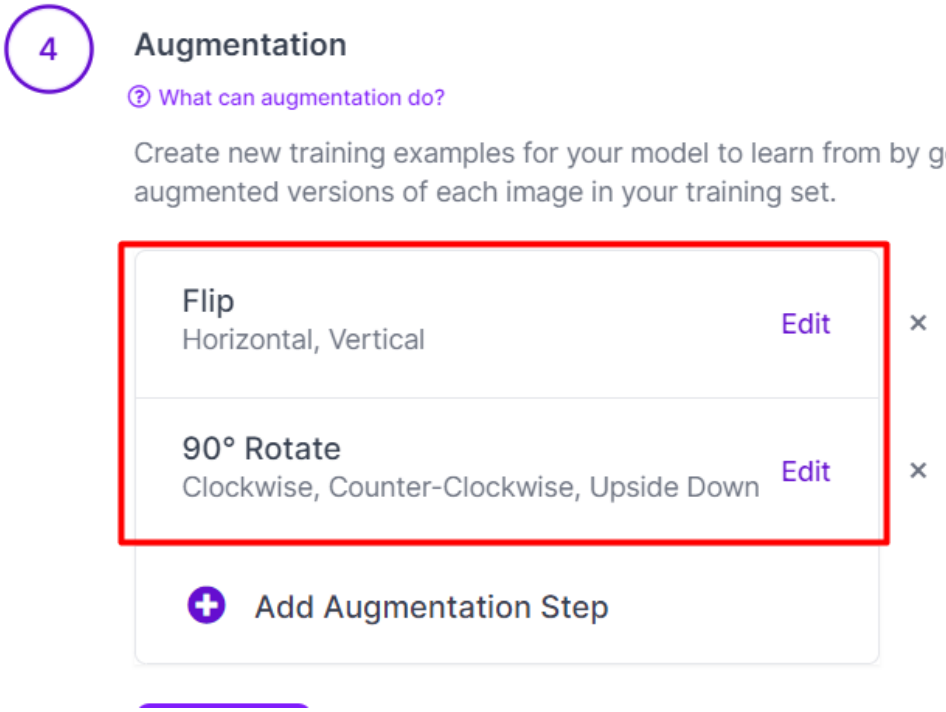


Nota. Autoría propia

En el apartado de argumentación, se realiza dos tipos de técnicas para el incremento de número de muestras, en este caso se utiliza “flip” para la rotación cada 180 grados, y “90° rotate”, esto para realizar un giro cada 90 grado, visualizar Figura 31, esto se realiza en todo el dataset de forma aleatoria.

Figura 31

Rotación de las imágenes del dataset

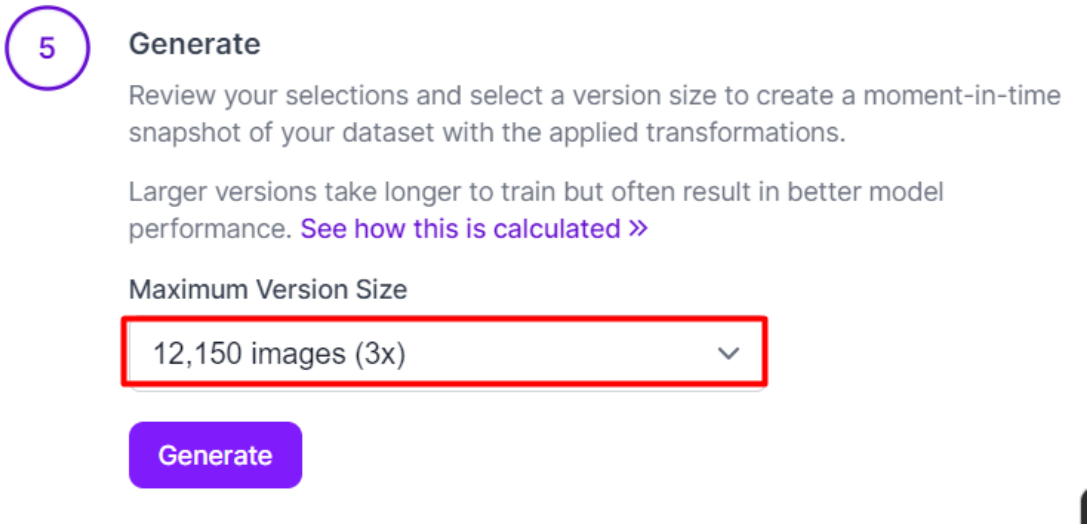


Nota. Autoría propia

Y siguiendo con el preprocesamiento de las imágenes, se puede genera el nuevo dataset como se indica en la Figura 32, incluyendo todos los cambios y adiciones que se han descrito anteriormente; la nueva versión de la base de datos tendrá un total de 12150 imágenes. Y para finalizar se exporta los archivos generados en YOLO Pytorch, a partir de esta exportación se generará un código con una clave para la importación dentro del espacio de entrenamiento.

Figura 32

Generación del aumento de imágenes



Nota. Autoría propia

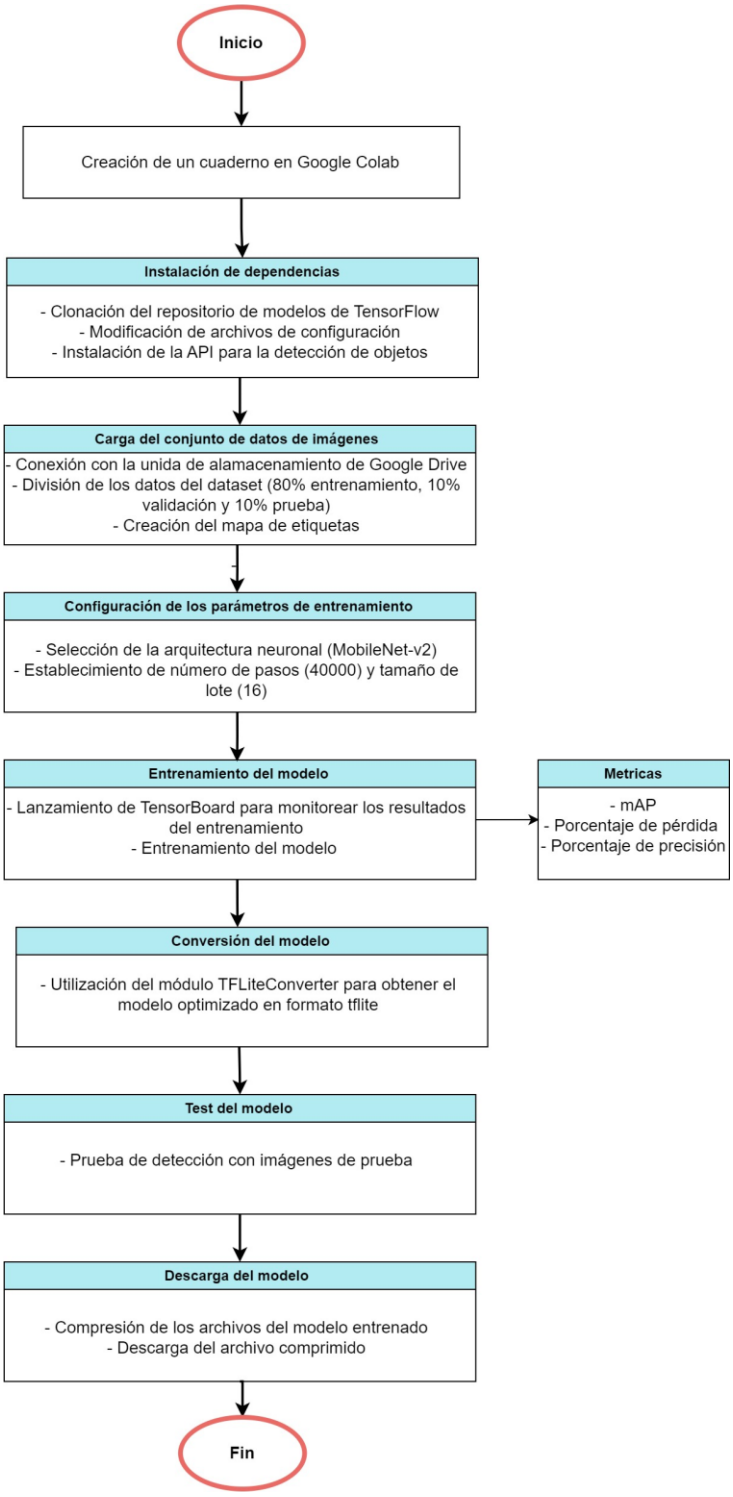
3.1. Diagrama de flujo

3.1.1. Flujograma del proceso de entrenamiento del modelo (MobileNet-v2)

Para el entrenamiento del modelo, se utilizado la plataforma de Colaboratory de Google, en la cual, se realiza todos los procesos de convolución para obtener un modelo el cual contiene las características de cada una de las 10 clasificaciones (inmaduro – plátano, fresco inmaduro – plátano, fresco – plátano, maduro – plátano, maduración excesiva – plátano, podrido – plátano, inmaduro – fresa, maduro – fresa, maduración excesiva – fresa, podrido – fresa). En la Figura 33 se indica todo el proceso de manera progresiva que se ha realizado para llevar a cada este procedimiento.

Figura 33

Flujograma del proceso de entrenamiento del modelo (MobileNet-v2)



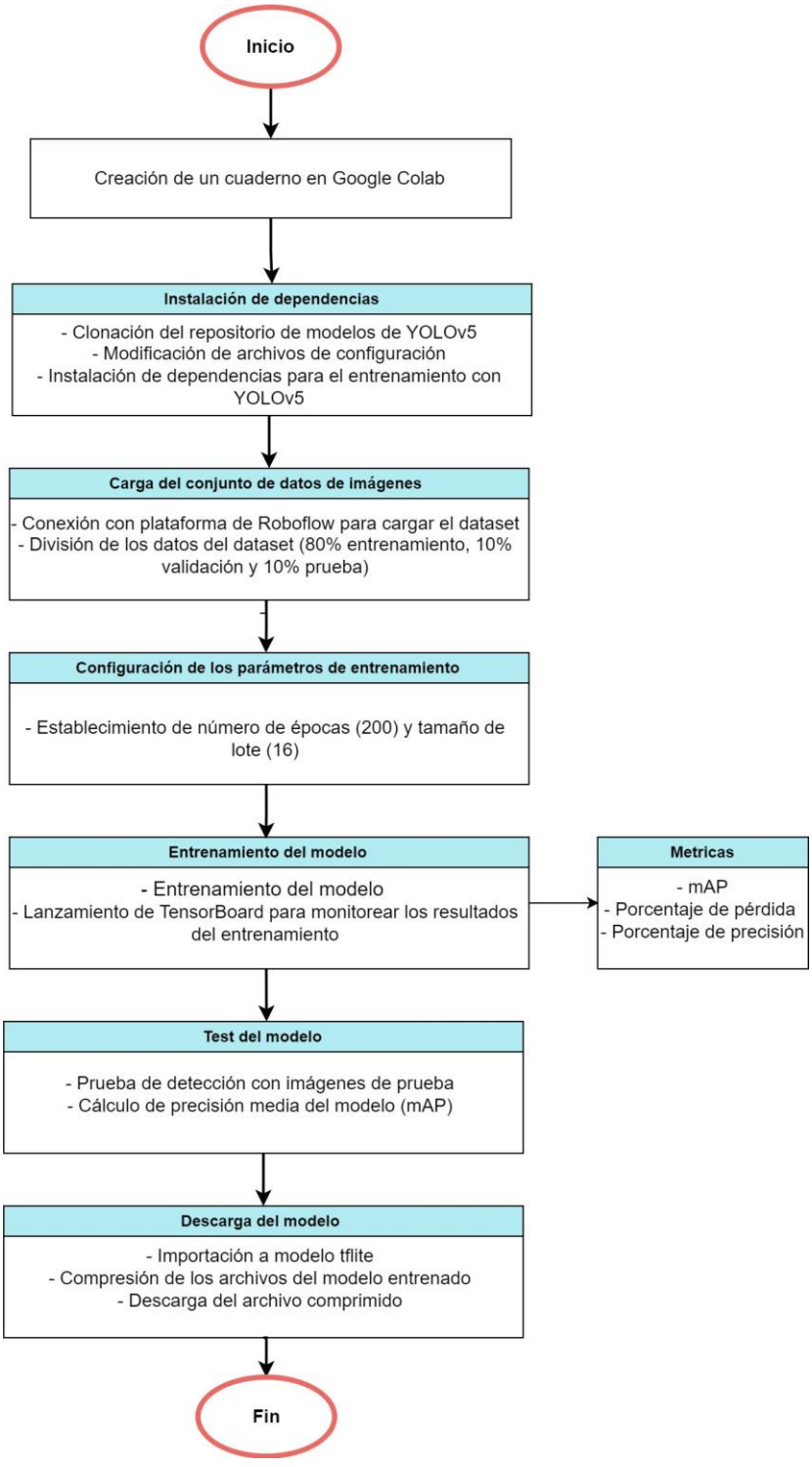
Nota. Autoría propia

3.1.2. Flujograma del proceso de entrenamiento del modelo (YOLOv5)

Ahora para el entrenamiento del modelo utilizando la arquitectura YOLOv5 se realiza un procedimiento similar al anterior apartado, en donde se crea un nuevo cuaderno de trabajo dentro de la plataforma de Google Colaboratory, en donde se realizan la instalación de las dependencias necesarias, para luego entrenar el modelo en base al dataset construido anteriormente, en donde se toma en cuenta las diferentes clasificaciones (inmaduro – plátano, fresco inmaduro – plátano, fresco – plátano, maduro – plátano, maduración excesiva – plátano, podrido – plátano, inmaduro – fresa, maduro – fresa, maduración excesiva – fresa, podrido – fresa). El detalle del proceso de entrenamiento se especifica en la Figura 34.

Figura 34

Flujograma del proceso de entrenamiento del modelo (YOLOv5)



Nota. Autoría propia

3.1.3. Flujograma del script captura de imágenes

Cuando se haya finalizado con el entrenamiento del modelo de detección, se procede con la integración dentro del sistema embebido de monitoreo, en este caso para la primera lógica del sistema se tiene la captura de imágenes mediante la utilización de Raspberry Pi junto con el sensor de Arducam.

Como guía para el flujograma se ha descrito cada una de las librerías utilizadas, esto se puede visualizar en la Tabla 14.

La captura de imágenes (Figura 35) de las frutas inicia con la importación de librerías y dependencias. A continuación, mediante la estructuración de una función se monitorea la hora cada segundo, luego se tiene una condición establecida en las horas 07:00 y 19:00, si la condición llega a cumplirse, este script procede a eliminar las imágenes existentes, después toma y guarda la fotografía en el directorio asignado, y como paso final ejecuta el script de detección del estado de madurez de las frutas.

Tabla 14

Librerías utilizadas para el script de captura de imagen

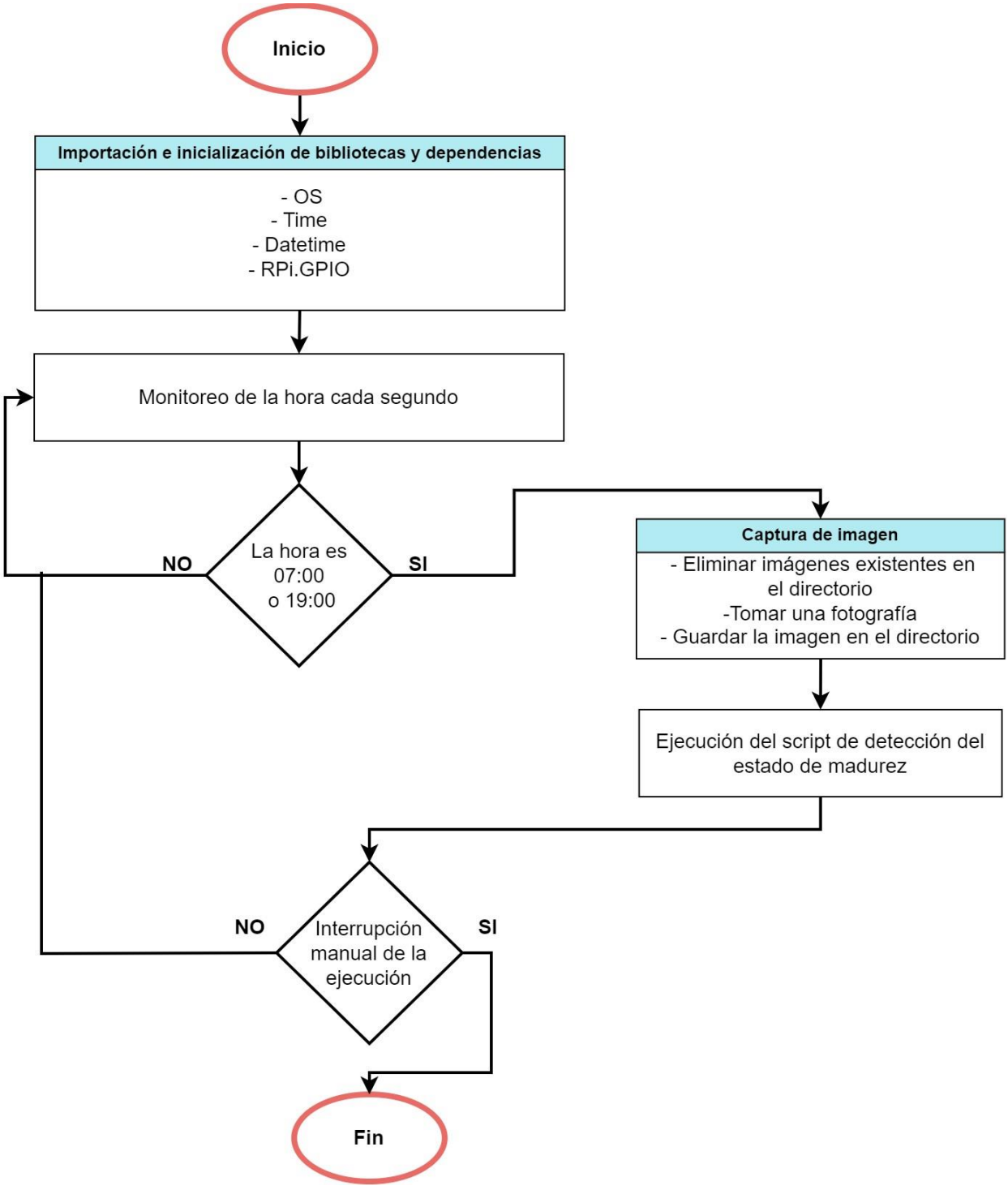
Librerías utilizadas	
Librería	Descripción
OS	Esta librería proporciona una interfaz para trabajar con funcionalidades relacionadas con el sistema operativo.
RPi.GPIO	Esta biblioteca permite controlar los pines GPIO (General Purpose Input/Output) en una Raspberry Pi desde Python

Time	Es una librería de Python que proporciona varias funciones relacionadas con el tiempo
Datetime	Permite el uso de utilidades para trabajar con fechas y horas en Python.

Nota. La tabla indica las diferentes librerías que se ha utilizado para el flujograma de captura de imagenes. Fuente: Autoría propia

Figura 35

Flujograma del script captura de imágenes



Nota. Autoría propia

3.1.4. Flujograma del script del proceso de detección de imágenes y conexión con la base de datos

En esta siguiente parte del sistema embebido los archivos generados a partir del modelo entrenado son colocados dentro del dispositivo Raspberry Pi 4, estos archivos contienen el mapa de etiquetas, además, el modelo entrenado.

En la Tabla 15 se indican las librerías que se han utilizado para el procesamiento de las imágenes por visión artificial y la conexión con la base de datos.

Para la Figura 36 se realiza la representación del proceso que se lleva a cabo para la detección del estado de madurez de las frutas (fresas y plátanos). Inicialmente se importa y se inicializa las librerías y dependencias necesarias, después se realiza la selección y carga de la imagen que se ha tomado en una resolución de 2048x1080. A continuación, en el apartado de "Procesamiento de la imagen" se convierte la imagen a un espacio de color, es decir, a un arreglo en 3 dimensiones, luego se redimensiona la imagen a una resolución de 416x416. Mediante el modelo entrenado se extrae operaciones para la detección y clasificación de los estados de madurez de la fruta en inmaduro – plátano, fresco inmaduro – plátano, fresco – plátano, maduro – plátano, maduración excesiva – plátano, podrido – plátano, inmaduro – fresa, maduro – fresa, maduración excesiva – fresa, podrido – fresa. Finalmente, se realiza el envío de los objectos detectados y clasificados a la base de datos, así como la imagen procesada.

Tabla 15

Librerías utilizadas para el script del proceso de detección

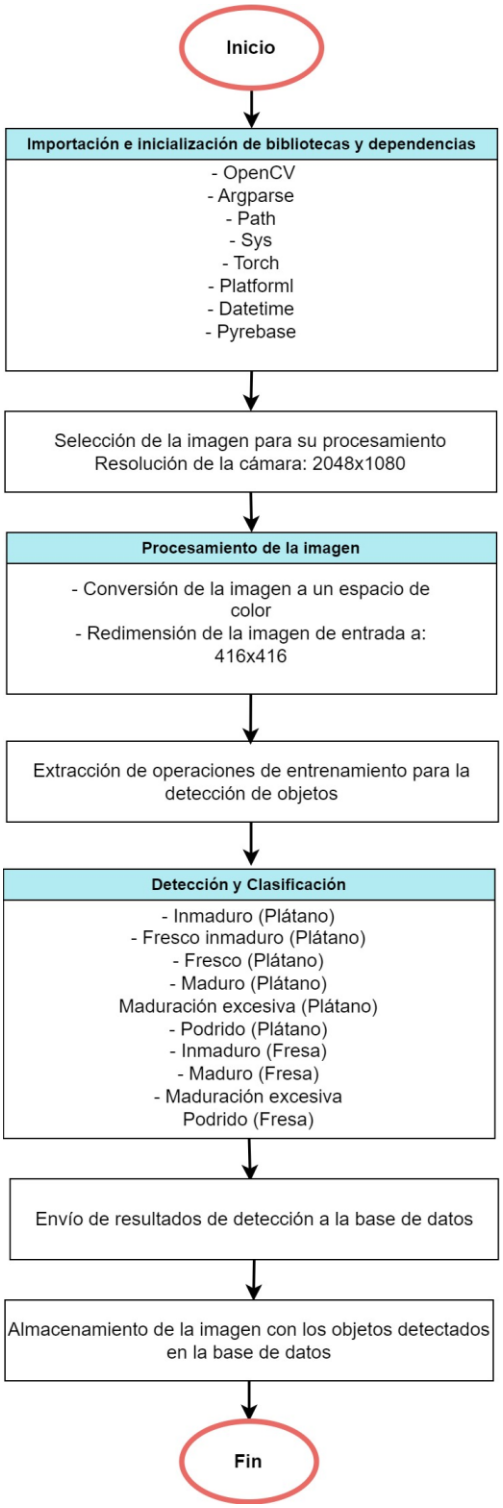
Librerías utilizadas	
Librería	Descripción

Argparse	Permite la definición y análisis de argumentos de línea de comando, es decir, proporciona argumentos al script en la línea de comandos
CV2	Es una biblioteca de visión por computadora que se utiliza para leer, procesar y manipular imágenes y videos.
Path	Proporciona una forma más orientada a objetos y segura de trabajar con rutas de archivos y directorios.
Sys	Esta biblioteca proporciona acceso a variables y funciones utilizadas o mantenidas por el intérprete de Python.
Torch	Se utiliza principalmente para la creación y entrenamiento de redes neuronales y otros modelos de aprendizaje profundo.
Platform	Permite extraer información específica sobre la plataforma en la que se está ejecutando.
Datetime	Permite el uso de utilidades para trabajar con fechas y horas en Python.
Pyrebase	Biblioteca utilizada interactuar con la base de datos en tiempo real y el almacenamiento en la nube de Firebase.

Nota. En esta tabla se describe las librerías que se ha utilizado dentro del script de detección de imágenes y conexión con la base de datos. Fuente: Autoría propia

Figura 36

Flujograma del script del proceso de detección de imágenes y conexión con la base de datos



Nota. Autoría propia

3.1.5. Flujograma del funcionamiento de la aplicación móvil

Ahora, otra parte de este sistema embebido de monitoreo es la visualización de los resultados de forma amigable con el usuario, por lo cual, se ha desarrollado una aplicación móvil que permite la visualización de los objetos detectados en la imagen procesada mediante el modelo entrenado. Además, indicando cada una de las etiquetas del estado de maduración de las frutas según se presenten en la imagen, adicionalmente proporcionando una recomendación y alerta en los estados más avanzados de maduración o descomposición de la fruta.

En la Tabla 16 se indican las bibliotecas que se han usado para el proceso de desarrollo de la aplicación móvil para Android, además de indicando una breve descripción de estas librerías.

El funcionamiento de la aplicación móvil se representa en la Figura 37 en donde el proceso para la obtención de los resultados de la imagen procesada por medio del modelo entrenado empieza con, la importación de las librerías correspondientes, a continuación se verifica los permisos para el acceso a notificaciones del teléfono, si se cumple con esta condición se realiza un descarga directa de la imagen desde la base de datos de Firebase, esta misma imagen se redimensiona a una escala de 1000x1000 y posteriormente se muestra en un recuadro de visualización de imágenes (ImageView), después se descarga las etiquetas que se han detectado dentro de la imagen, para ser visualizadas en un recuadro de visualización de texto (TextView). En caso de que, una o más etiquetas tengan el valor de maduración excesiva o podrido, ya sea de plátanos o fresas, se activa una notificación de alerta, además emitiendo una recomendación según sea el caso. En caso de no cumplir con esta condición, se asume que la fruta detectada se encuentra en un estado óptimo.

Tabla 16

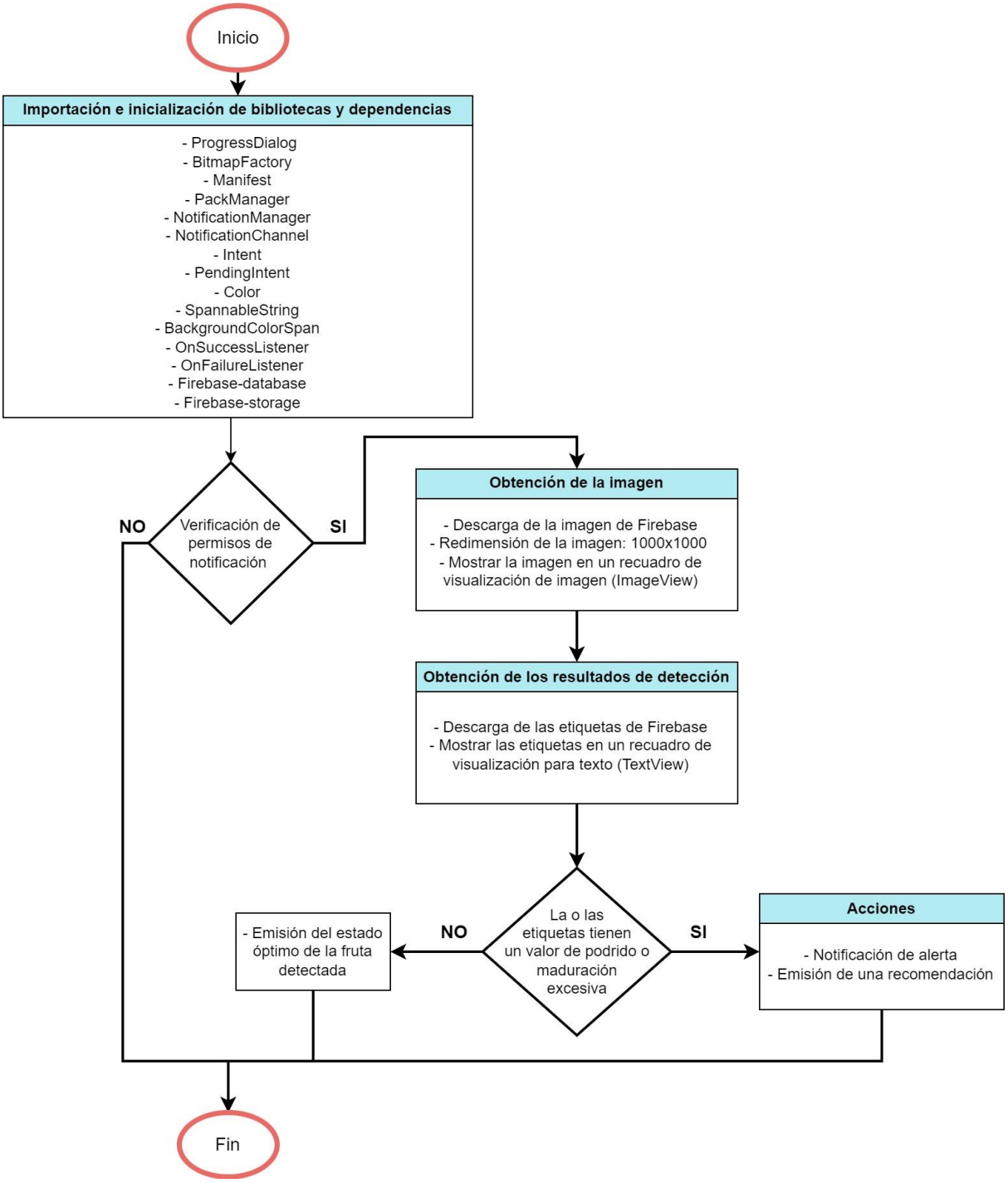
Librerías utilizadas para el script de captura de imagen

Librerías utilizadas	
Librería	Descripción
ProgressDialog	Proporciona una interfaz gráfica de progreso para mostrar el avance de una operación en segundo plano.
BitmapFactory	Permite la utilización de clases y métodos para trabajar con imágenes y mapas de bits.
Manifest y PackManager	Proporciona utilidades para la solicitud y gestión de permisos en la aplicación
NotificationManager y NotificationChannel	Utilizadas para la administración de notificaciones y canales de notificación en Android
Intent y PendingIntent	Se utilizan para iniciar actividades y crear intenciones pendientes.
Color	Proporciona métodos para trabajar con colores en Android.
SpannableString y BackgroundColorSpan	Librerías utilizadas para aplicación de estilos en el texto
OnSuccessListener y OnFailureListener	Proporcionan interfaces manejar el éxito o el fracaso de las operaciones asíncronas en Firebase.
Firebase-database	Permite el uso de clases y métodos para interactuar con la base de datos de Firebase.
Firebase-storage	Proporciona clases y métodos para trabajar con el almacenamiento de Firebase.

Nota. Tabla de librerías utilizadas en la aplicación móvil. Fuente: Autoría propia

Figura 37

Flujograma del funcionamiento de la aplicación móvil



Nota. Autoría propia

3.2. Entrenamiento de la red neuronal

Para el entrenamiento del modelo que se utilizará dentro del sistema embebido de monitoreo por visión artificial, se han elegido dos tipos de arquitecturas neuronales MobileNet-v2 y YOLOv5, esto con el propósito de elegir la arquitectura con mejor desempeño para el sistema.

3.2.1. MobileNet-v2

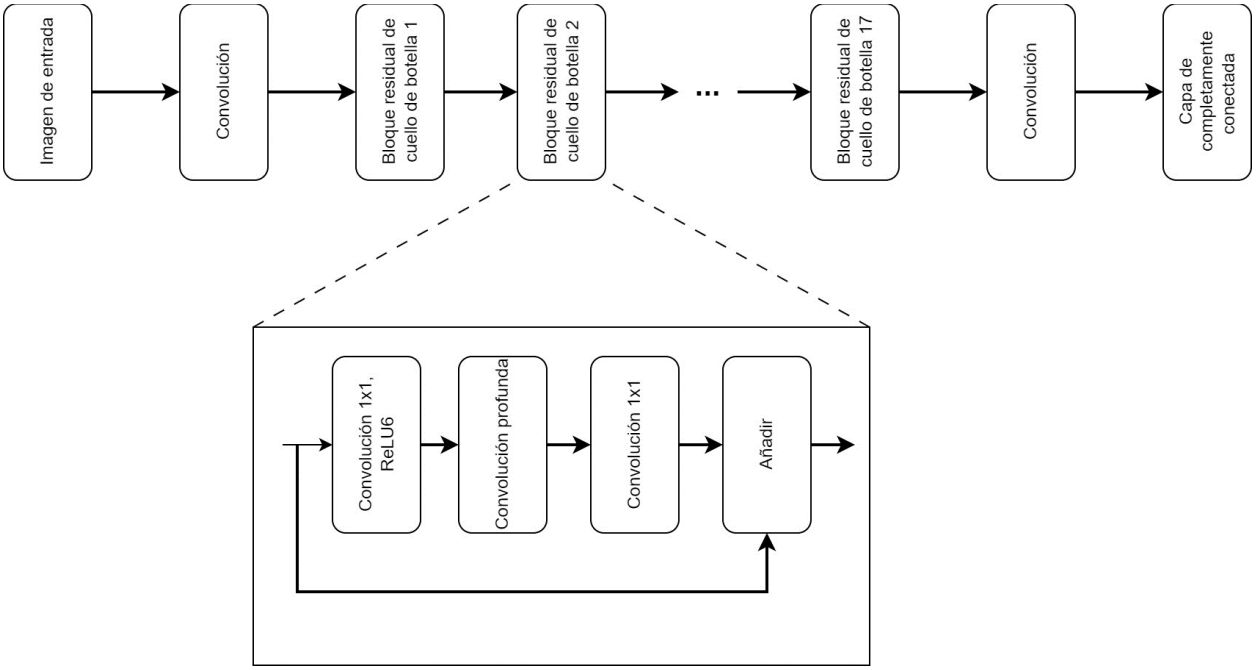
Para el entrenamiento de este modelo se ha utilizado la arquitectura de MobileNet-v2 la cual está compuesta por 19 bloques de convolución: 2 bloques de convolución con una capa única, y 17 bloques que contienen 3 tipos de capas diferentes (Capa de convolución 1x1 con ReLU6, Capa de convolución profunda y capa de convolución sin linealidad).

- Capa de convolución 1x1 con ReLU6: La convolución 1x1 es un tipo especial de convolución que utiliza un filtro de tamaño 1x1. A diferencia de las convoluciones tradicionales que utilizan filtros de tamaño mayor, la convolución 1x1 tiene una ventana de tamaño 1x1 y se utiliza principalmente para realizar operaciones de reducción de dimensionalidad o mezclar características en una red neuronal convolucional. Después de este primer proceso se agrega una función ReLU6 el cual que satura los valores negativos en cero y los valores mayores a 6 en 6.
- Capa de convolución profunda: Esta capa también denominada como “Depthwise Convolution”, es otra técnica que se utiliza para reducir la cantidad de parámetros y operaciones en una red neuronal convolucional. Esta capa aplica más de un filtro a todas las entradas, la convolución profunda aplica un filtro separado a cada canal de entrada.
- Capa de convolución sin linealidad: Esta capa es similar a la primera, sin embargo, se elimina la función ReLU6, esto con la finalidad de conservar la linealidad de las características.

Si se suma las todas las capas internas, se tienen un total de 53 capas ocultas para esta arquitectura según indica (OpenGenus IQ, 2023), en donde se encontrar la tabla de convoluciones con las distintas características en el enlace https://iq.opengenus.org/mobilenetv2-architecture/#google_vignette, además, el diagrama de la arquitectura se encuentra representado en la Figura 38.

Figura 38

Arquitectura MobileNetV2



Nota. En figura se la estructura de la arquitectura neuronal de MobileNetv2. Adaptado de *Real-Time and Accurate Drone Detection in a Video with a Static Background* (Seidaliyeva et al., 2020).

Para determinar el número de neuronas que se utiliza en las capas ocultas de la arquitectura MobileNet-v2, inicialmente se considera un entrada de 320x320x3 para imágenes, y a su salida es capaz de diferenciar hasta 1000 categorías diferentes, sin embargo para este caso solo se utiliza

una salida de 10 categorías; por lo cual se indica un valor de 307200 neuronas en la capa de entrada y para la capa de salida 10 neuronas; ahora para el cálculo de las neuronas de las 53 capas ocultas que contiene esta arquitectura neuronal, hace referencia al uso de neuronas compartidas entre capas, de esta manera no se puede determinar de manera exacta el número de neuronas que hay en cada una de las capas ocultas, sin embargo, se especifica que el número de neuronas en las capas ocultas debe ser 2/3 del tamaño de la capa de entrada, más el tamaño de la capa de salida (Krishnan Sandhya, 2021).

En la ecuación 1, se indica la fórmula para el cálculo de las neuronas en capas ocultas.

$$h = \frac{2}{3} * in + out \quad (1)$$

donde;

h = número de neuronas de las capas ocultas

in = número de neuronas en la capa de entrada

out = número de neuronas en la capa de salida

Aplicando la fórmula anterior con los datos conocidos, se tiene el siguiente cálculo:

$$h = \frac{2}{3} * (307200) + 10$$

$$h = 204800 + 10$$

$$\boldsymbol{h = 204810 \text{ neuronas}}$$

De acuerdo al resultado obtenido se indica que se tiene un total de 204810 neuronas para las capas ocultas.

Este proceso fue realizado a partir de la modificación de un ejemplo de TensorFlow (https://colab.research.google.com/github/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/Train_TFLite2_Object_Detction_Model.ipynb) para el entrenamiento de modelos personalizados.

En la Figura 39 se indica como primer paso, la clonación de un repositorio de GitHub, en donde mediante la opción “depth” se realiza la descarga la versión más reciente del repositorio.

Figura 39

Clonación del repositorio de modelos de TensorFlow

```
# Clonar el repositorio de modelos tensorflow desde GitHub
!git clone --depth 1 https://github.com/tensorflow/models

Cloning into 'models'...
remote: Enumerating objects: 3916, done.
remote: Counting objects: 100% (3916/3916), done.
remote: Compressing objects: 100% (3027/3027), done.
remote: Total 3916 (delta 1130), reused 1811 (delta 836), pack-reused 0
Receiving objects: 100% (3916/3916), 49.65 MiB | 17.72 MiB/s, done.
```

Nota. Autoría propia

Luego de clonar los archivos dentro del entorno de trabajo, se realizan modificaciones en el archivo de configuración de TensorFlow, esto mediante la utilización “protoc”, y a continuación se modifica la versión de TensorFlow de 2.5.1 a 2.8.0 en este mismo archivo, esto con el fin de permitir la compatibilidad con librerías y dependencias más actuales

Cuando se haya finalizado con la modificación de los archivos de configuración se procede con la instalación de las dependencias requeridas para el entrenamiento del modelo, además como se indica en la Figura 40 se puede observar la instalación de TensorFlow en su versión 2.8.0.

Figura 40

Instalación de TensorFlow

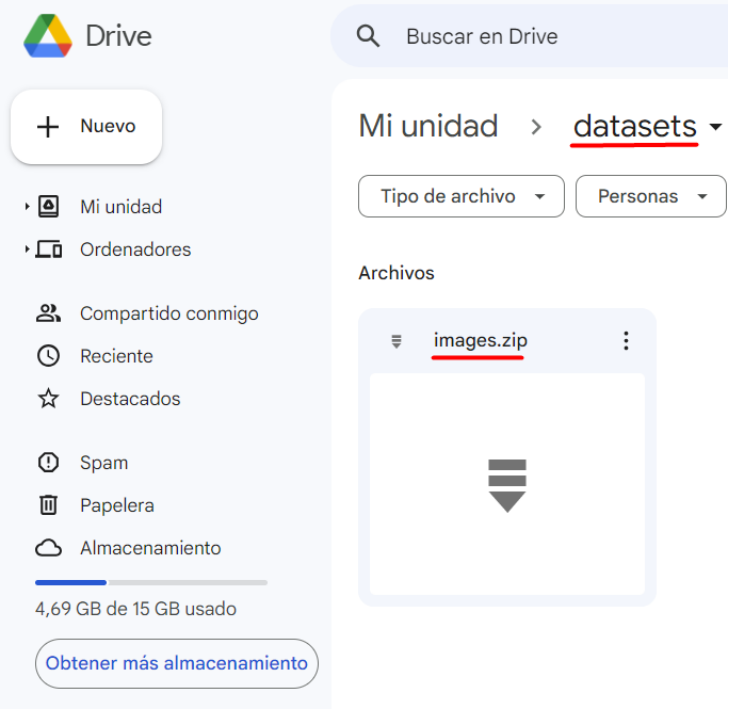
```
# Instalar la API de detección de objetos
!pip install /content/models/research/

# Instalar tensorflow
!pip install tensorflow==2.8.0
```

Nota. Autoría propia

El conjunto de datos de imágenes, los cuales han sido etiquetados, son cargados dentro de la unidad de Google Drive, en el cual, se ha creado un directorio denominado “datasets”. Este conjunto de imágenes ha sido comprimido dentro de un mismo archivo como se observa en la Figura 41.

Figura 41

Subida de los archivos etiquetados a Google Drive



Nota. Autoría propia

Para la conexión de Google Drive con el cuaderno de Colab se procede con las siguientes líneas de código presentadas en la Figura 42, se importa la librería “drive” y se copia el archivo subido anteriormente.

Figura 42

Importación de los archivos etiquetados dentro del entorno

```
from google.colab import drive
drive.mount('/content/gdrive')

!cp /content/gdrive/MyDrive/datasets/images.zip /content
```

```
Mounted at /content/gdrive
```

Nota. Autoría propia

Cuando se hayan cargado las imágenes correspondientes dentro del entorno de Colab se procede con la extracción de todas estas imágenes, además con la creación de los respectivos directorios y subdirectorios para las imágenes (entrenamiento, validación y pruebas), todo esto en la Figura 43.

Figura 43

Creación directorios y subdirectorios

```
!mkdir /content/images
!unzip -q images.zip -d /content/images/all
!mkdir /content/images/train; mkdir /content/images/validation; mkdir /content/images/test
```

Nota. Autoría propia

Para división del porcentaje de imágenes en los diferentes directorios, se utiliza un script creado, el cual únicamente debe ser descargado y ejecutado. En la Figura 44 se indica este proceso; mediante este procedimiento se ha logrado clasificar las imágenes en un 80% para datos de entrenamiento, 10% para validación y 10% para pruebas.

Figura 44

Clasificación de los datos de validación, entrenamiento y prueba

```
!wget https://raw.githubusercontent.com/EdjeElectronics/TensorFlow-Lite-Object
!python train_val_test_split.py
```

Nota. Autoría propia

Después de la clasificación de las imágenes en los diferentes directorios, se define la lista de clases que se tienen dentro del conjunto de dato, visualizar Figura 45, cabe recalcar que estas clases deben ser definidas tal cual se han etiquetado.

Figura 45

Definición de las clases

```
### Creación de la lista de clases
%%bash
cat <<EOF >> /content/labelmap.txt
unripe
freshunripe
freshripe
ripe
overripe
rotten
inmaduro
maduro
demasiadoM
podrido
EOF
```

Nota. Autoría propia

Para esta sección de forma inicial se tiene la selección de la arquitectura, indicada en la Figura 46, en este caso se ha seleccionado la arquitectura denominada “ssd-mobilenet-v2-fpnlite-320”, dentro de este mismo apartado al seleccionar esta opción se asignan valores a las tres variables “model_name” (nombre del modelo), base_pipeline_file (contiene el nombre del archivo de configuración base del modelo, como los hiperparámetros necesarios para el entrenamiento) y pretrained_checkpoint (el nombre del archivo de punto de control pre-entrenado del modelo), este archivo contiene los pesos y parámetros del modelo previamente entrenado en un conjunto de datos grande.

Figura 46

Selección de la arquitectura que se va a utilizar

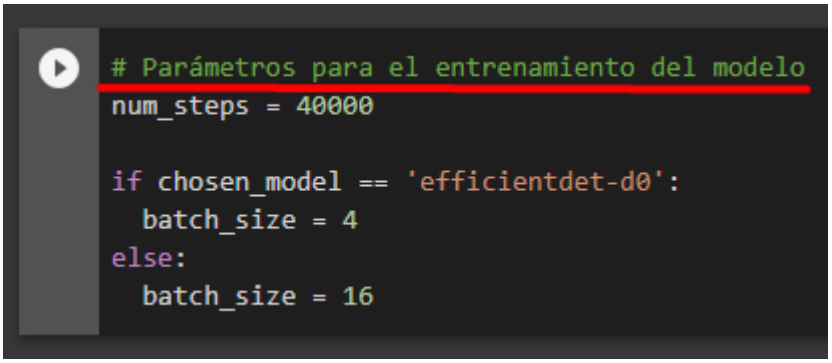
```
[ ]  # Elección de la arquitectura
     chosen_model = 'ssd-mobilenet-v2-fpnlite-320'

     MODELS_CONFIG = {
         'ssd-mobilenet-v2': {
             'model_name': 'ssd_mobilenet_v2_320x320_coco17_tpu-8',
             'base_pipeline_file': 'ssd_mobilenet_v2_320x320_coco17_tpu-8.config',
             'pretrained_checkpoint': 'ssd_mobilenet_v2_320x320_coco17_tpu-8.tar.gz',
         },
         'efficientdet-d0': {
             'model_name': 'efficientdet_d0_coco17_tpu-32',
             'base_pipeline_file': 'ssd_efficientdet_d0_512x512_coco17_tpu-8.config',
             'pretrained_checkpoint': 'efficientdet_d0_coco17_tpu-32.tar.gz',
         },
         'ssd-mobilenet-v2-fpnlite-320': {
             'model_name': 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8',
             'base_pipeline_file': 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.config',
             'pretrained_checkpoint': 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz',
         },
     }

     model_name = MODELS_CONFIG[chosen_model]['model_name']
     pretrained_checkpoint = MODELS_CONFIG[chosen_model]['pretrained_checkpoint']
     base_pipeline_file = MODELS_CONFIG[chosen_model]['base_pipeline_file']
```

Nota. Autoría propia

Para la descarga de la arquitectura que se ha seleccionado anteriormente, se debe dirigir al enlace http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd-mobilenet-2-fplite-320, en donde se encontraran las configuraciones predefinidas para la arquitectura MobileNet-v2. A continuación, en la Figura 47 se establece el número de pasos para el entrenamiento del modelo, el cual para este caso se guarda dentro la variable num_steps asignado con un valor de 40000, y además se indica el uso de un tamaño de lote de 16, esto asignado en la variable batch_size.

Figura 47

Configuración de número de pasos y tamaño de lote



Nota. Autoría propia

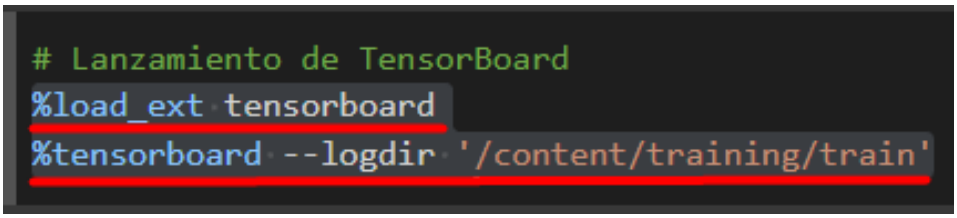
Cuando se finaliza la definición del número de pasos y el tamaño del lote, se procede a guardar el archivo de configuración para el entrenamiento, y luego se establecen las rutas para los archivos de configuración que se ha modificado y otros archivos adicionales.

En la Figura 48 se indica el lanzamiento de TensorBoard, la cual es una herramienta que nos permite medir algunas métricas del entrenamiento del modelo como Tasa de pérdida y Tasa de aprendizaje por paso. Mediante el comando "%load_ext tensorboard", se carga la extensión de TensorBoard en el entorno y por medio del comando "%tensorboard –logdir

‘/content/training/train’” se inicia TensorBoard y especifica el directorio donde se encuentran los registros de entrenamiento.

Figura 48

Lanzamiento de tensorboard



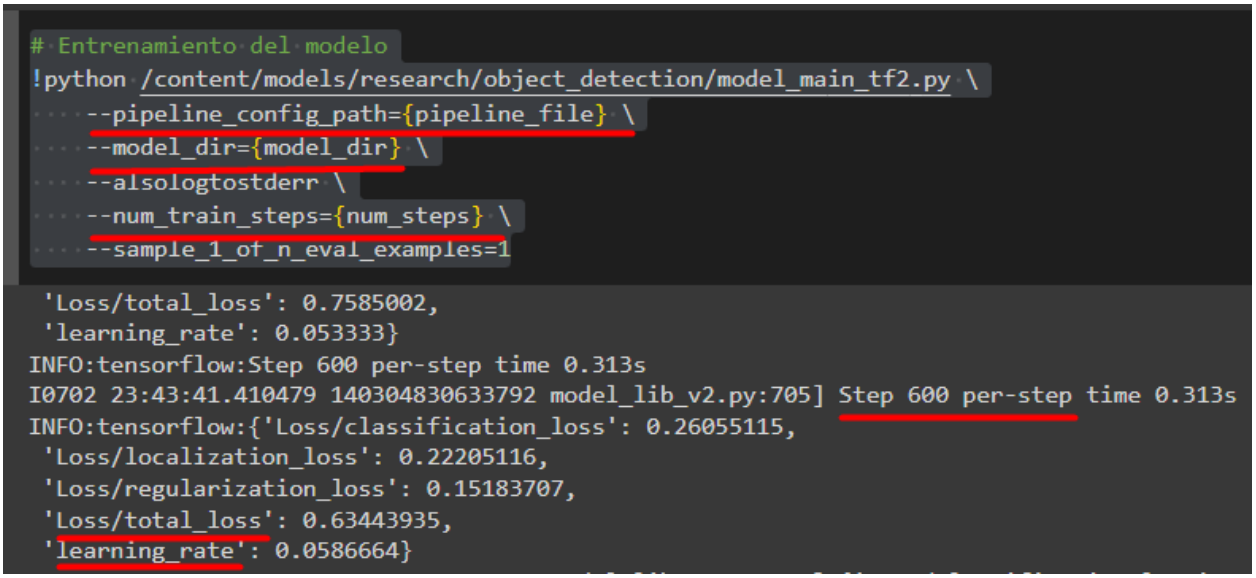
Nota. Autoría propia

En este código visualizado en la Figura 49, se ejecuta un script de TensorFlow para entrenar un modelo de detección de objetos utilizando TensorFlow 2 y el API de detección de objetos de TensorFlow. Además, se indican argumentos como:

- “pipeline_config_path={pipeline_file}”: el cual contiene la configuración para el entrenamiento del modelo.
- “model_dir={model_dir}”: en donde se especifica el directorio donde se guardarán los archivos relacionados con el entrenamiento del modelo.
- “alsologtostderr”: es una opción que indica para mostrar la información de registro tanto en la salida estándar como en los archivos de registro.
- “num_train_steps{num_steps}: especifica el número total de pasos de entrenamiento que se realizarán durante el proceso de entrenamiento.

Figura 49

Entrenamiento del modelo



Nota. Autoría propia

Para la Figura 50 el comando crea un nuevo directorio llamado "custom_model_lite" en la ruta "/content/" que se utilizará para almacenar el modelo TFLite convertido. Luego se define la variable “last_model_path” para almacenar el contenido del modelo entrenado. A continuación, se ejecuta un script de Python para la conversión del modelo a un formato TFLite, en donde, se asignan argumentos como:

- “trained_checkpoint_dir {last_model_path}”: Indica el directorio que contiene el último punto de control (checkpoint) del modelo entrenado.
- “output_directory {output_directory}”: Especifica el directorio donde se guardará el modelo TFLite exportado.
- “pipeline_config_path {pipeline_file}”: Especifica la ruta del archivo de configuración del pipeline utilizado durante el entrenamiento del modelo.

Figura 50

Creación de rutas para el nuevo modelo

```
# Creación un directorio para almacenar el modelo TFLite entrenado
!mkdir /content/custom_model_lite
output_directory = '/content/custom_model_lite'

# Ruta del archivo de entrenamiento
last_model_path = '/content/training'

!python /content/models/research/object_detection/export_tflite_graph_tf2.py \
    --trained_checkpoint_dir {last_model_path} \
    --output_directory {output_directory} \
    --pipeline_config_path {pipeline_file}
```

Nota. Autoría propia

Finalmente, para la conversión del modelo entrenado a un formato TFLite se tiene el siguiente código, indicado en la Figura 51, en donde inicialmente se realiza la importación de la biblioteca de TensorFlow para utilizar sus funciones y clases. Posteriormente, por medio del módulo TFLiteConverter se realiza la conversión del modelo TensorFlow a un modelo TFLite, y como último paso de este proceso, se genera un archivo del modelo TFLite.

Figura 51

Conversión del modelo de TensorFlow a TensorFlow Lite

```
# Conversión del archivo gráfico exportado en un archivo de modelo TFLite
import tensorflow as tf

converter = tf.lite.TFLiteConverter.from_saved_model('/content/custom_model_lite/saved_model')
tflite_model = converter.convert()

with open('/content/custom_model_lite/detect.tflite', 'wb') as f:
  f.write(tflite_model)
```

Nota. Autoría propia

Para la evaluación de este modelo entrenado se ha plateado el siguiente script, el cual es capaz de detectar los objetos mediante el uso de este mismo modelo.

En la Figura 52 de forma inicial, se define ciertos valores para las variables, en donde principalmente se asigna rutas de directorios acerca del modelo entrenado. A continuación, mediante el uso de la función de “glob” se obtiene el nombre de las imágenes, finalmente se ejecuta el script que se ha explicado anteriormente con todos los parámetros que se ha definido inicialmente en este fragmento. Básicamente se realiza una inferencia de las imágenes, para ser exactos se realiza la inferencia de 470 imágenes.

Figura 52

Inferencia de las imágenes

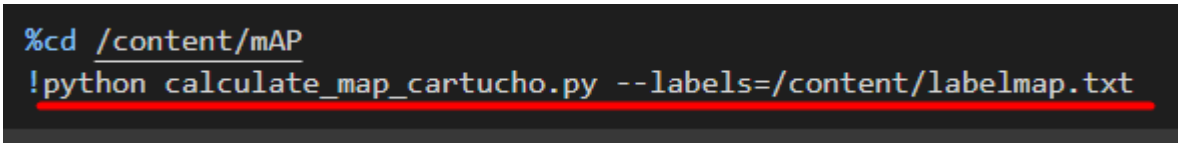
```
# Configuración de variables
PATH_TO_IMAGES='/content/images/test'   # Ruta a la carpeta de imágenes de prueba
PATH_TO_MODEL='/content/custom_model_lite/detect.tflite'   # Ruta de acceso al archivo de modelo .tflite
PATH_TO_LABELS='/content/labelmap.txt'   # Ruta al archivo labelmap.txt
PATH_TO_RESULTS='/content/mAP/input/detection-results' # Carpeta en la que guardar los resultados de la detección
min_conf_threshold=0.1   # Umbral de confianza

# Utilizar todas las imágenes de la carpeta de prueba
image_list = glob.glob(PATH_TO_IMAGES + '/*.jpg') + glob.glob(PATH_TO_IMAGES + '/*.JPG') +
             glob.glob(PATH_TO_IMAGES + '/*.png') + glob.glob(PATH_TO_IMAGES + '/*.bmp')
images_to_test = min(500, len(image_list)) # Si hay más de 500 imágenes en la carpeta, utilice 500

# Indicar a la función que sólo guarde los resultados
txt_only = True

# Ejecutar función de inferencia
print('Starting inference on %d images...' % images_to_test)
tflite_detect_images(PATH_TO_MODEL, PATH_TO_IMAGES, PATH_TO_LABELS, min_conf_threshold, images_to_test, PATH_TO_RESULTS, txt_only)
print('Interferencia finalizada!')
```

Nota. Autoría propia

Para calcular la precisión del modelo que se ha entrenado, se ha utilizado un método denominado “precisión promedio” o mAP, el cual define que cuanto mayor sea el porcentaje de puntuación (score) mejor será el modelo, en donde inicialmente es descarga el repositorio para luego ejecutar el comando mostrado en la Figura 53.

Figura 53

Ejecución del script para determinar el grado de precisión



Nota. Autoría propia

Finalmente, se realiza la compresión y la descarga de los archivos del modelo entrenado.

3.2.2. YOLOv5

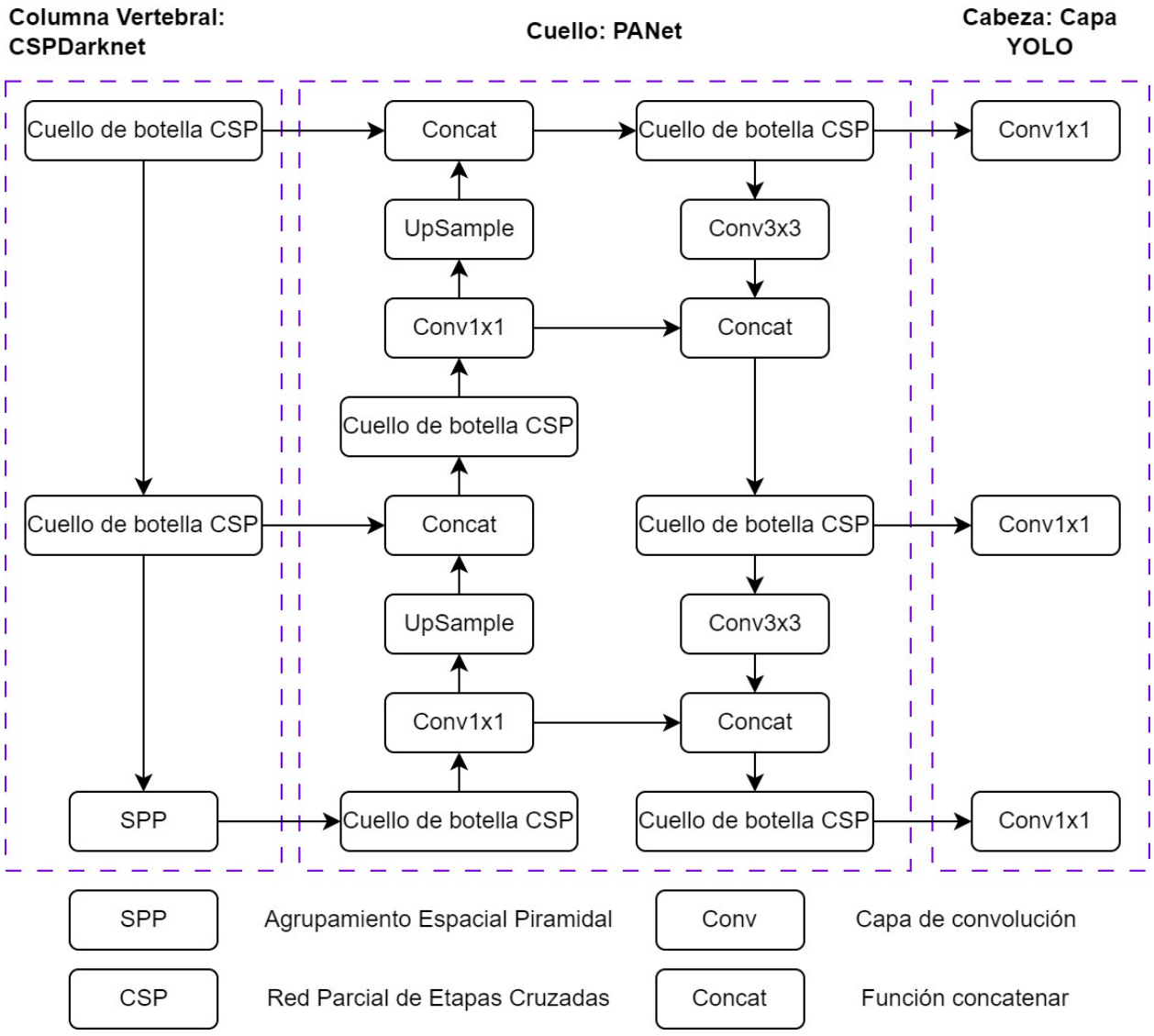
YOLO (You Only Look Once), es un enfoque para la detección de objetos en imágenes, videos y en tiempo real, a diferencia de otras formas de detección, que se realizan por la segmentación de la imagen para la extracción de características, YOLO realiza la detección múltiple a través de una sola imagen. Específicamente en este entrenamiento se ha utilizado YOLOv5 la cual está compuesta por tres partes principales:

- Columna vertebral: Para la parte principal de esta arquitectura se utiliza una variación de Darknet53, que en este caso es CSPDarknet53 la cual contiene 29 capas convolucionales de 3x3 (Bochkovskiy et al., 2020).
- Cuello: Para este apartado se utiliza una arquitectura denominada PANet, la cual está diseñada para mejorar la capacidad de las redes de segmentación semántica para fusionar y procesar información de características de diferentes escalas en una imagen (Sik-Ho, 2020), en la cual se posee 10 capas para este proceso.
- Cabeza: Y para la última sección llamada Capa YOLO, se identifican 3 capas de convolución.

La arquitectura de red para YOLOv5 es presentada en la Figura 54.

Figura 54

Arquitectura Neuronal de YOLOv5



Nota. En figura se indica la arquitectura neuronal de YOLOv5. Adaptado de *A forest fire detection system based on ensemble Learning* (Xu et al., 2021).

Para un mejor entendimiento se explican mejor algunos bloques

- Cuello de botella CSP: Estos bloques hacen referencia las capas convolucionales de la arquitectura CSPDarknet53.
- SPP: Denominado agrupamiento espacial piramidal se utiliza para procesar características de diferentes escalas espaciales en una imagen.
- UpSample: Es un bloque que se usa en redes neuronales convolucionales para aumentar la resolución espacial de las características.
- Concat: Para concatenar o unir características de múltiples canales o fuentes
- Conv1x1: Se refiere a una operación de convolución con un kernel (filtro) de tamaño 1x1
- Conv3x3: Hace referencia una operación de convolución con un kernel (filtro) de tamaño 1x1

Otro factor a considerar dentro de esta arquitectura es el número de neuronas que conforma la red neuronal, se sabe que la arquitectura de YOLOv5 se tiene un entrada de 640x640x3 para imágenes, y a su salida es capaz de diferenciar hasta 1000 posibles categorías diferentes, pero como se había mencionado anteriormente solo se usará la clasificación de 10 posibles clases, con estos datos se puede indicar que, para la capa de entrada se obtiene un valor de 1228800 neuronas y para la capa de salida 10 neuronas; ahora para el cálculo de las neuronas de las 42 capas constituidas entre CSPDarknet23, PANet y Capa YOLO, se indica el uso de neuronas compartidas entre capas, por lo cual no se puede calcular de forma exacta las neuronas en cada una de estas capas, a pesar de ello, se puede realizar un cálculo aproximado de la neuronas en las capas ocultas, haciendo referencia a lo que propone (Krishnan Sandhya, 2021), explicado en el siguiente apartado Se reescribe nuevamente la Ecuación 1.

$$h = \frac{2}{3} * in + out \tag{1}$$

donde;

h = número de neuronas de las capas ocultas

in = número de neuronas en la capa de entrada

out = número de neuronas en la capa de salida

A continuación, se realiza el reemplazo de los valores en la ecuación:

$$h = \frac{2}{3} * (1228800) + 10$$

$$h = 819200 + 10$$

$$h = \ 820210$$

El resultado que se ha obtenido son un total de 820210 neuronas en las capas ocultas.

A continuación, se inicia con el entrenamiento del modelo utilizando la arquitectura de YOLOv5, cabe resaltar que este proceso es realizado en base al ejemplo que proporciona la plataforma de Roboflow https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/train-yolov5-object-detection-on-custom-data.ipynb.

Como primer paso se realiza la clonación de los repositorios oficiales, observar la Figura 55.

Figura 55

Clonación del repositorio de YOLOv5

```
[ ]  # Clonar el repositorio de YOLOv5
     !git clone https://github.com/ultralytics/yolov5  # clone repo
     %cd yolov5
     !git reset --hard 064365d8683fd002e9ad789c1e91fa3d021b44f0

     Cloning into 'yolov5'...
     remote: Enumerating objects: 15937, done.
     remote: Counting objects: 100% (57/57), done.
     remote: Compressing objects: 100% (35/35), done.
     remote: Total 15937 (delta 27), reused 39 (delta 22), pack-reused 15880
     Receiving objects: 100% (15937/15937), 14.66 MiB | 29.61 MiB/s, done.
     Resolving deltas: 100% (10926/10926), done.
     /content/yolov5
     HEAD is now at 064365d Update parse_opt() in export.py to work as in train.py (#10789)
```

Nota. Autoría propia

A continuación, también se instalan las dependencias necesarias para el entrenamiento y el funcionamiento de YOLOv5, en este caso dentro del repositorio que se ha clonado anteriormente se encuentra un archivo con todos los complementos necesarios para la instalación, observar Figura 56.

Figura 56

Instalación de requerimientos para YOLOv5

```
# Instalación de instancias
!pip install -qr requirements.txt  # Instalación de requerimientos
import torch

from IPython.display import Image, clear_output  # Librerías para visualizar imagenes
from utils.downloads import attempt_download  # Importación para el uso de modelos

# clear_output()
```

Nota. Autoría propia

Para la importación del dataset que se ha creado en la sección de “Preprocesamiento de imagenes”, se instala la dependencia de “roboflow” y se coloca las líneas y el código proporcionado por la plataforma de Roboflow cuando se realizó la exportación, todo esto se visualiza en la Figura 57.

Figura 57

Importación del dataset dentro del entorno de Colab

```
[ ]  %cd /content/yolov5
     # Descarga del dataset construido dentro del entorno de trabajo
     !pip install roboflow

     from roboflow import Roboflow
     rf = Roboflow(api_key="[redacted]")
     project = rf.workspace("tesis-z9cju").project("sistema-de-monitoreo")
     dataset = project.version(1).download("yolov5")
```

Nota. Autoría propia

Antes del entrenamiento del modelo se ha realizado la modificación en el modelo que se utilizará, en este caso se indica el número de clases, la profundidad del modelo y un valor para el canal de capa múltiple. Además, se tiene el apartado de “anclajes”, los cuales son como guías para el trazado de las cajas delimitadoras, finalmente se tiene algunas configuraciones de los bloques de esta red neuronal como se muestra en la Figura 58.

Figura 58

Modificación de los parámetros del archivo de configuración de YOLOv5

```
[ ]  %%writetemplate /content/yolov5/models/custom_yolov5s.yaml

     # Parametros
     nc: {num_classes}  # Número de clases
     depth_multiple: 0.33  # Modelo de profundidad múltiple
     width_multiple: 0.50  # Canal de capa múltiple

     # Anclajes
     anchors:
       - [10,13, 16,30, 33,23]  # P3/8
       - [30,61, 62,45, 59,119]  # P4/16
       - [116,90, 156,198, 373,326]  # P5/32

     # Estructura de YOLOv5
     backbone:
       # [from, number, module, args]
       [[-1, 1, Focus, [64, 3]],  # 0-P1/2
        [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
        [-1, 3, BottleneckCSP, [128]],
        [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
        [-1, 9, BottleneckCSP, [256]],
        [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
        [-1, 9, BottleneckCSP, [512]],
        [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
        [-1, 1, SPP, [1024, [5, 9, 13]]],
        [-1, 3, BottleneckCSP, [1024, False]],  # 9
       ]

     # Cabecera YOLOv5
```

Nota. Autoría propia

En la Figura 59, se indica el entrenamiento del modelo, en donde se indica las dimensiones de la imagen (460x460), un tamaño de lote de 16 y 200 épocas; los otros parámetros indican únicamente el uso del archivo de comunicación anterior y en que variables se van a almacenar los resultados.

Figura 59

Entrenamiento del modelo

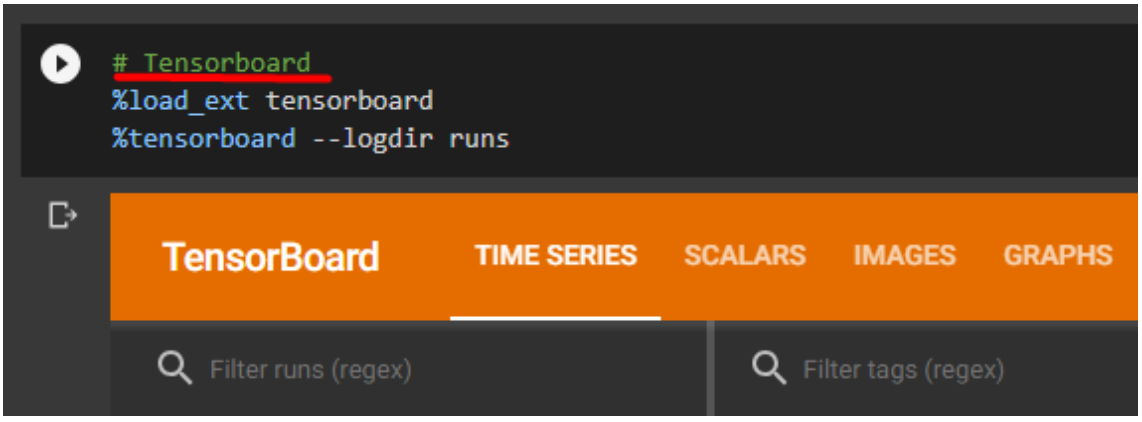
```
# Entramiento del modelo (200 épocas)
%%time
%cd /content/yolov5/
!python train.py --img 460 --batch 16 --epochs 200 --data {dataset.location}/data.yaml
  --cfg ./models/custom_yolov5s.yaml --weights '' --name yolov5s_results  --cache

/content/yolov5
2023-09-01 01:51:50.517442: I tensorflow/core/platform/cpu_feature_guard.cc:182] This Te
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow
```

Nota. Autoría propia

Cuando finalice el proceso de entrenamiento, se implementa dentro del entorno la dependencia de TensorBoard, observar Figura 60, en donde se indica el directorio en el que se ha almacenado los resultados del entrenamiento.

Figura 60

Lanzamiento de TensorBoard



Nota. Autoría propia

Como último paso de este entrenamiento, se realiza la conversión del modelo a una versión tflite, esto se realiza por medio del script que viene dentro del repositorio clonado de YOLOv5, se

agregan otros parámetros como el tamaño de la imagen, el mapa de etiquetas y los pesos del entrenamiento como se indica en la Figura 61. Luego de esto se descarga el archivo con extensión tflite y el mapa de etiquetas

Figura 61

Conversión del modelo a formato tflite

```
!python export.py --weights runs/train/yolov5s_results/weights/best.pt --include tflite --int8 --img 416 --data dataset.yaml

export: data=dataset.yaml, weights=['runs/train/yolov5s_results/weights/best.pt'], imgsz=[416], batch_size=1, device=cpu, hal
YOLOv5 🚀 2022-10-4 Python-3.10.12 torch-2.0.1+cu118 CPU
```

Nota. Autoría propia

3.2.3. Métricas del entrenamiento

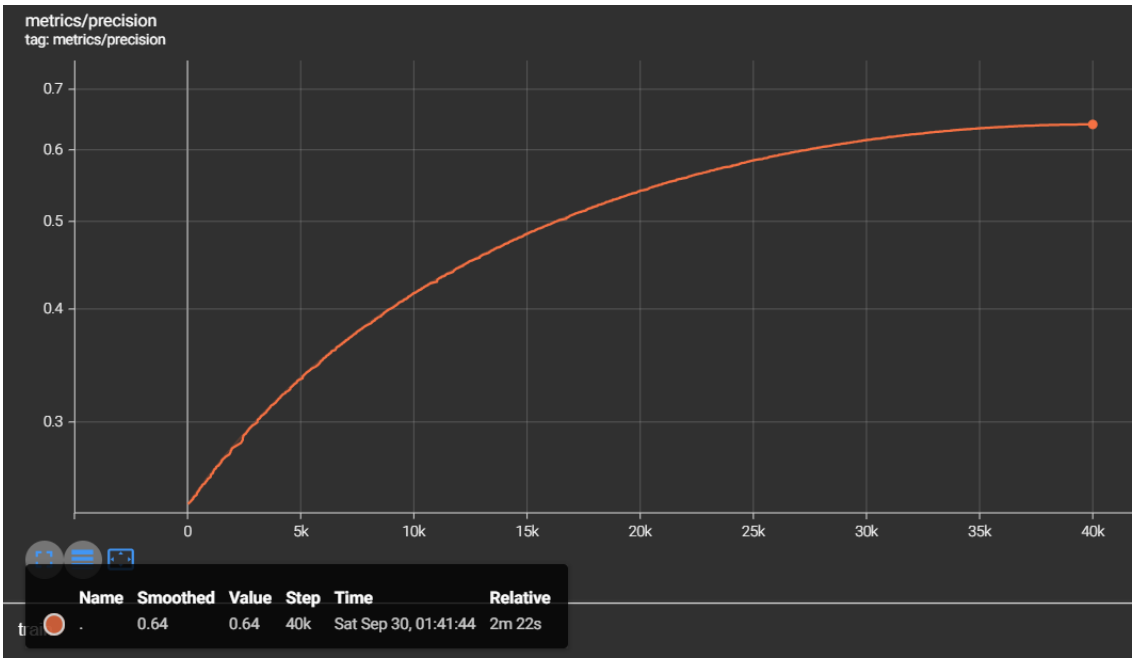
En este apartado se indica las diferentes métricas obtenidas a partir del entrenamiento de los modelos utilizando MobileNet-v2 y YOLOv5.

La métrica de presión describe la efectividad que tiene el modelo para la clasificación y la detección de los objetos, se puede observar en la Figura 62 (a) perteneciente a la arquitectura de MobileNet-v2 se tiene un crecimiento considerable hasta el paso 37000, sin embargo, después de este paso, tiende a mantener un solo valor aproximado, finalizando en 0.64. Ahora, para la precisión del modelo entrenado con la arquitectura de YOLOv5 se tiene un valor de crecimiento considerable hasta la época 180 y estabilizándose alrededor de un valor hasta la época 200, obteniendo un valor final de 0.9199 mirar la Figura 62 (b). Convirtiendo los valores a porcentaje se indica que YOLOv5 tiene una mejor precisión con un 91.99%, frente a MobileNet-v2 con 64% de precisión.

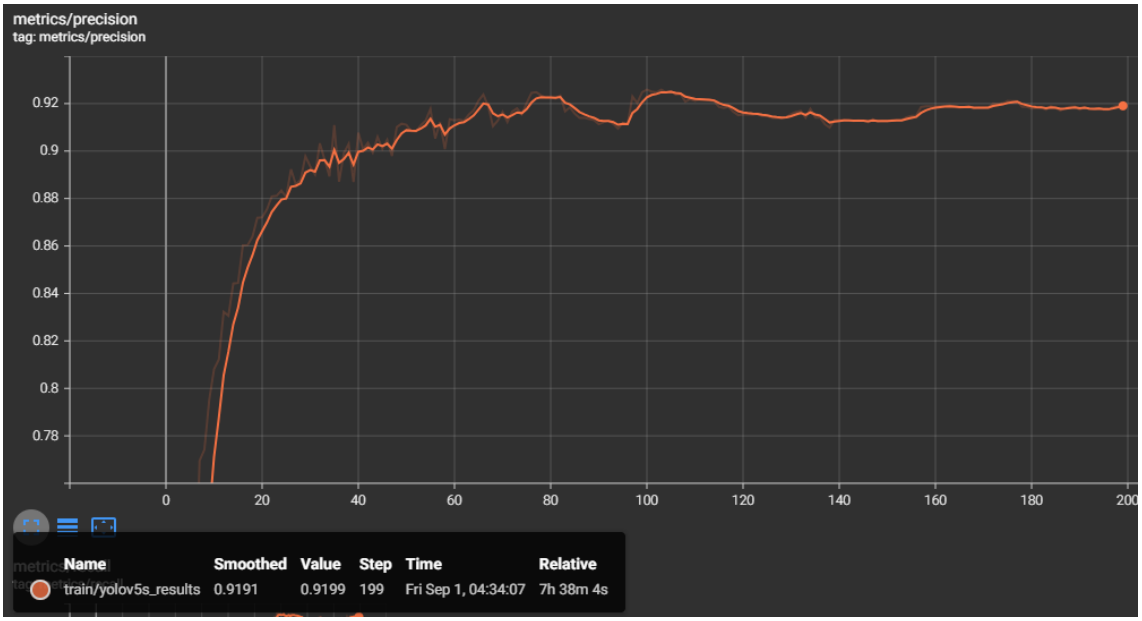
Figura 62

Gráficas de la precisión de los modelos entrenados

a) *Gráfica de la precisión del modelo entrenado con la arquitectura MobileNet-v2*



b) *Gráfica de la precisión del modelo entrenado con la arquitectura YOLOv5*



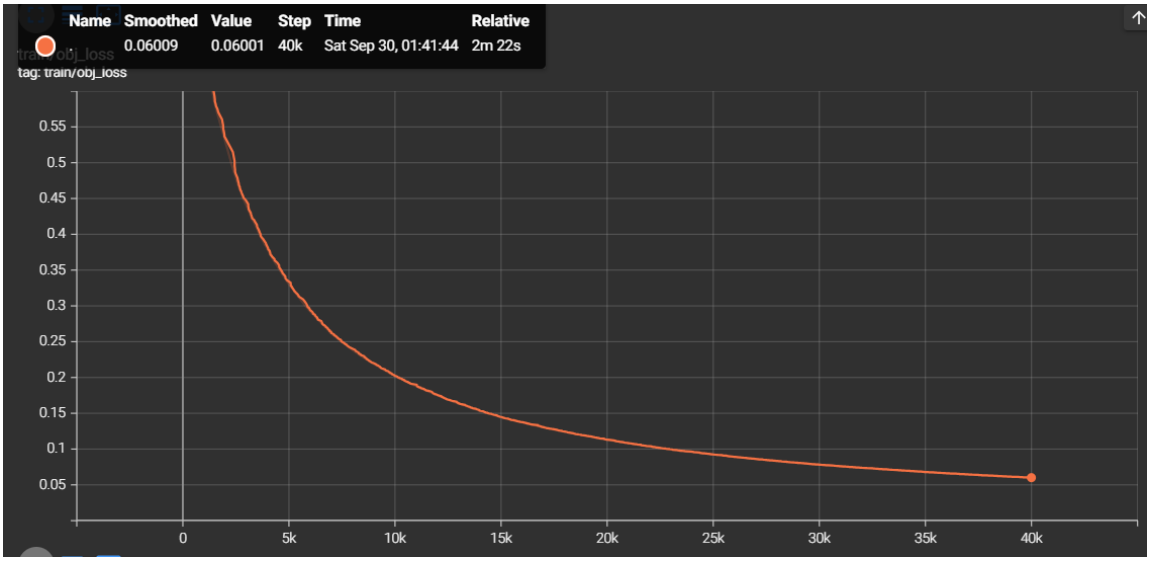
Nota. Autoría propia

Otra métrica resultante es la pérdida en la clasificación de los objetos, en la cual, para la arquitectura de MobileNet-v2 mostrada en la Figura 63 (a), se obtiene un valor final de 0.6001 al final de 40000 pasos del entrenamiento, mientras que para la arquitectura de YOLOv5 se tiene un valor final de 0.04823 en las 200 épocas de entrenamiento, observar Figura 63 (b).

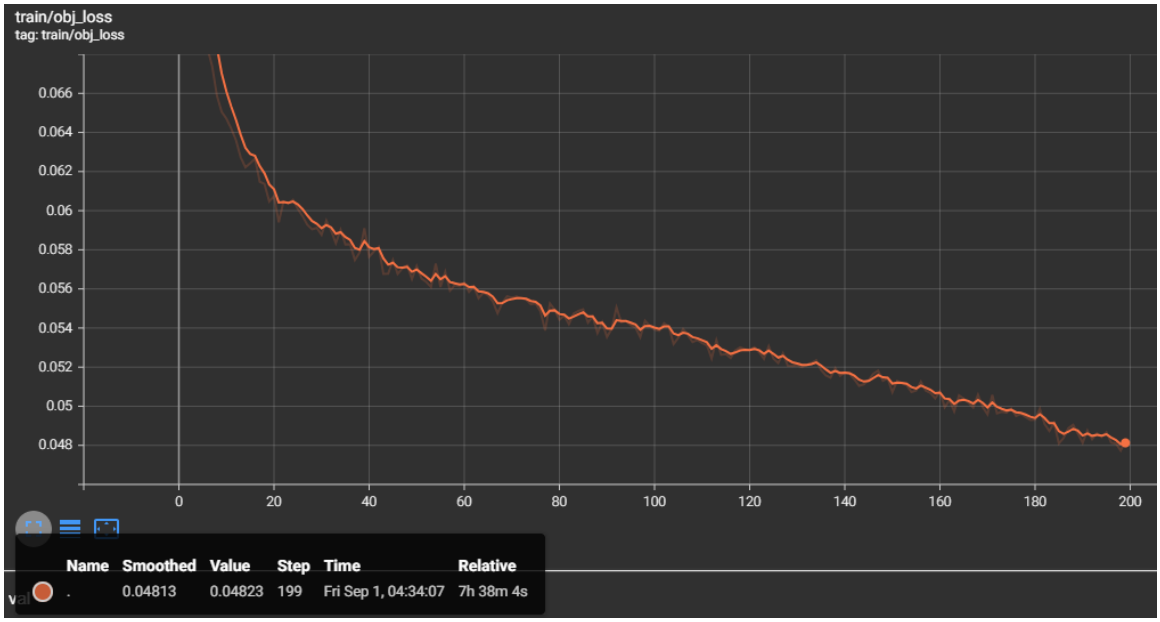
Figura 63

Gráficas de la pérdida en la clasificación de objetos de los modelos entrenados

a) Gráfica de pérdida en la clasificación de objetos del modelo entrenado con la arquitectura MobileNet-v2



b) Gráfica de pérdida en la clasificación de objetos del modelo entrenado con la arquitectura YOLOv5



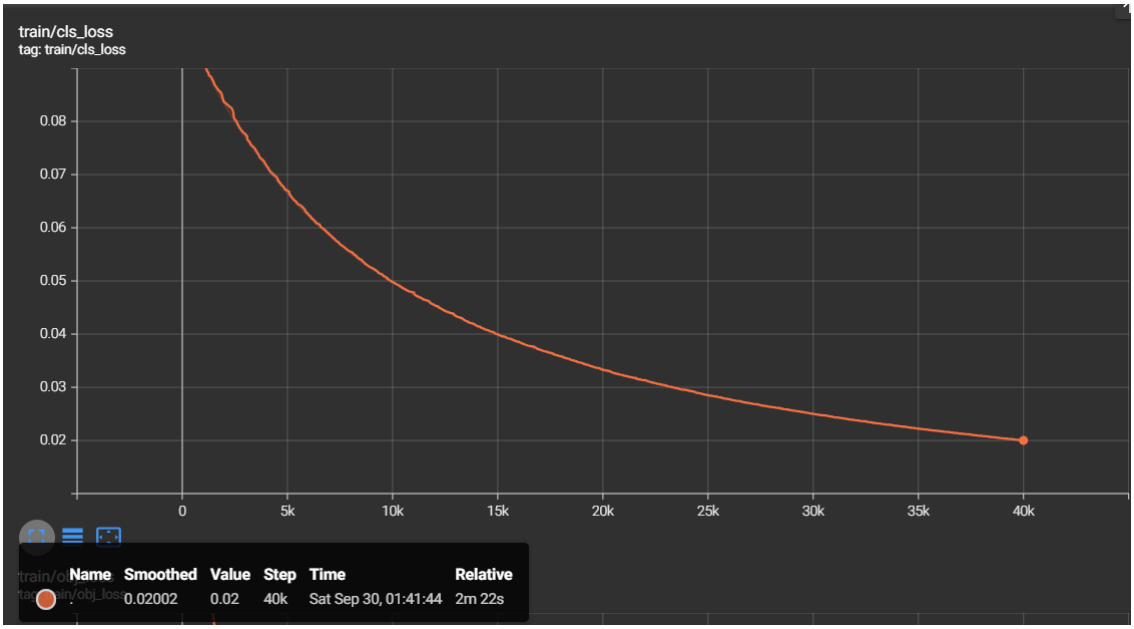
Nota. Autoría propia

En la métrica de pérdida clasificación de clases, se hace referencia a los errores entre las predicciones de las clases y los valores reales, MobileNet-v2 tiene un valor de pérdida que disminuye a lo largo del entrenamiento hasta obtener un valor final de 0.02, observar Figura 64 (a). Para YOLOv5 se tiene una pérdida de 0.004 al finalizar el entrenamiento, como se indica en la Figura 64 (b); en base a los resultados expuestos se indica que claramente YOLOv5 tiene menos errores de predicción de clases.

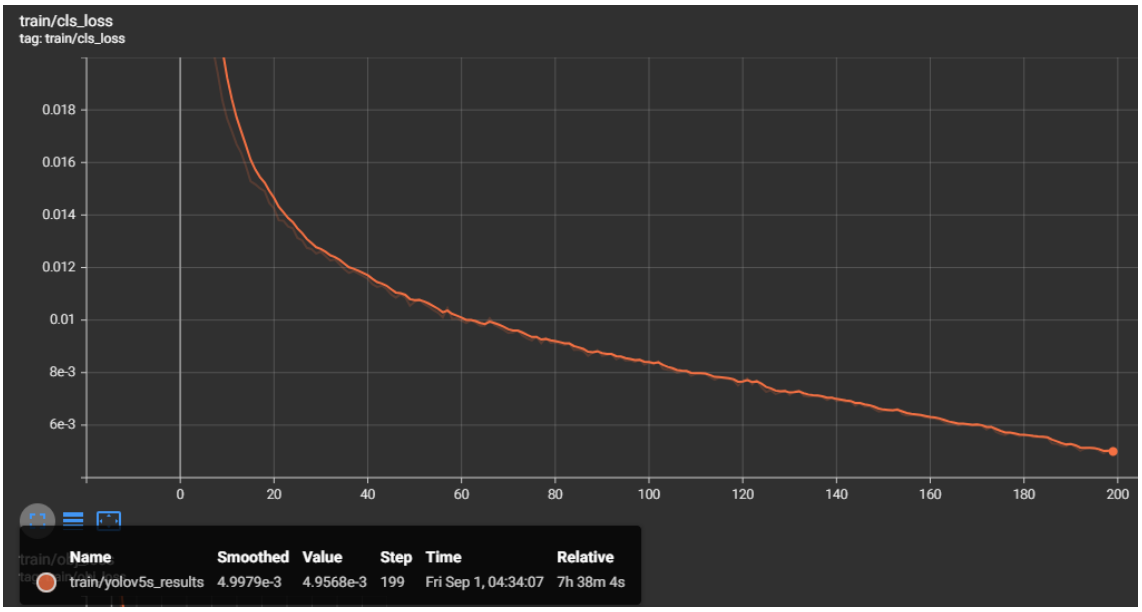
Figura 64

Gráficas de la pérdida en la clasificación de clases de los modelos entrenados

a) Gráfica de pérdida en la clasificación de clases del modelo entrenado con la arquitectura MobileNet-v2



b) Gráfica de pérdida en la clasificación de clases del modelo entrenado con la arquitectura YOLOv5



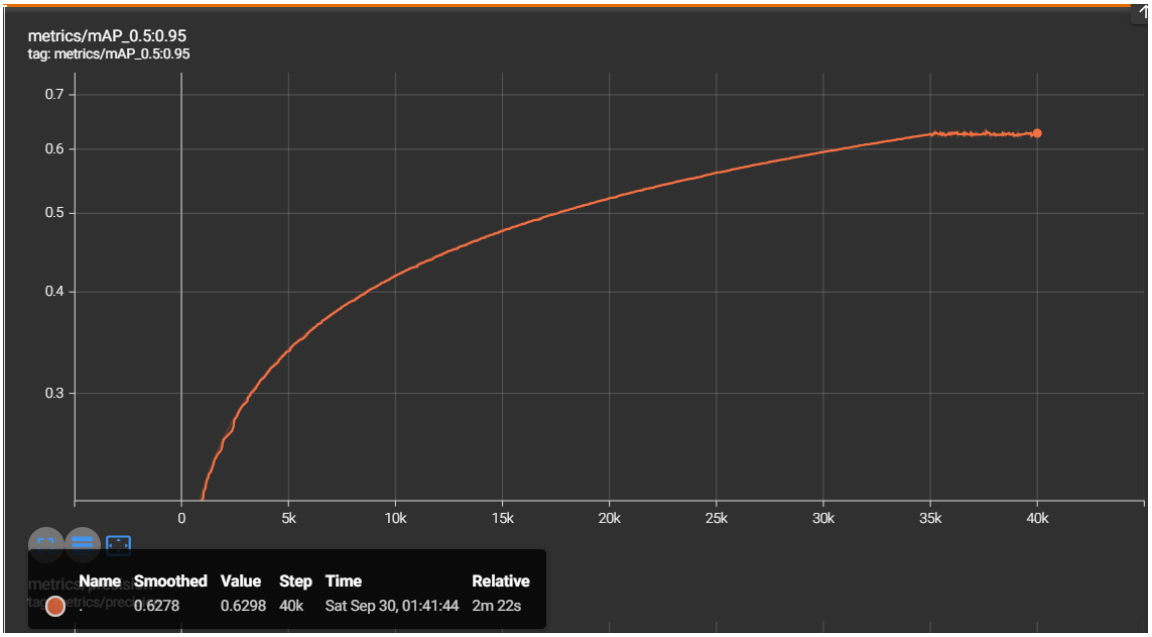
Nota. Autoría propia

Como última métrica obtenida a partir del entrenamiento, es el mAP con un valor de 0.6298 utilizando la arquitectura MobileNet-v2 mostrada en la Figura 65 (a) y un valor de 0.75 utilizando la arquitectura de YOLOv5 en la Figura 65 (b), este parámetro mide la detección global del modelo para todas las clases en un conjunto de datos. Cabe recalcar que el resultado de las gráficas se da partir del promedio de 0.5 a 0.95 de IoU, en donde se realiza un cálculo de área entre la unión de la caja delimitadora de la detección con la caja delimitadora etiquetada, sobre la intersección de estos dos mismos valores. Además, se indica que, la arquitectura de YOLOv5 tiene una mayor capacidad de detección sobre la arquitectura de MobileNet-v2.

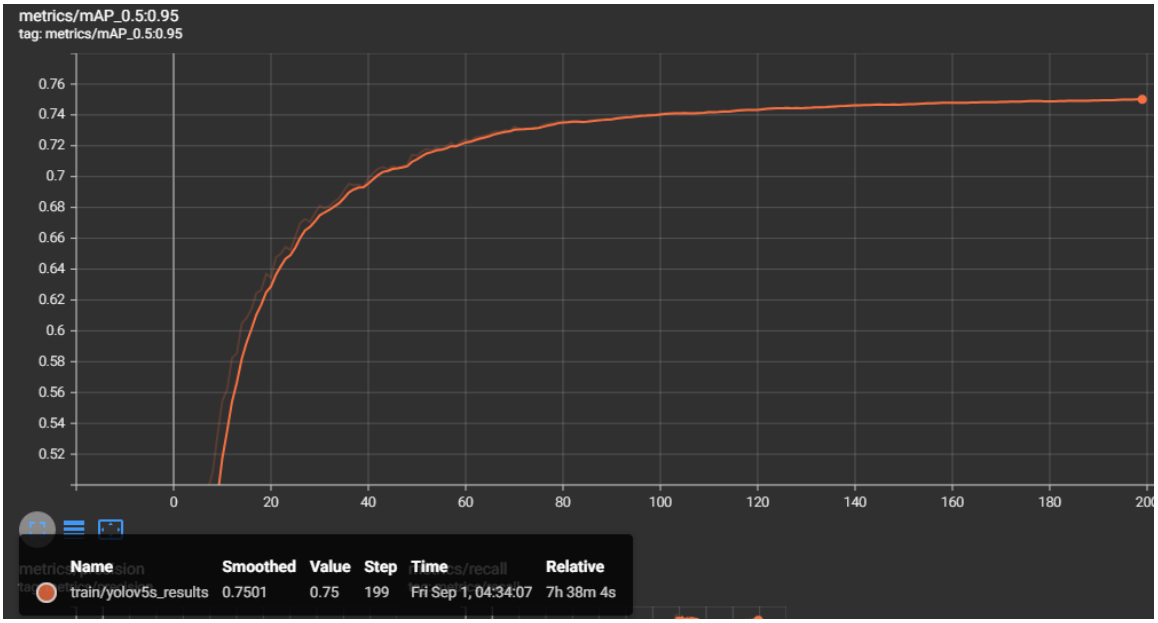
Figura 65

Gráficas de mAP de los modelos entrenados

a) Gráfica de mAP del modelo entrenado con la arquitectura MobileNet-v2



b) Gráfica de mAP del modelo entrenado con la arquitectura YOLOv5



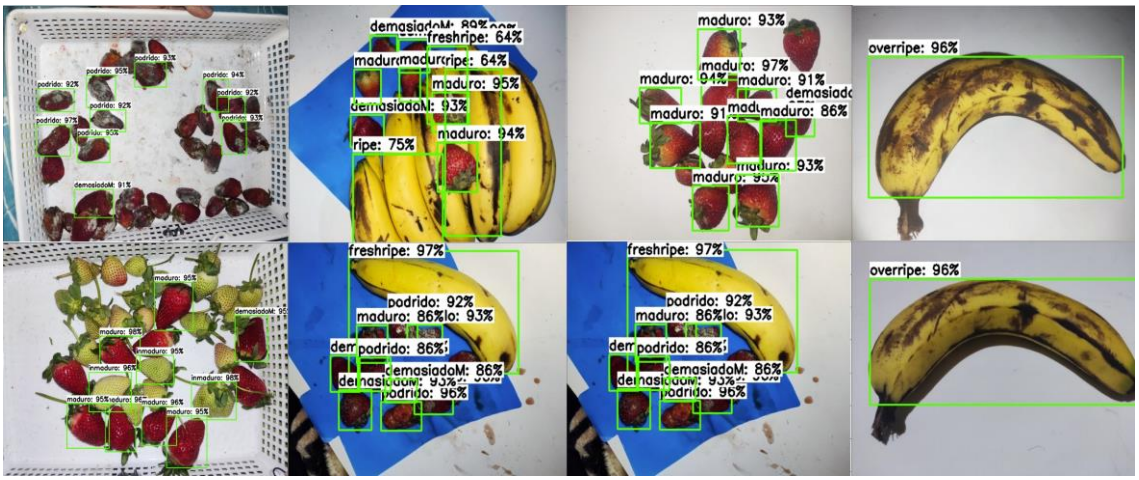
Nota. Autoría propia

Adicionalmente, se realiza la inferencia de varias imágenes de forma simultánea utilizando las imágenes de prueba del dataset. Utilizando MobileNet-v2 representado en la Figura 66 (a), se indica la detección y clasificación de objetos de manera aceptable, sin embargo, hay que resaltar que esta arquitectura tiene cierta deficiencia para detectar muchos objetos sobre una misma imagen. En la Figura 66 (b) YOLOv5 cumple un mejor desempeño en comparación a la arquitectura anterior, detectando y clasificando casi todos los objetos de una misma imagen, sin importar el número.

Figura 66

Inferencia de las imágenes de prueba con los modelos entrenados

a) *Inferencia de la imágenes de prueba con el modelo entrenado con la arquitectura MobileNet-v2*



b) *Inferencia de la imágenes de prueba con el modelo entrenado con la arquitectura YOLOv5*

Nota. Autoría propia

3.3. Desarrollo de Software

3.3.1. Script para la captura de imágenes

Para el script de captura de imágenes se empieza con la importación de las bibliotecas necesarias para el manejo de la cámara y el monitoreo de la hora.

Ahora siguiente con el script, para la captura de imágenes, en la Figura 67 se define dos funciones “encenderLED” y “apagarLED”, la primera, establece un modo de numeración de pines disponibles en las Raspberry, luego mediante la función “GPIO.setup”, se indica a los cada uno de los pines asignados en cada variable que actuen como una salida lógica, y como parte final de esta función se indica un estado “HIGH” para los pines que se ha definido, es decir, los pines se encienden. Ahora, para la segunda función unicamente mediante el comando “GPIO.output(LED#, GPIO.LOW)” se indica el apago de todos los leds y con la función GPIO.cleanup se inidca la liberación de los pines que se ha utilizado.

Figura 67

Definición de funciones para la iluminación

```
def encenderLED():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(LED1, GPIO.OUT)
    GPIO.setup(LED2, GPIO.OUT)
    GPIO.setup(LED3, GPIO.OUT)
    GPIO.setup(LED4, GPIO.OUT)
    GPIO.setup(LED5, GPIO.OUT)
    GPIO.setup(LED6, GPIO.OUT)
    GPIO.output(LED1, GPIO.HIGH)
    GPIO.output(LED2, GPIO.HIGH)
    GPIO.output(LED3, GPIO.HIGH)
    GPIO.output(LED4, GPIO.HIGH)
    GPIO.output(LED5, GPIO.HIGH)
    GPIO.output(LED6, GPIO.HIGH)

# Funcion para apagar las luces led
def apagarLED():
    GPIO.output(LED1, GPIO.LOW)
    GPIO.output(LED2, GPIO.LOW)
    GPIO.output(LED3, GPIO.LOW)
    GPIO.output(LED4, GPIO.LOW)
    GPIO.output(LED5, GPIO.LOW)
    GPIO.output(LED6, GPIO.LOW)
    GPIO.cleanup()
```

Nota. Autoría propia

En esta tercera y última parte del script, observar Figura 68, se define dos variables “t1” y “t2”, las cuales contienen las horas en que se va a realizar monitoreo. También se define una nueva función la cual va a estar constantemente obteniendo la hora, esto mediante la función “datetime.datetime.now” y dando un formato a la hora con la función “now.strftime”. Dentro de esta función también se ha establecido un condicional con el cual tiene como objetivo hacer “match” con las variables “t1” o “t2” finalmente si se cumple la condición, se borra todas las imágenes existentes en el directorio y a continuación se toma una fotografía utilizando el comando “libcamera-still -o /home/pi/yolo/imagen.jpg”, además, agregando otros parámetros como nivel de brillo y contraste. Y para finalizar se ejecuta el script “deteccion1.py” el cual permite la detección del grado de madurez de las frutas (plátanos y fresas), cabe recalcar que para esta ejecución se agregan argumentos como: el modelo en formato tflite, la dimensión de la imagen de entrada, el porcentaje de confianza mínimo, la imagen que se procesará y el mapa de etiquetas. Esta función se repite de forma cíclica cada segundo con la finalidad de obtener un monitoreo constante.

Figura 68

Función de tiempo para la ejecución de la detección cada cierta hora

```
t1 = "07:00:00"
t2 = "19:00:00"
#--brightness=0.2

def tiempo():
    while True:
        now = datetime.datetime.now()
        time = now.strftime("%H:%M:%S")
        print(time)
        if time == t1 or time == t2:
            encenderLED()
            # Captura de imagen por medio de la camara Raspberry
            system("libcamera-still --brightness=0.27 --contrast=1.3 -o /home/pi/yolo/imagen.jpg")
            apagarLED()
            system("python3 deteccion1.py --weights best-int8.tflite --img 416 --conf 0.4 --source imagen.jpg --data dataset.yaml")
        sleep(1)

tiempo()
```

Nota. Autoría propia

3.3.2. Script de proceso de detección de imagen

El apartado que se describe a continuación se enfoca principalmente en las partes de detección de los objetos dentro de las imágenes, en el Anexo 1, se encuentra todo el código fuente comentado para mayor información.

Este script fue modificado a partir del archivo de detección proporcionado desde el repositorio de YOLOv5, inicialmente se importa las librerías necesarias para este archivo y se agrega las credenciales para la conexión con la base de datos; en esta parte del script que es encuentra en la Figura 69, se identifica todos los parámetros que se van a utilizar para la detección, de manera puntual, al utilizar únicamente una imagen se cambia el apartado denominado “source”, también se modifica la variable “weights”, es decir, los pesos del modelo que se ha entrenado, además se indica el porcentaje de confianza para la detección de 40% y un valor de confianza de IoU de 0.45, y finalmente en “project”, se establece el directorio para guardar los resultados de la imagen con las detecciones realizadas.

Figura 69

Parámetros de la detección

```
# Definición de parámetros de la detección
@smart_inference_mode()
def run(
        weights=ROOT / 'best.pt',  # Ruta del modelo
        source=ROOT / 'imagen.jpg',  # Archivo de imagen
        data=ROOT / 'data/coco128.yaml',  # Ruta del archivo dataset.yaml
        imgsz=(640, 640),  # Tamaño de la imagen para inferencia (alto, ancho)
        conf_thres=0.4,  # Umbral de confianza (confianza mínima) 0.25
        iou_thres=0.45,  # Umbral de IoU
        max_det=1000,  # Detecciones máximas por imagen
        device='',  # Dispositivo de procesamiento (cuda, por ejemplo, 0 para GPU o 'cpu' para CPU)
        view_img=False,  # Mostrar resultados
        save_txt=False,  # Guardar resultados en un archivo *.txt
        save_conf=False,  # Guardar confianzas en --save-txt etiquetas
        save_crop=False,  # Guardar recortes de predicciones
        nosave=False,  # No guardar imágenes/videos
        classes=None,  # Filtrar por clase: --class 0, o --class 0 2 3
        agnostic_nms=False,  # NMS (Supresión de No Máximo) // detección de objetos redundantes
        augment=False,  # Inferencia aumentada
        visualize=False,  # Visualizar características
        update=False,  # Actualizar todos los modelos
        project=ROOT / 'runs/detect',  # Guardar resultados en la carpeta del proyecto/nombre runs/detect
        name='exp',  # Nombre del proyecto/nombre para guardar resultados #exp
        exist_ok=False,  # Si el proyecto/nombre existe, no incrementarlo
        line_thickness=3,  # Grosor de línea del cuadro delimitador (píxeles)
        hide_labels=False,  # Ocultar etiquetas
        hide_conf=False,  # Ocultar confianzas
        half=False,  # Usar inferencia en media precisión FP16
):
```

Nota. Autoría propia

En la Figura 70, se convierte a una cadena string el nombre de la imagen que se ingresa, luego se define el formato de las imágenes y luego se comprueba la existencia del archivo. A continuación, se carga el modelo, con los pesos que se ha entrenado "DetectMultibackend(weights)" y también se verifica la dimensión de las imágenes que se va a procesar

Figura 70

Carga de imágenes, directorios y modelo

```
# Carga de imagenes
source = str(source)
save_img = not nosave and not source.endswith('.txt')
is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
if is_url and is_file:
    source = check_file(source)

# Directorios
save_dir = increment_path(Path(project) / name, exist_ok=exist_ok)
(save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  # crea un directorio

# Cargar modelo
device = select_device(device)
model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data, fp16=half)
stride, names, pt = model.stride, model.names, model.pt
imgsz = check_img_size(imgsz, s=stride)  # verifica el tamaño de la imagen
```

Nota. Autoría propia

Para la Figura 71, se realiza la inferencia de la imagen cargada, a partir del modelo entrenado, esto se destaca en la línea de código “pred = model(im, augment=augment, visualize=visualize)”. Además, este script también realiza un proceso denominado “NMS”, el cual sirve para reducir las detecciones redundantes, esto se realiza a partir de la predicción anterior, tomando en cuenta los valores de confiabilidad (conf_thres), clases (classes), elementos máximos detectados por imagen (max_dent) y valor de confiabilidad en las cajas delimitadoras (iou_thres).

Figura 71

Inferencia en la imagen

```
# Inferencia
with dt[1]:
    visualize = increment_path(save_dir / Path(path).stem, mkdir=True) if visualize else False
    pred = model(im, augment=augment, visualize=visualize)

# NMS (Reducción de detecciones redundantes)
with dt[2]:
    pred = non_max_suppression(pred, conf_thres, iou_thres, classes, agnostic_nms, max_det=max_det)
```

Nota. Autoría propia

Si existe detección de objetos dentro de la imagen procesada a continuación en la Figura 72, se realiza un reescalamiento de las cajas delimitadoras, “det” contiene resultados de detección de objetos y ajusta las coordenadas de las cajas delimitadoras en función del tamaño de la imagen original “im0”. Esto se logra escalando las coordenadas de las cajas en “det” de acuerdo con las dimensiones de la imagen original, esto realizado utilizando la función “shape”, y redondeando los valores con la función “round()”.

Figura 72

Trazado de las cajas delimitadoras

```
if len(det):
    # Rescalamiento de las cajas delimitadoras
    det[:, :4] = scale_boxes(im.shape[2:], det[:, :4], im0.shape).round()

    # Impresión de los resultados
    for c in det[:, 5].unique():
        n = (det[:, 5] == c).sum()  # Detecciones por clase
        s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # Adición del nombre de la clase

    # Trazado de las cajas delimitadoras
    for *xyxy, conf, cls in reversed(det):
        if save_txt:  # Condicional para la escritura de las cajas delimitadoras
            xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()
            line = (cls, *xywh, conf) if save_conf else (cls, *xywh)  # Formato de etiqueta
            with open(f'{txt_path}.txt', 'a') as f:
                f.write(('%g ' * len(line)).rstrip() % line + '\n')

        if save_img or save_crop or view_img:  # Escritura en el archivo de imagen
```

Nota. Autoría propia

Para definir un color para cada estado de madurez, ya sea de fresas o plátanos, se ha establecido varios condicionales en donde, se compara de acuerdo a la etiqueta que se obtiene en la detección de la imagen, asignado un color diferente de acuerdo al estado de madurez que se obtenga, en la Figura 73, se muestran estos condicionales, además, se guardan los valores dentro de una variable para la condición (estadoMadurez#).

Figura 73

Condicionales según el estado de madurez de las frutas

```
    # Condicionales según el estado de madurez
    if names[c]== "unripe" or names[c] == "freshunripe" or names[c] == "freshripe"
        or names[c]== "ripe" or names[c] == "overripe" or names[c]== "rotten":
        tipoFruta1 = "Platanos"
        db.child("datos/tipoFruta1").set(tipoFruta1)
        if names[c] == "unripe":
            annotator.box_label(xyxy, label, (255, 0, 255))
            estadoMadurez1 = "Inmaduro"
        elif names[c] == "freshunripe":
            annotator.box_label(xyxy, label, (0, 100, 0))
            estadoMadurez2 = "Fresco inmaduro"
        elif names[c] == "freshripe":
            annotator.box_label(xyxy, label, (128, 0, 128))
            estadoMadurez3 = "Fresco"
        elif names[c] == "ripe":
            annotator.box_label(xyxy, label, (0, 102, 255))
            estadoMadurez4 = "Maduro"
        elif names[c] == "overripe":
            annotator.box_label(xyxy, label, (255, 255, 0))  #(235, 206, 135)
            estadoMadurez5 = "Maduracion excesiva"
        elif names[c]== "rotten":
            annotator.box_label(xyxy, label, (0, 0, 255))
            estadoMadurez6 = "Podrido"
    if names[c] == "inmaduro" or names[c]== "maduro" or names[c]== "demasiadoM" or names[c] == "podrido":
        tipoFruta2 = "Fresas"
        #db.child("datos/tipoFruta2").set(tipoFruta2)
        print(tipoFruta2)
        if names[c]== "inmaduro":
            annotator.box_label(xyxy, label, (140, 0, 179))
            estadoMadurez7 = "Inmaduro"
        elif names[c]== "maduro":
            annotator.box_label(xyxy, label, (128, 0, 0))
            estadoMadurez8 = "Maduro"
        elif names[c] == "demasiadoM":
            annotator.box_label(xyxy, label, (42, 42, 165))
            estadoMadurez9 = "Demasiado maduro"
        elif names[c] == "podrido":
            annotator.box_label(xyxy, label, (51, 51, 51))
            estadoMadurez10 = "Podrido"
```

Nota. Autoría propia

Finalmente, se guarda los resultados de la detección en conjunto con las imágenes, esto se realiza mediante las utilidades de OpenCV, dentro de este mismo apartado se ha colocado la actualización de los valores detectados de la imagen a la base de datos, observar Figura 74.

Figura 74

Actualización de la base de datos

```
# Guardar resultados de la detección
if save_img:
    if dataset.mode == 'image':

        # Actualizar los datos en Firebase
        datos = {"estadoMadurez1": estadoMadurez1, "estadoMadurez2": estadoMadurez2, "estadoMadurez3": estadoMadurez3,
                 "estadoMadurez4": estadoMadurez4,"estadoMadurez5": estadoMadurez5, "estadoMadurez6": estadoMadurez6,
                 "estadoMadurez7": estadoMadurez7, "estadoMadurez8": estadoMadurez8, "estadoMadurez9": estadoMadurez9,
                 "estadoMadurez10": estadoMadurez10, "tipoFruta1": tipoFruta1, "tipoFruta1": tipoFruta1, "fecha_hora": fecha_hora}
        # Escritura en la base de datos
        db.child("datos").update(datos)
        # Almacenamiento de la imagen con las detecciones
        cv2.imwrite(save_path, im0)
```

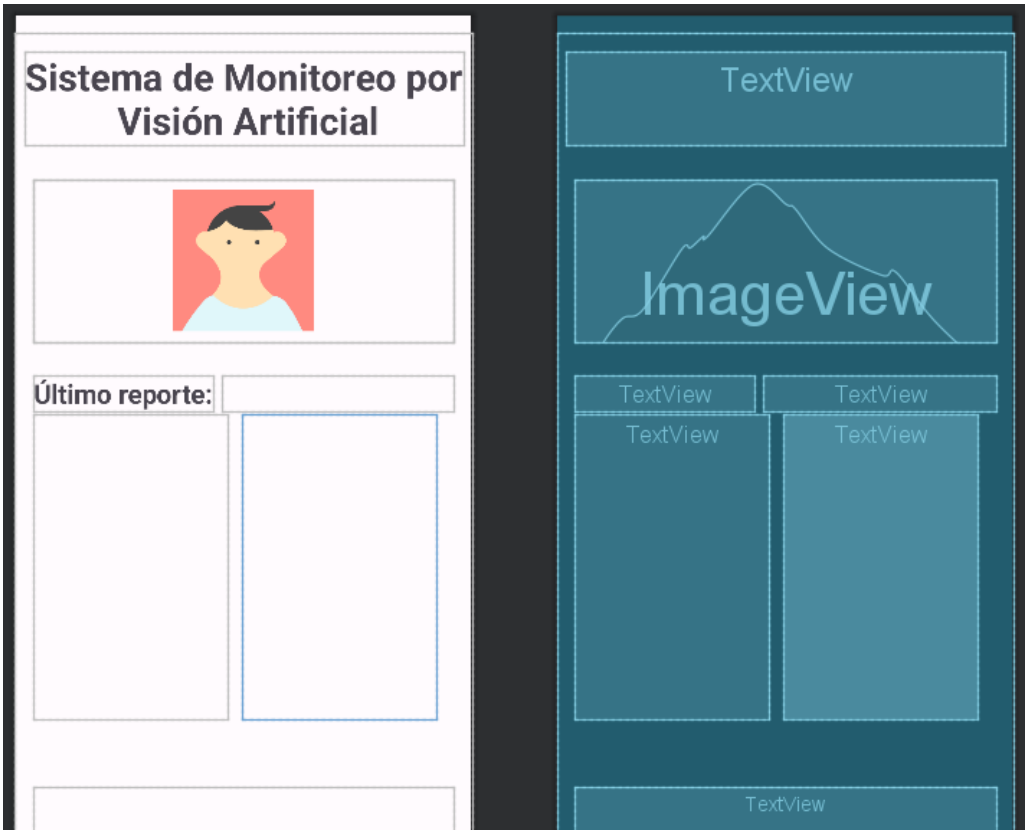
Nota. Autoría propia

3.3.3. Aplicación Móvil

En la sección de aplicación móvil se describen principalmente las principales funciones de la aplicación, en caso de requerir información más detallada de la aplicación, se encuentra en la Anexo 2. Para el desarrollo de la aplicación móvil la cual permitirá la visualización del sistema embebido de monitoreo se ha realizado por medio del software de Android Studio. Como paso inicial se indica la creación de la interfaz del usuario, mirar Figura 75, en este plano se agrega cuadros de visualización de texto (TextView) y cuadros de visualización de imagen (ImageView), de forma específica se han agregado 6 “TextView” y 1 “ImageView”, esto con la finalidad de presentar todos los datos enviados desde script de detección del grado de madurez de las frutas.

Figura 75

Estructura gráfica de la aplicación



Nota. Autoría propia

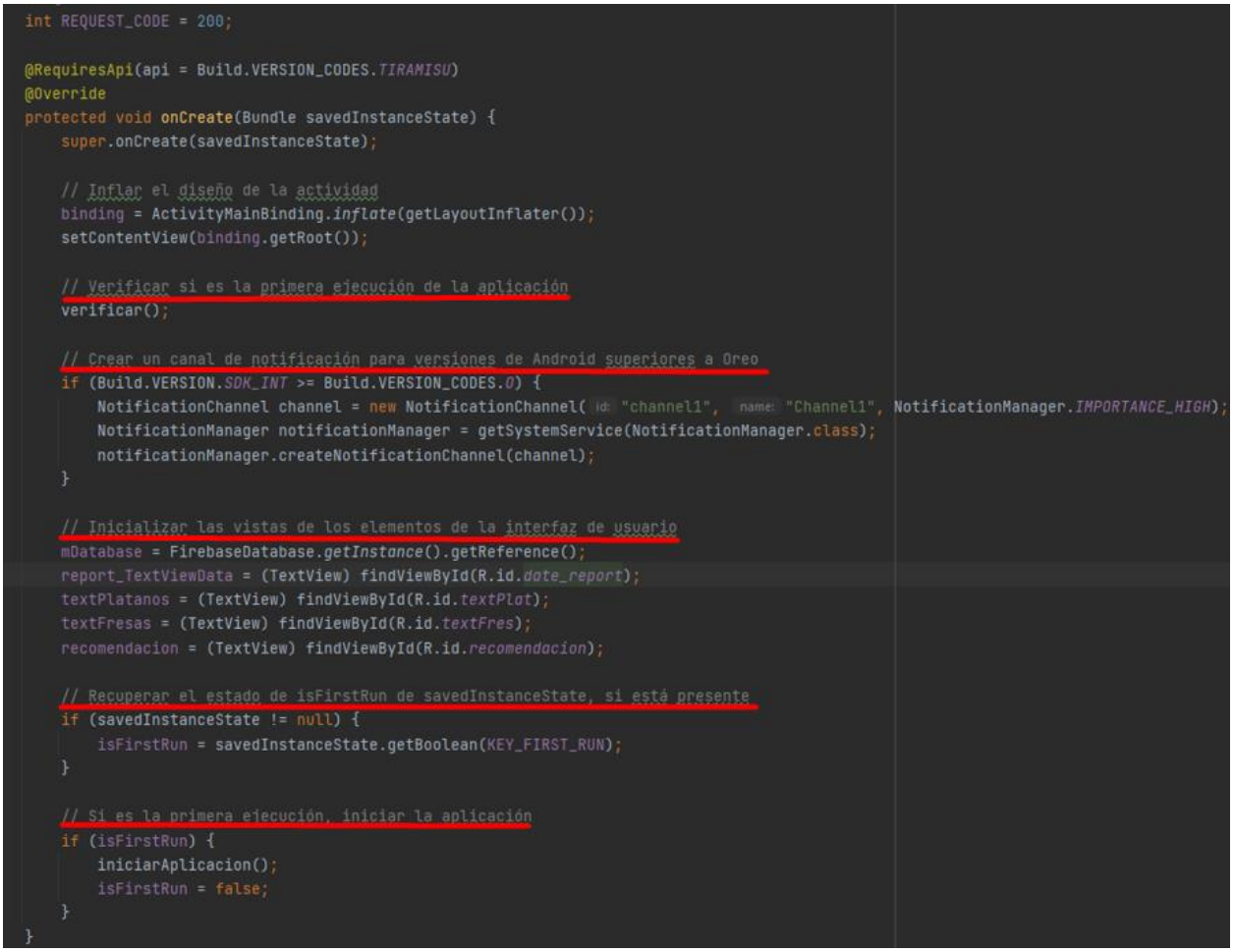
Para la parte lógica de la aplicación inicialmente se realiza la importación de todas las librerías y dependencias necesarias para el proyecto. También se toma en cuenta la asignación de las librerías dependiendo el objeto que se utilizará.

Continuando con la descripción de la parte lógica de la aplicación, en la Figura 76 se indica la función OnCreate, la cual es la primera función que se ejecuta al iniciar la aplicación. Dentro de esta aplicación se ha agregado un condicional para verificar la versión de la API de Android, esto con el fin para la creación de canales de notificación que se usará a posterior dentro de la lógica de esta actividad. Además, se indica el uso de la función “verificar”, la cual se usa para comprobar

los permisos de la aplicación con respecto al acceso a notificaciones. Luego se castea las variables con respecto a los cuadros de visualización de texto e imagen, es decir, se establece la relación de conexión con las variables que se creó anteriormente con los cuadros de texto y de imagen. Para finalizar esta función principal se define otro condicional “savedInstanceState” con un valor nulo, para lo cual se guarda con un valor de primera vez (“isFirstRun”) cuando se inicia la aplicación. Y si la aplicación ha sido iniciada se ejecuta la función “iniciarAplicacion” y se retorna a un valor falso la variable anterior.

Figura 76

Definición de condicionales para el inicio de la aplicación



```
int REQUEST_CODE = 200;

@RequiresApi(api = Build.VERSION_CODES.TIRAMISU)
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Inflar el diseño de la actividad
    binding = ActivityMainBinding.inflate(getLayoutInflater());
    setContentView(binding.getRoot());

    // Verificar si es la primera ejecución de la aplicación
    verificar();

    // Crear un canal de notificación para versiones de Android superiores a Oreo
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        NotificationChannel channel = new NotificationChannel( id: "channel1",  name: "Channel1", NotificationManager.IMPORTANCE_HIGH);
        NotificationManager notificationManager = getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
    }

    // Inicializar las vistas de los elementos de la interfaz de usuario
    mDatabase = FirebaseDatabase.getInstance().getReference();
    report_TextViewData = (TextView) findViewById(R.id.date_report);
    textPlatanos = (TextView) findViewById(R.id.textPlat);
    textFresas = (TextView) findViewById(R.id.textFres);
    recomendacion = (TextView) findViewById(R.id.recomendacion);

    // Recuperar el estado de isFirstRun de savedInstanceState, si está presente
    if (savedInstanceState != null) {
        isFirstRun = savedInstanceState.getBoolean(KEY_FIRST_RUN);
    }

    // Si es la primera ejecución, iniciar la aplicación
    if (isFirstRun) {
        iniciarAplicacion();
        isFirstRun = false;
    }
}
```

Nota. Autoría propia

Ahora para la función que notifica al usuario el estado de “podrido” en las frutas monitoreadas se representa en la Figura 77, se indicia definiendo el nombre de la función, a continuación, se define un nuevo objeto “NotificationCompat.Builder” junto con la creación del canal de notificaciones, después se le asignan atributos a la notificación como: Cuerpo (“setContentText”), título (“sentContenTittle”), autocancelación (setAutoCancel), color (setColor), entre otros atributos. Luego se crea un objeto “Intent” la cual indica principalmente hacia la clase principal, denominada “MainActivity”, además también se crea un objeto “PendingIntent” el cual tendrá como atributo a los valores de la aplicación, todo esto para finalmente construir o llamar de nuevo a la aplicación cuando se presione sobre la notificación de alerta.

Figura 77

Código para la creación de la notificación de "podrido"

```
// Método para mostrar una notificación de frutas podridas
public void notificationRotten(View view) {
    NotificationCompat.Builder builder = new NotificationCompat.Builder( context: this,  channelId: "channel1")
            .setSmallIcon(R.drawable.baseline_notifications_24)
            .setContentTitle("Sistema de Visión Artificial")
            .setContentText("Estado de madurez de las frutas")
            .setAutoCancel(true)
            .setColor(Color.RED)
            .setStyle(new NotificationCompat.BigTextStyle()
                    .setBigContentTitle("Estado de madurez de las frutas")
                    .bigText( cs: "Alerta el estado de las frutas es podrido"));
    NotificationManager notificationManager = getSystemService(NotificationManager.class);

    Intent intent = new Intent(getApplicationContext(), MainActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);

    PendingIntent pendingIntent = PendingIntent.getActivity(getApplicationContext(),
            requestCode: 0, intent, PendingIntent.FLAG_MUTABLE);
    builder.setContentIntent(pendingIntent);

    notificationManager.notify( id: 1, builder.build());
}
```

Nota. Autoría propia

Para las notificaciones para las frutas que tengan un estado de madurez excesivo o demasiado maduro, como se indica en la Figura 78, es el mismo proceso de estructuración de la función anterior, sin embargo, se han modificado ciertos valores como, el contenido del cuerpo de la notificación y el color de la misma, la cual es verde.

Figura 78

Código para la creación de la notificación de "demasiado maduro"

```
// Método para mostrar una notificación de frutas demasiado maduras
public void notificationOverripe(View view) {
    NotificationCompat.Builder builder = new NotificationCompat.Builder( context: this,  channelId: "channel1")
            .setSmallIcon(R.drawable.baseline_notifications_24)
            .setContentTitle("Sistema de Visión Artificial")
            .setContentText("Estado de madurez de las frutas")
            .setAutoCancel(true)
            .setColor(Color.GREEN)
            .setStyle(new NotificationCompat.BigTextStyle()
                    .setBigContentTitle("Estado de madurez de las frutas")
                    .bigText( cs: "Alerta el estado de las frutas es demasiado maduro"));
    NotificationManager notificationManager = getSystemService(NotificationManager.class);

    Intent intent = new Intent(getApplicationContext(), MainActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);

    PendingIntent pendingIntent = PendingIntent.getActivity(getApplicationContext(),
            requestCode: 0, intent, PendingIntent.FLAG_MUTABLE);
    builder.setContentIntent(pendingIntent);

    notificationManager.notify( id: 1, builder.build());
}
```

Nota. Autoría propia

Para el método o función “verificar”, se tiene la siguiente lógica presentada en la Figura 79 en donde la variable “permisosNotificaciones”, comprueba en el archivo “Manifest” los permisos para acceso a notificaciones que tiene la aplicación. En caso de no se tenga los permisos de

notificación. A través de la función, “requestPermissions” se posibilita la accion de permitir al usuario escoger el activar el acceso o no, los permisos para notificaciones en Android.

Figura 79

Condicionales para solicitar los permisos de notificaciones

```
// Método para verificar los permisos de notificación
1 usage
@RequiresApi(api = Build.VERSION_CODES.TIRAMISU)
public void verificar() {
    int permisosNotificaciones = ContextCompat.checkSelfPermission( context: this, Manifest.permission.POST_NOTIFICATIONS);

    if (permisosNotificaciones == PackageManager.PERMISSION_GRANTED) {
        Toast.makeText( context: this,  text: "Permisos de notificaciones concedidas", Toast.LENGTH_SHORT).show();
    } else {
        requestPermissions(new String[]{Manifest.permission.POST_NOTIFICATIONS}, REQUEST_CODE);
    }
}
```

Nota. Autoría propia

Para el método “iniciarAplicación” visualizado en la Figura 80, iniciando se construye un objeto de tipo “ProgressDialog”, al cual, se le define atributos como: un mensaje (setMessage), la cancelación descativada (setCancelable(false)). Después se define una variable “imagenID”, con el nombre de la imagen que se requiere descargar; mediante la función “FirebaseStorage.*getInstance*().getReference("images/"+imageID+".jpg")” se obtiene la imagen desde la base de datos de Firebase. Además, también se extraen los valores de las etiquetas desde Firebase Database Realtime con la función “addValueEvenListener”, indicando el apartado que se requiere adquirir, todo esto usando la función “child”.

Figura 80

Código para la barra de progreso

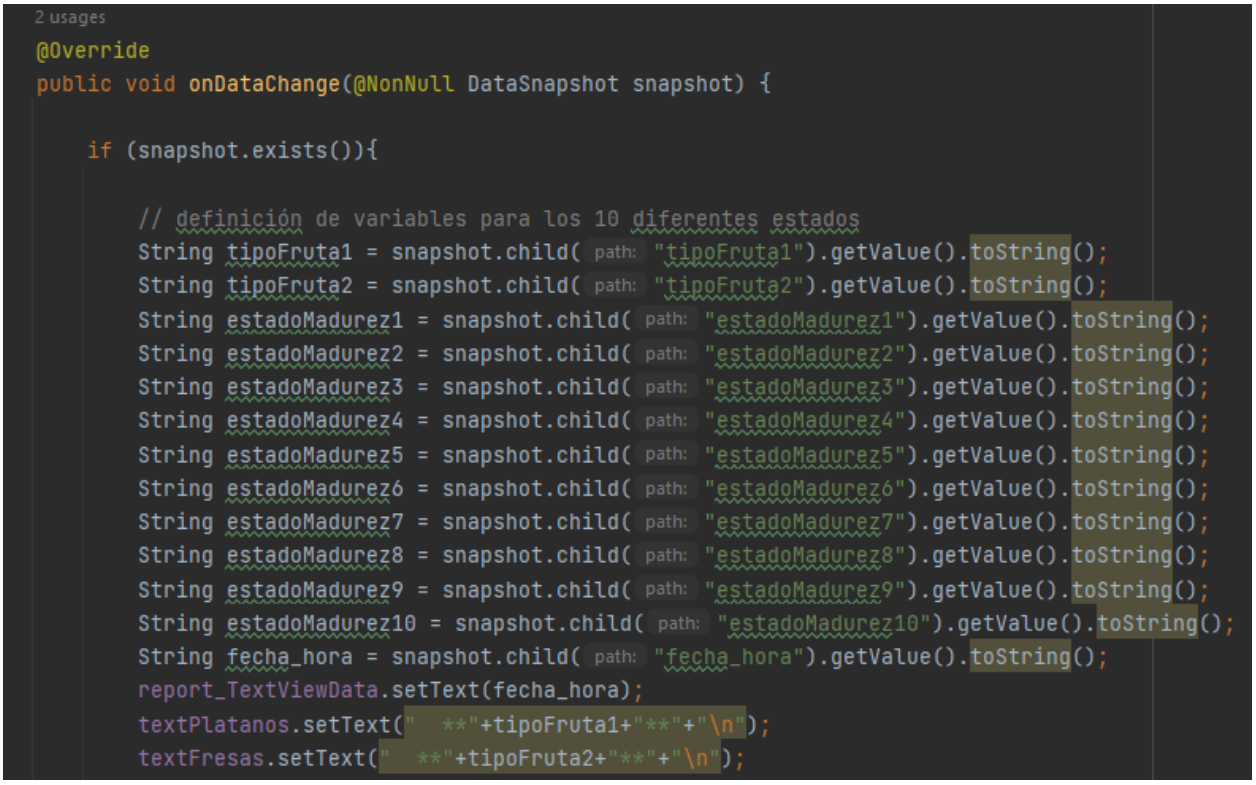
```
// Método para iniciar la aplicación y obtener los datos de Firebase
2 usages
@RequiresApi(api = Build.VERSION_CODES.TIRAMISU)
public void iniciarAplicacion() {
    progressDialog = new ProgressDialog( context: MainActivity.this);
    progressDialog.setMessage("Obteniendo reporte... Por favor espere.");
    progressDialog.setCancelable(false);
    progressDialog.show();

    String imageID = "imagen";

    storageReference = FirebaseStorage.getInstance().getReference( location: "images/" + imageID + ".jpg");

    mDatabase.child( pathString: "datos").addValueEventListener(new ValueEventListener() {
```

Nota. Autoría propia

Para verificar los cambios que se realicen en la base datos de Firebase se introduce un método denominado “onDataChange”, el cual se muestra en la Figura 81. Dentro de este método se agrega un condicional para comprobar si la base de datos que se desea obtener existe; a partir las funciones “snapshot.child("Valor a obtener").getValue().toString()” se consigue extraer los valores especificados de la base de datos, además de convertirlos a un formato string, y para finalizar esta sección del método se envía algunos valores ya adquiridos hacia los cuadros de visualización de texto.

Figura 81

Extracción de los datos desde la base de datos



```
2 usages
@Override
public void onDataChange(@NonNull DataSnapshot snapshot) {

    if (snapshot.exists()){

        // definición de variables para los 10 diferentes estados
        String tipoFruta1 = snapshot.child( path: "tipoFruta1").getValue().toString();
        String tipoFruta2 = snapshot.child( path: "tipoFruta2").getValue().toString();
        String estadoMadurez1 = snapshot.child( path: "estadoMadurez1").getValue().toString();
        String estadoMadurez2 = snapshot.child( path: "estadoMadurez2").getValue().toString();
        String estadoMadurez3 = snapshot.child( path: "estadoMadurez3").getValue().toString();
        String estadoMadurez4 = snapshot.child( path: "estadoMadurez4").getValue().toString();
        String estadoMadurez5 = snapshot.child( path: "estadoMadurez5").getValue().toString();
        String estadoMadurez6 = snapshot.child( path: "estadoMadurez6").getValue().toString();
        String estadoMadurez7 = snapshot.child( path: "estadoMadurez7").getValue().toString();
        String estadoMadurez8 = snapshot.child( path: "estadoMadurez8").getValue().toString();
        String estadoMadurez9 = snapshot.child( path: "estadoMadurez9").getValue().toString();
        String estadoMadurez10 = snapshot.child( path: "estadoMadurez10").getValue().toString();
        String fecha_hora = snapshot.child( path: "fecha_hora").getValue().toString();
        report_TextViewData.setText(fecha_hora);
        textPlatanos.setText("  **"+tipoFruta1+"**"+"\n");
        textFresas.setText("  **"+tipoFruta2+"**"+"\n");
```

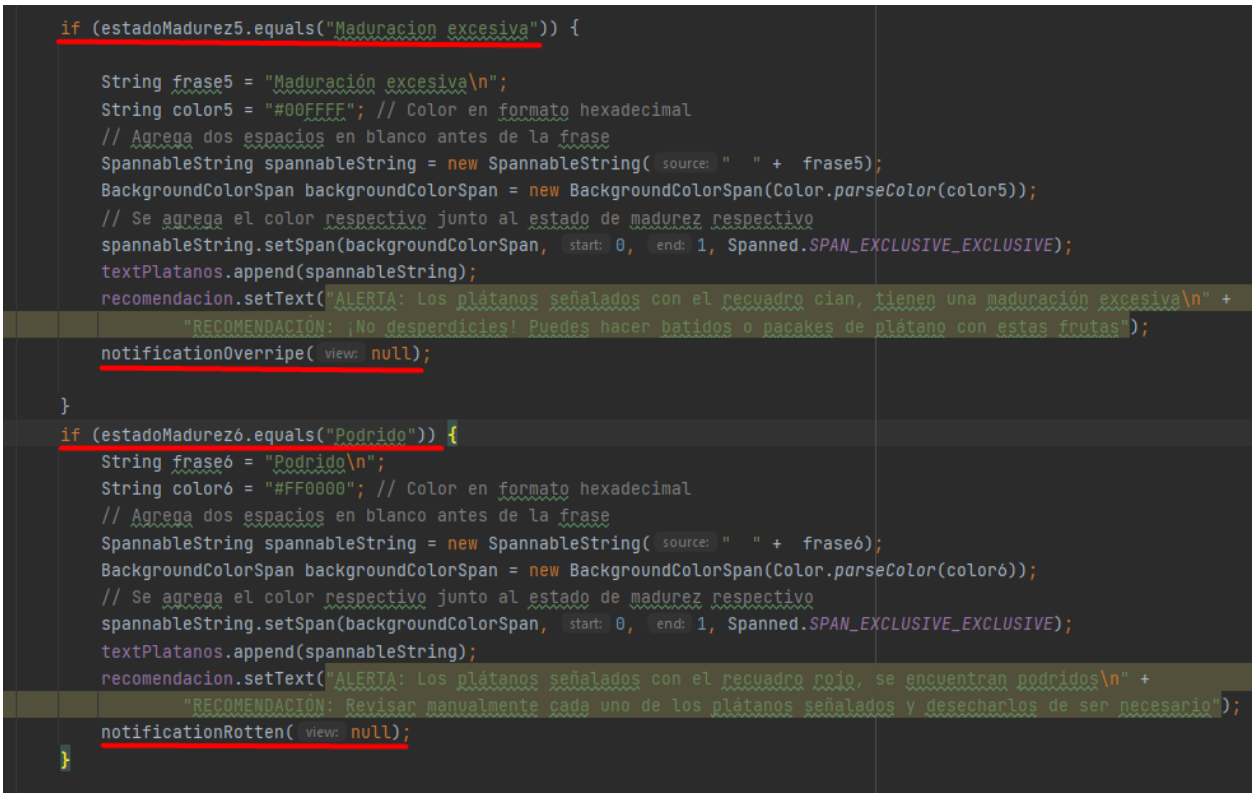
Nota. Autoría propia

Para la siguiente parte de este método, visualizado en la Figura 82, se agrega otros condicionales, los cuales indican ciertos mensajes e indicadores de colores para cada uno de los mismos. De manera específica para la variable “estadoMadurez5” toma un valor de “Maduración excesiva”, se envía el valor del texto al cuadro de texto correspondiente. Además, para esta condición también se manda una recomendación, al cuadro de texto “recomendación” y se establece la ejecución de la alerta de notificación por medio del método “notificationOverripe”. Este proceso se repite para el siguiente condicional de la variable “estadoMadurez6” cuando toma un valor de “Podrido”, este proceso es similar, sin embargo, los mensajes enviados al cuadro de

visualización texto cambian y el tipo de notificación que se activa es mediante el método “notificationRotten”.

Figura 82

Condicionales para cada uno de los estados de madurez de las frutas



```
if (estadoMadurez5.equals("Maduracion excesiva")) {

    String frase5 = "Maduración excesiva\n";
    String color5 = "#00FFFF"; // Color en formato hexadecimal
    // Agrega dos espacios en blanco antes de la frase
    SpannableString spannableString = new SpannableString( source: "  " + frase5);
    BackgroundColorSpan backgroundColorSpan = new BackgroundColorSpan(Color.parseColor(color5));
    // Se agrega el color respectivo junto al estado de madurez respectivo
    spannableString.setSpan(backgroundColorSpan, start: 0, end: 1, Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
    textPlatanos.append(spannableString);
    recomendacion.setText("ALERTA: Los plátanos señalados con el recuadro cian, tienen una maduración excesiva\n" +
            "RECOMENDACIÓN: ¡No desperdicies! Puedes hacer batidos o pacakes de plátano con estas frutas");
    notificationOverripe( view: null);

}
if (estadoMadurez6.equals("Podrido")) {
    String fraseó = "Podrido\n";
    String coloró = "#FF0000"; // Color en formato hexadecimal
    // Agrega dos espacios en blanco antes de la frase
    SpannableString spannableString = new SpannableString( source: "  " + fraseó);
    BackgroundColorSpan backgroundColorSpan = new BackgroundColorSpan(Color.parseColor(coloró));
    // Se agrega el color respectivo junto al estado de madurez respectivo
    spannableString.setSpan(backgroundColorSpan, start: 0, end: 1, Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
    textPlatanos.append(spannableString);
    recomendacion.setText("ALERTA: Los plátanos señalados con el recuadro rojo, se encuentran podridos\n" +
            "RECOMENDACIÓN: Revisar manualmente cada uno de los plátanos señalados y desecharlos de ser necesario");
    notificationRotten( view: null);
}
```

Nota. Autoría propia

Para finalizar, se realiza la redimensión de la imagen a 1000x1000 pixeles, esto con el fin poder apreciar la imagen. También, como indicativo de la finalización de todo el proceso se termina con la barra de progreso.

3.4. Diseño

3.4.1. Descripción general del funcionamiento del sistema embebido

El sistema embebido de monitoreo consiste en el desarrollo de un prototipo de electrónico el cual está integrado por una placa SBC y una cámara de alta resolución, la cual permite capturar las imágenes de las frutas, para posteriormente procesar estas imágenes mediante el uso de un modelo de Machine Learning y establecer el grado de madurez ya sea de plátanos o de fresas.

De forma inicial, los datos recolectados para el modelo de Machine Learning se realiza mediante de imágenes encontrados en datasets, lo cual permitirá tener una amplia variedad de muestras de las frutas (plátanos y fresas) a monitorear. El prototipo estará conformado por la placa Raspberry Pi, la cual, a su vez, tendrá una conexión con la cámara de alta resolución; estos dos componentes serán montados sobre una estructura en forma de recipiente, en donde la cámara se sitúa en la parte superior del mismo recipiente para tener una visión de todas las frutas que se desea monitorear. Este proceso de monitoreo se realiza mediante la captura de una imagen de las frutas (plátanos y fresas) utilizando el Raspberry Pi en conjunto con la cámara, que posteriormente va a ser procesado por el modelo de Deep Learning, determinando de esta manera el estado de madurez de las frutas.

Como parte final, el resultado obtenido mediante el prototipo electrónico se envía a una aplicación móvil, en la cual, se presenta el estado de madurez, ya sea de los plátanos o de las fresas, además, anexando la imagen respectiva del proceso que se ha llevado a cabo, en caso de que el estado de las frutas (fresas y plátanos), tenga un estado de maduración excesiva o podrido, se emitirá una notificación de alerta.

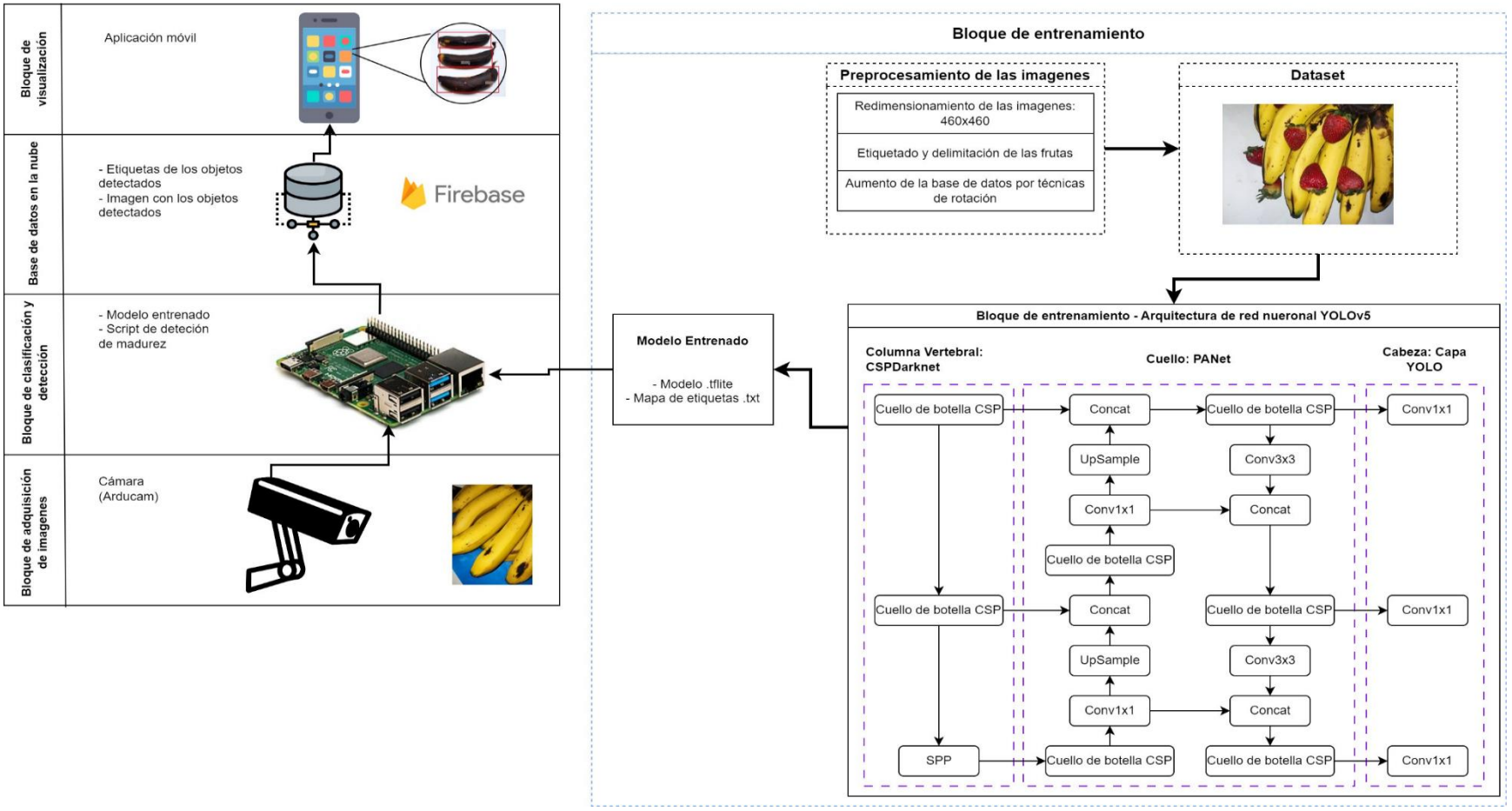
3.4.2. Arquitectura del sistema embebido

En la Figura 83 se indica la arquitectura del sistema, en donde está conformado por algunos bloques descritos a continuación:

- Bloque de entrenamiento: En el bloque de entrenamiento se define subbloques como Dataset, en el cual se encuentran las imágenes con las cuales se va a entrenar el modelo, otro subbloque de Preprocesamiento de imágenes el cual actúa sobre el bloque del Dataset, realizando acciones como la redimensión de las imágenes y el etiquetado.
- Bloque de adquisición de imágenes: Está compuesto principalmente por un sensor compatible con Raspberry Pi 4, específicamente se ha utilizado el sensor ArduCam.
- Bloque de clasificación y detección: Este bloque está compuesto por la placa SBC Raspberry Pi 4, además contiene los modelos entrenados, así también el script de captura de imágenes y de detección del estado de madurez de las frutas (fresas y plátanos).
- Bloque (Base de datos en la nube): Está constituido por una base de datos alojado en la nube, específicamente para este sistema embebido se ha utilizado Firebase. La base de datos almacena las etiquetas de los objetos detectados y además la imagen procesada con las detecciones.
- Bloque de visualización: Este último bloque se centra en una aplicación móvil para dispositivos Android, en la cual se puede visualizar el resultado de la detección de las frutas.

Figura 83

Arquitectura del sistema embebido



Nota. Autoría propia

Capitulo IV

EJECUCIÓN Y PRUEBAS DE FUNCIONAMIENTO

En este capítulo se evalúa y se comprueba la implementación del sistema embebido que se ha propuesto para la detección de madurez en las frutas (plátanos y fresas). Dentro de este apartado se realizan pruebas de funcionamiento del sistema, en donde se analiza la capacidad de detección frente a las 10 clases que se ha propuesto para determinar el grado de madurez de las frutas. Cabe recalcar que, para realizar estas pruebas, se ha montado el sistema sobre una estructura realizada a partir de un recipiente y una estructura impresa en tres dimensiones.

4.1. Pruebas funcionales

- Pruebas con la misma clase: Estas pruebas hacen referencia a la detección y clasificación de las frutas (fresas o plátanos) teniendo en cuenta un mismo nivel de madurez para las mismas. Este proceso es llevado a cabo mediante el modelo entrenado.
- Pruebas con diferentes clases: Las pruebas con diferentes clases buscan comprobar el desempeño del sistema embebido con respecto a la detección de 2 o más clases, ya sea de plátanos o de fresas.

4.1.1. Pruebas con la misma clase

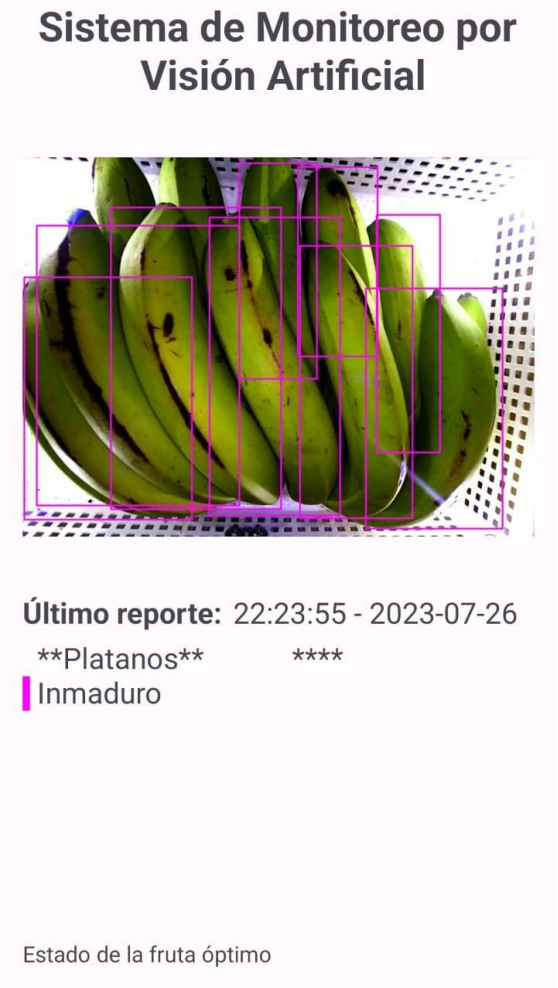
4.1.1.1. Inmaduro – Plátano.

En la Figura 84 se ha realizado una prueba con plátanos en un estado de madurez denominado “inmaduro”, como se indica en la misma figura se puede notar, que el sistema embebido detecta y clasifica de forma adecuada cada una de las frutas presentadas para este caso, además en la parte inferior se indica un mensaje de estado óptimo de las frutas, esto debido a que,

el grado de madurez “inmaduro” para un fruta no representa un riesgo de desperdicio o ingesta en algunos casos.

Figura 84

Resultados de la detección de los plátanos con estado "inmaduro"



Nota. Autoría propia

4.1.1.2. Fresco inmaduro – Plátano.

Para la prueba con los plátanos que tienen un estado de fresco inmaduro, se puede visualizar en la Figura 85, que el modelo de entrenado, logra delimitar por medio de cajas los 4 plátanos visibles en la imagen tomada. Las cajas delimitadores se encuentran dibujadas con un color verde

oscuro, además se tiene un indicativo de acuerdo a este color en la parte infererior, y finalmente se tiene un mensaje en el cual se indica que los plátanos monitoreados se encuentran en un estado óptimo.

Figura 85

Resultados de la detección de los plátanos con estado "fresco inmaduro"

Nota. Autoría propia

4.1.1.3. Fresco – Plátano.

Para el resultado del estado de madurez “fresco”, viualizado en la Figura 86, se muestra que las cajas delimitadoras de color púpura oscura, encierran a cada una de la frutas, y como este

estado de madurez aún es óptimo, se indica un mensaje especificando que la fruta se encuentra en buen estado. Como dato adicional también dentro de este reporte se indica la fecha y la hora en la cual se ha realizado el monitoreo de las frutas.

Figura 86

Resultados de la detección de los plátanos con estado "fresco"

Nota. Autoría propia

4.1.1.4. Maduro – Plátano.

En esta prueba se ha capturado de manera correcta la imagen de la fruta que se está monitoreando, en este caso, los plátanos se encuentran en un estado de madurez denominado “maduro”, el sistema detecta de forma adecuada cada uno de los plátanos, los cuales son encerrados dentro de recuadros color naranja oscuro. Ya que el estado de la fruta no respresenta

ningún riesgo de desperdicio ni de consumo, se muestra un mensaje especificando que los plátanos se encuentran consumibles, observar Figura 87.

Figura 87

Resultados de la detección de los plátanos con estado "maduro"

Nota. Autoría propia

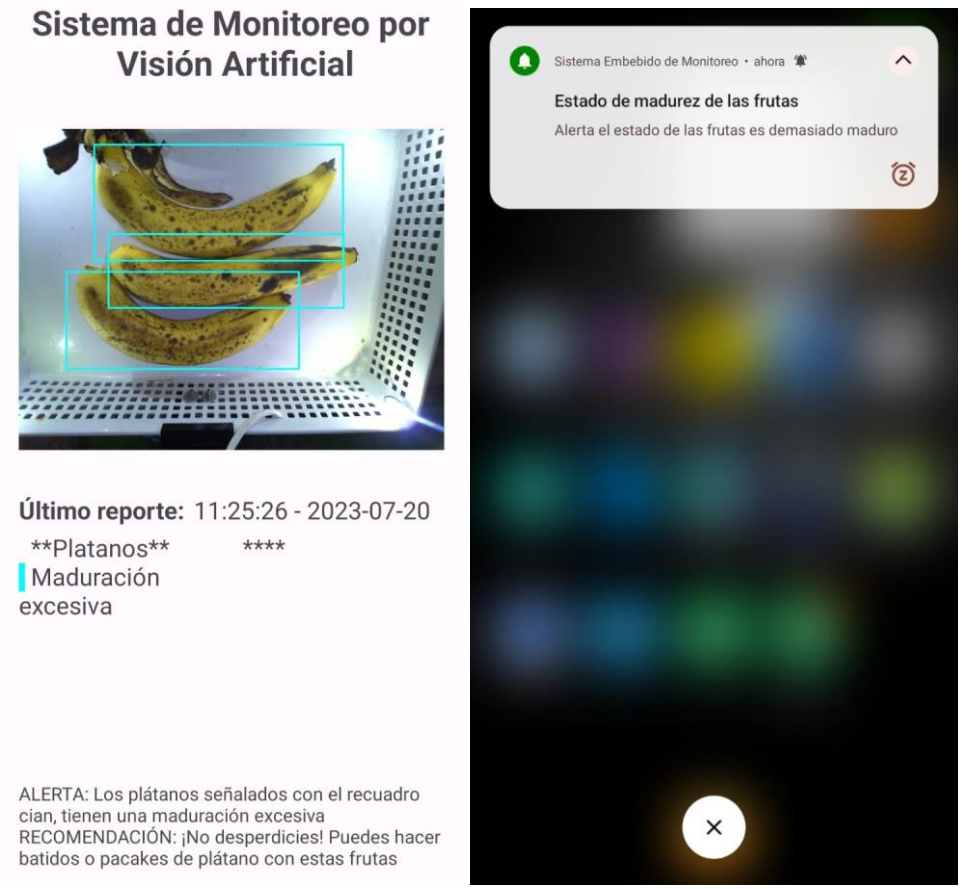
4.1.1.5. Maduración excesiva – Plátano.

Para la prueba funcional con plátanos que tienen un estado de madurez excesivo mostrado en la Figura 88, se utiliza un recuadros color cian para encerrar a cada una de las frutas en este estado, esto proceso es realizado de forma automática por el sistema; una fruta al encontrarse en este grado de madurez, se emite un mensaje de alerta en donde se recalca las frutas con esta

excesiva madurez, y también una recomendación para evitar su desperdicio. En caso de que el usuario no tenga en primer plano la ejecución de la aplicación, se recibirá una notificación en donde se indicará acerca del estado de las frutas que se esten monitoreando.

Figura 88

Resultados de la detección de los plátanos con estado "maduración excesiva"



Nota. Autoría propia

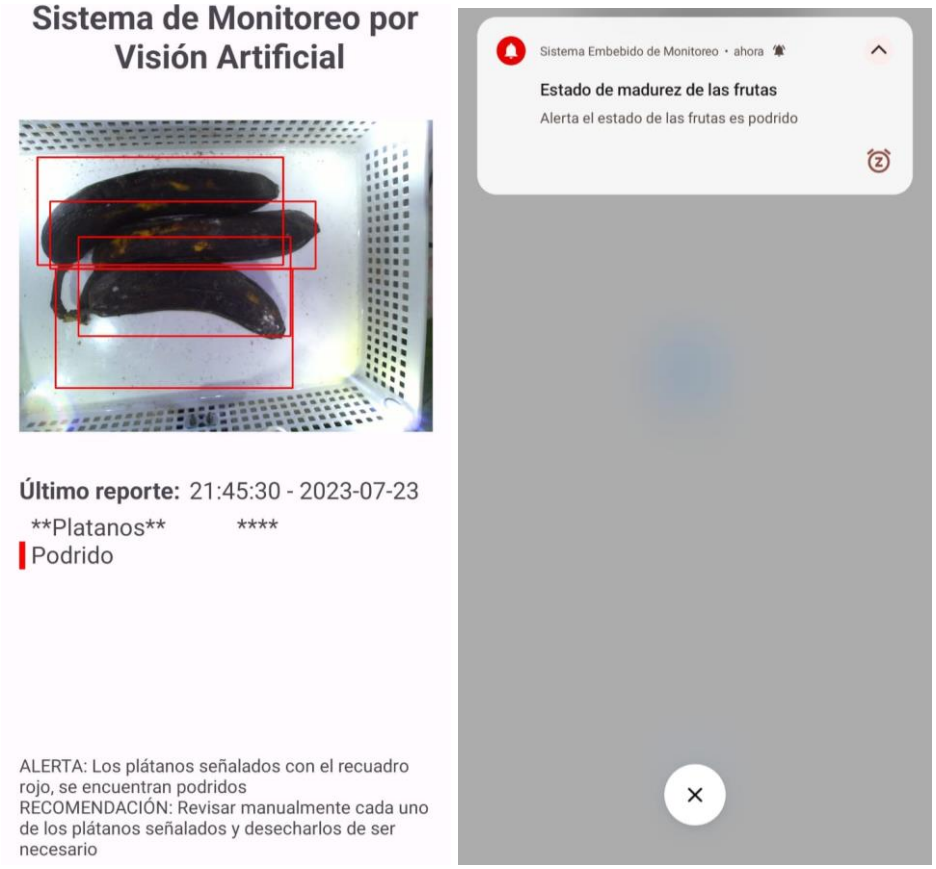
4.1.1.6. Podrido – Plátano.

Como prueba final de los plátanos se establece el último estado de la fruta, el cual es el estado de descomposición, o también denominado “podrido”, en la Figura 89 se ha realizado en

donde se ha monitoreado plátanos podridos, para este caso las cajas delimitadoras son de color rojo intenso, en la parte inferior se muestra la alerta correspondiente a este caso y también una recomendación para verificar el estado de los plátanos. Los plátanos en este estado no son comestibles lo cual representan un riesgo de salud si se llega a ingerir. Al igual que la prueba anterior, el estado “podrido” emite una notificación de alerta para indicar al usuario acerca del estado de la fruta.

Figura 89

Resultados de la detección de los plátanos con estado "podrido"



Nota. Autoría propia

4.1.1.7. Inmaduro – Fresa.

Para la prueba con fresas inmaduras que es indica en la Figura 90, se muestra que los recuadros que se han utlilizado para delimitar las frutas son de color magenta oscuro. Este estado de madurez, al no representar ningún riesgo de desperdicio simplemente se emite un mensaje dentro de la misma aplicación indicando que las frutas se encuentran en estado óptimo.

Figura 90

Resultados de la detección de las fresas con estado "inmaduro"

Nota. Autoría propia

Las pruebas en fresas maduras, es similar al caso anterior, el cambio principal es el color de la caja delimitadora, los recuadros en color azul marino representan las fresas maduras, siendo este uno de los grados de madurez óptimo de la fruta no se emite ningún tipo de alerta, simplemente mostrando con un mensaje en la parte inferior que las frutas tienen un estado óptimo, visualizar Figura 91.

4.1.1.8. Maduro – Fresa.

Figura 91

Resultados de la detección de las fresas con estado "maduro"

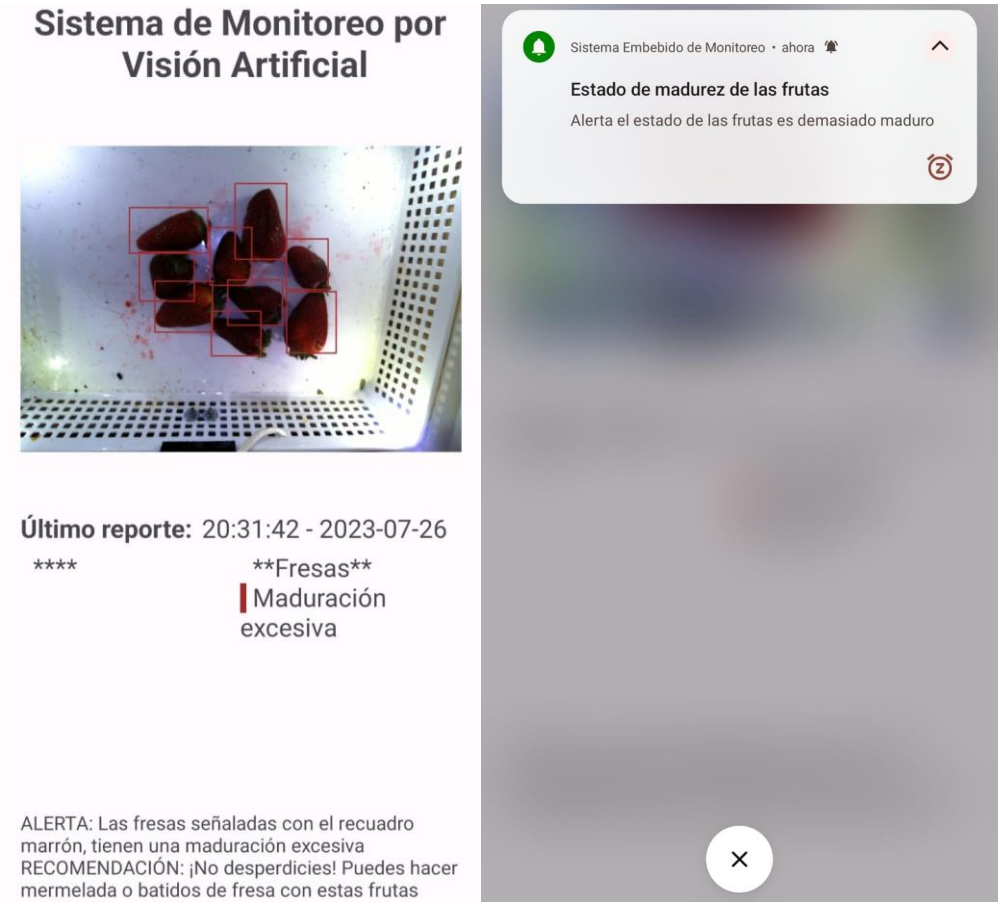
Nota. Autoría propia

4.1.1.9. Maduración excesiva – Fresa.

Cuando las fresas tienen un nivel de madurez excesiva, los cajas delimitadoras tienen un color marrón, además se ajusta cierto tipo de mensaje diferente, en cual como recomendación se indica que se puede realizar mermelada o batido con las frutas en este estado. Y con el fin de evitar el desperdicio de las fresas, mientras las aplicación se encuentre ejecutandose en segundo plano, se notificará del estado de madurez avanzado para las fresas, como se visualiza en la Figura 92.

Figura 92

Resultados de la detección de las fresas con estado "Maduración excesiva"



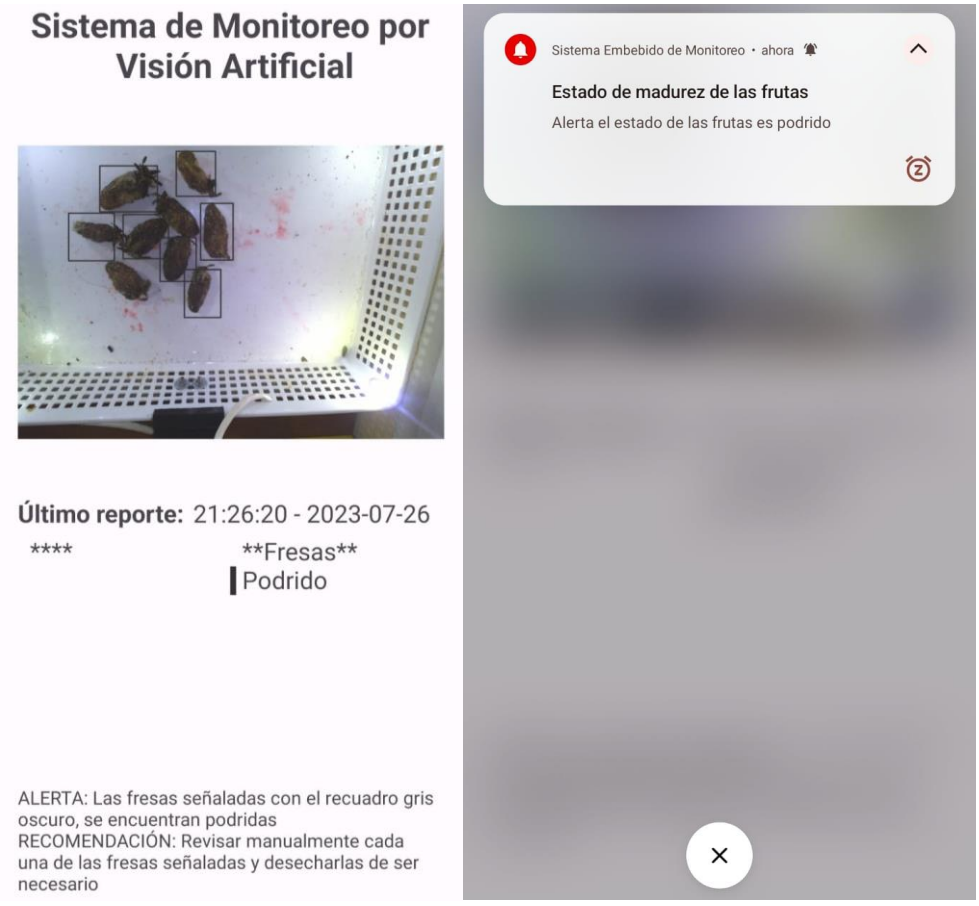
Nota. Autoría propia

4.1.1.10. Podrido – Fresa.

Para el último estado que alcanza las fresas, es denonimado “podrido”, los recuadros delimitadores se pintan de color gris oscuro como se puede observar en la Figura 93, además se efectúa la emisión de una alerta y una recomendación dentro de la aplicación; la recomendación sugiere al usuario revisar de forma manual las fresas monitoreadas, y de ser el caso desecharlas. La notificación de alerta cuando la aplicación se encuentre en segundo plano, también señala el estado de las fresas.

Figura 93

Resultados de la detección de las fresas con estado "podrido"



Nota. Autoría propia

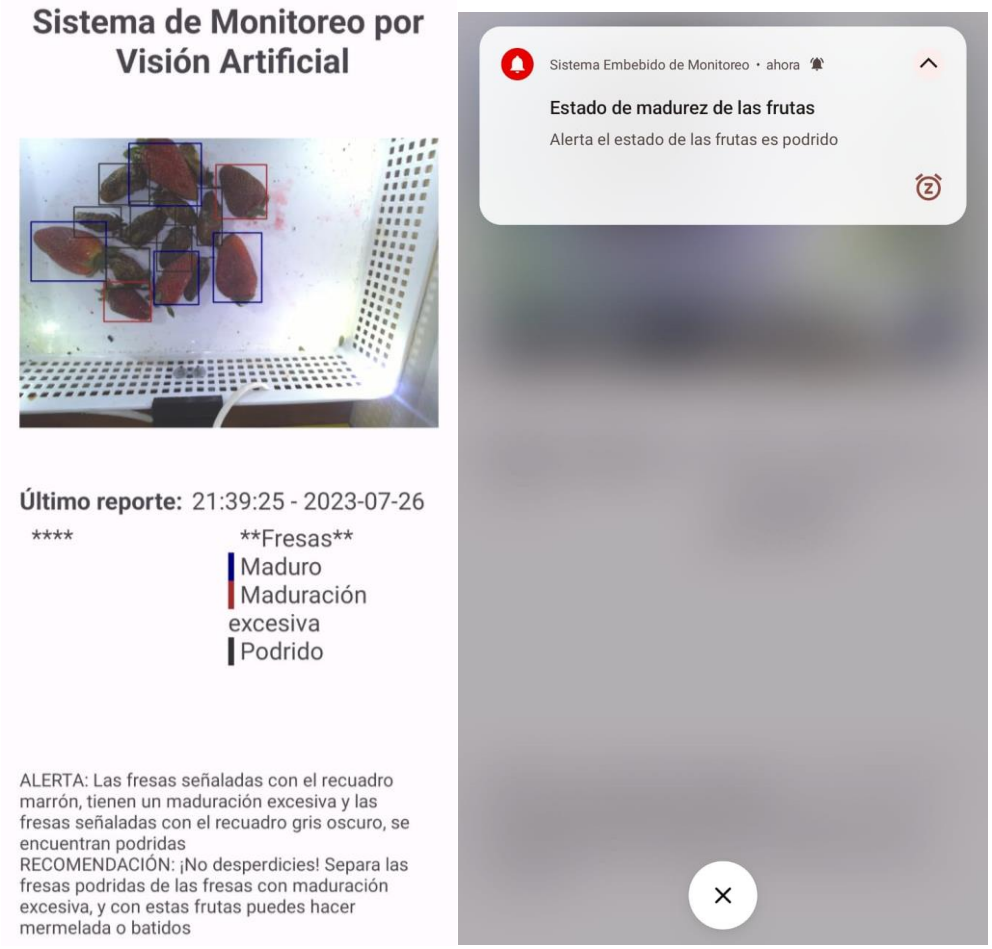
4.1.2. Pruebas con diferentes clases

4.1.2.1. Prueba 1 (Fresas).

Para esta primera prueba se ha utilizado únicamente fresas con diferentes estados de madurez, por lo cual, el sistema debe ser capaz de identificar diferentes clases dentro de la misma fruta. Específicamente en la Figura 94 se muestra los cuadros delimitadores colocados sobre la imagen capturada, en donde cada uno de estos recuadros, de acuerdo a su color indican el estado madurez que poseen estas fresas (maduro, maduración excesiva y podrido). Dentro del primer plano de ejecución de la aplicación, el apartado de la alerta reafirma las fresas que se encuentran podridas y las fresas con demasiada madurez, para la recomendación se sugiere la separación de las frutas podridas de las demás fresas. En este caso, al igual que en apartados anteriores se tiene una notificación de alerta, esto con la finalidad de que se pueda evitar la descomposición y desperdicio de las otras fresas. Cabe recalcar que las notificaciones toman grado de madurez.

Figura 94

Resultados prueba 1



Nota. Autoría propia

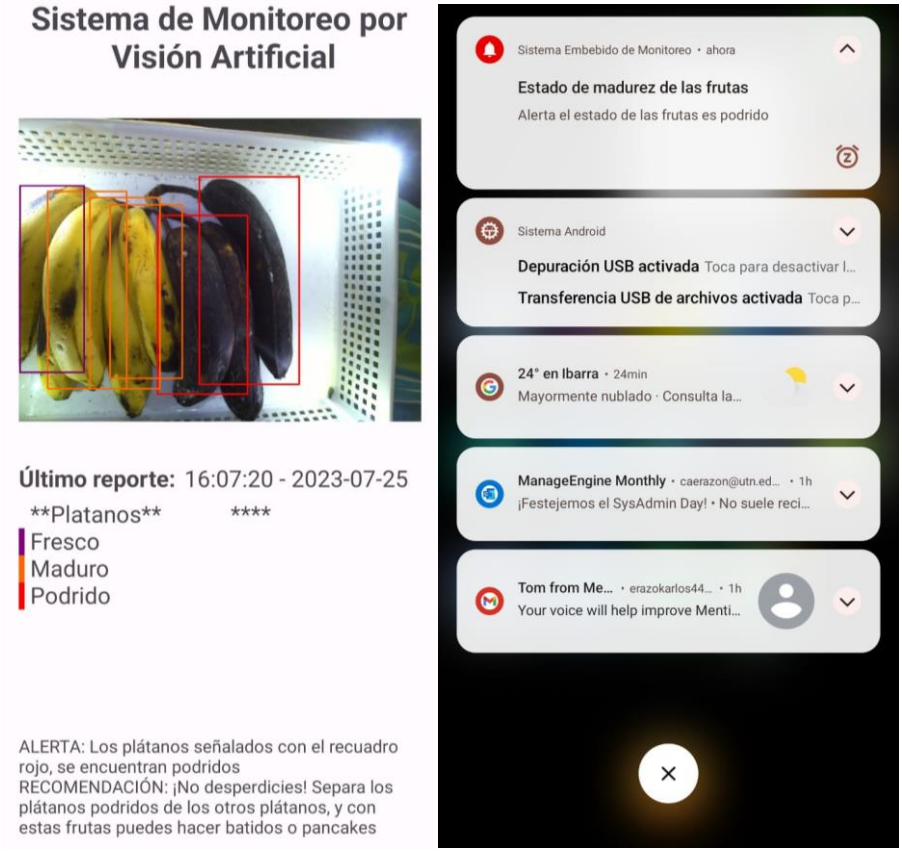
4.1.2.2. Prueba 2 (Plátanos).

Para los plátanos se ha realizado el mismo procedimiento que en la prueba anterior, en donde, dentro del sistema se colocan plátanos con diferentes estados de madurez. En la Figura 95 se puede visualizar los resultados de esta prueba, en donde se aprecia claramente la detección y clasificación de los plátanos en estado maduro, fresco y podrido, cada uno estas frutas delimitadas

mediante las cajas delimitadores con el respectivo indicador de color para cada uno. En la parte inferior se observa el mensaje de alerta y recomendación adecuada a la situación de la prueba, y en la parte derecha de la misma figura se muestra la notificación que se recibe cuando los plátanos tienen un estado de podrido. En el estado de los plátanos podridos es necesario revisar forma manual las frutas para poder desecharlas de ser el caso, tal y cual se recomendia en el apartado de la zona inferior del reporte.

Figura 95

Resultados prueba 2



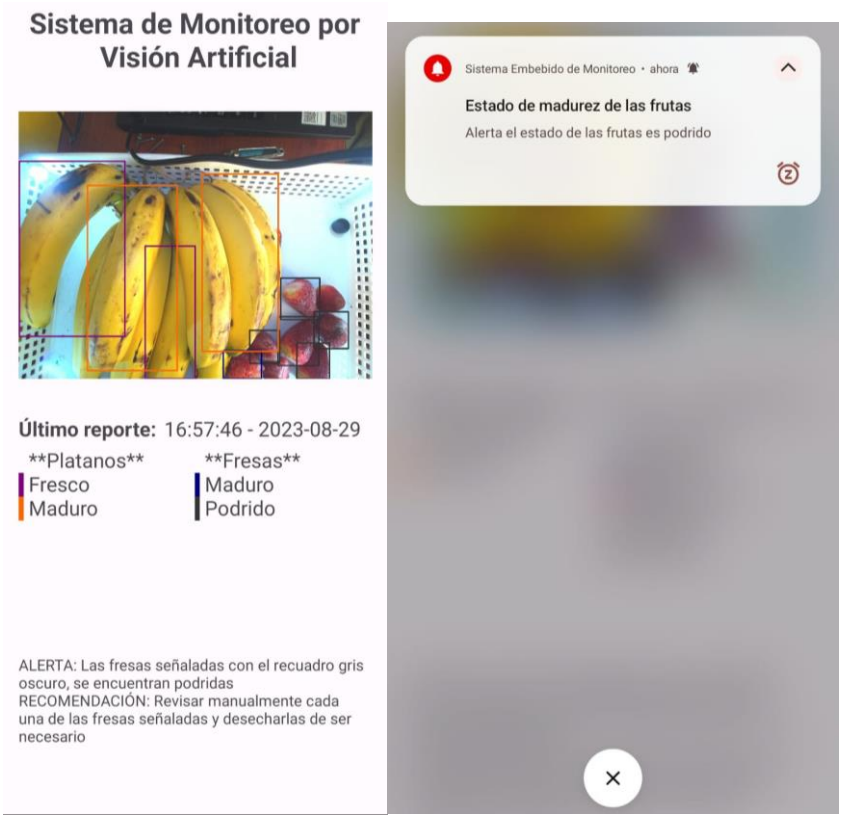
Nota. Autoría propia

4.1.2.3. Prueba 3 (Fresas y plátanos).

En la Figura 96 se visualiza la realización de esta prueba con fresas y plátanos; para las fresas se detecta 3 estados posibles: maduro, maduración excesiva y podrido, y de parte de los plátanos se detecta unicamente los plátanos como maduros. Las fresas podridas al permanacer o estar cerca del espacio de otras frutas, existe un riesgo de desperdicio, debido a que la propagación de los hongos y otros microorganismos nacen a partir de este estado de descomposición. A la derecha de la misma figura se indica la alerta que se recibe tras detectar que una o más de las fresas se encuentran podridas.

Figura 96

Resultados prueba 3



Nota. Autoría propia

4.1.3. Umbral de confianza

En este apartado se evalúa precisión del sistema de monitoreo por visión artificial, estas pruebas han sido tabuladas en la Tabla 17.

Tabla 17

Umbral de confianza

Umbral de confianza				
Fruta	Nivel de madurez	Valores detectados	Valores reales	Umbral de confianza (%)
Plátano	Inmaduro	5	7	71.42
	Fresco inmaduro	7	7	100
	Fresco	6	7	85.71
	Maduro	5	7	71.42
	Maduración excesiva	6	7	85.71
	Podrido	5	7	71.42
Fresa	Inmaduro	9	11	81.81
	Maduro	9	11	81.81
	Maduración excesiva	8	11	72.72
	Podrido	8	11	72.72
		Total		79.47

Nota. La tabla se indica los valores de detección y los valores reales para determinar el umbral de confianza que tiene el sistema. Fuente: Autoría propia

4.2. Análisis Costo/Beneficio

En esta sección se analiza los costos que se ha realizado para la construcción del sistema embebido de monitoreo. Dentro de este análisis se ha tomado en cuenta costos de hardware, software, infraestructura e ingeniería, además, enfocando también en los beneficios que conlleva el funcionamiento del sistema.

4.2.1. Costos de hardware

En la Tabla 18 se muestran los componentes de hardware que se ha usado para la creación de este sistema, además, indicando el costo unitario y el costo total de este apartado.

Tabla 18

Costos de hardware

Costos de hardware			
Cantidad	Hardware	Costo unitario	Costo total
1	Arducam (Sensor)	$ 80.85	$80.85
1	Raspberry Pi 4	$ 250	$ 250
	Total		$ 330.85

Nota. La tabla indica los costos unitarios y totales que se ha realizado para hardware. Fuente: Autoría propia

4.2.2. Costos de software

Para la especificar los costos de software se ha construido la Tabla 19 en donde se indica el uso de software de libre uso en su mayoría, sin embargo, para el entramiento del modelo se ha utilizado una suscripción pagada de Google Colab, esto con la finalidad de tener un mayor acceso a recursos para durante el entrenamiento.

Tabla 19

Costos de software

Costos de software		
Software	Costo unitario	Costo total
Google Colab	$ 10	$ 10
Python	$ 0	$ 0
Android Studio	$ 0	$ 0
Google Drive	$ 0	$ 0
Total		$ 10

Nota. La tabla indica los costos unitarios y totales que se ha realizado para el software. Fuente: Autoría propia

4.2.3. Costos de infraestructura

Los costos de infraestructura hacen referencia a los materiales extra que fueron utilizados para la construcción del sistema, en este caso la estructura principal fue creada a partir de un modelo en 3D, y para la iluminación se usó cable UTP y Leds como se muestra en la Tabla 20.

Tabla 20

Costos de infraestructura

Costos de infraestructura			
Cantidad	Elemento	Costo unitario	Costo total
1	Impresión 3D	$ 100	$ 100
6	Leds	$ 1.25	$ 7.50
2 metros	Cable UTP	$ 0.80	$ 1.60

Total	$ 109.1

Nota. La tabla indica los costos unitarios y totales que se ha realizado para la infraestructura. Fuente: Autoría propia

4.2.4. Costos de ingeniería

En la Tabla 21 se define las actividades que se han realizado para la construcción del sistema como: estudio del área y documentación, diseño de hardware y de programación del sistema, entrenamiento del modelo de detección y la construcción de la infraestructura del sistema. Todos los gatos de esta misma tabla son cubiertos por el autor del presente proyecto.

Tabla 21

Costos de ingeniería

Costos de ingeniería		
Actividad	Costo unitario	Costo total
Estudio de área y documentación	$ 30	$ 30
Entrenamiento del modelo de detección	$20	$ 20
Diseño de hardware y programación del sistema	$ 150	$ 150
Construcción de la infraestructura sistema	$ 100	$ 100
Total		$ 300

Nota. La tabla indica los costos unitarios y totales que se ha realizado en ingeniería. Fuente: Autoría propia

4.2.5. Costo general del sistema

Para recopilar todos los gastos realizados durante este proyecto, se ha organizado la Tabla 22, en donde se especifica los costos totales de hardware, software, infraestructura e ingeniería. El valor neto del proyecto fue de 449.95 dólares, en donde se especifica únicamente el costo de los materiales y otros implementos para la construcción del sistema; el valor de 300 dólares hace referencia al número de horas dedicadas en construcción de infraestructura, programación, investigación y diseño del sistema, todos los gastos han sido asumidos por el autor de este proyecto, dando un total de 749.95 dólares para todos los costos del sistema.

Tabla 22

Costo general del sistema

Costo general del sistema	
Descripción	Costos Unitario
Costos de hardware	$ 330.85
Costos de software	$ 10
Costos de infraestructura	$ 109.1
Costos de ingeniería	$ 300
Total	$ 749.95

Nota. La tabla indica los costos totales que se ha realizado para la construcción de este sistema. Fuente: Autoría propia

4.2.6. Beneficios

Los beneficios que conlleva la implementación de este sistema embebido de monitoreo en hogares se describen a continuación:

- Reducción del desperdicio de alimentos: Al medir el grado de madurez de las frutas (plátanos y fresas), el sistema permite a los usuarios consumir las frutas en el momento óptimo, evitando que se desperdicien al consumirlas antes de tiempo o después de que hayan sobrepasado su punto óptimo de maduración.
- Prevención de enfermedades relacionadas con alimentos en mal estado: Al evitar el consumo de frutas que se encuentran en un estado avanzado de descomposición, se disminuye el riesgo de enfermedades transmitidas por alimentos contaminados.
- Ahorro económico: Al reducir el desperdicio de frutas, los usuarios pueden ahorrar dinero aprovechando a su vez el máximo tiempo de vida útil de las frutas.
- Contribución medioambiental: La reducción del desperdicio de alimentos también tiene un impacto positivo en el medio ambiente, ya que disminuye la cantidad de alimentos que terminan en basureros.

CONCLUSIONES

Por medio de la investigación y fundamentación teórica, se ha podido recolectar información suficiente acerca de visión artificial, redes neuronales, Deep Learning, sobre tipos de frutas, permitiendo de esta manera establecer requerimientos que se permitan cumplir con el objetivo principal del sistema de monitoreo. Además, por medio de este mismo proceso de investigación se ha tomado en consideración otros aspectos más puntuales como: hiperparámetros e indicadores de maduración, los cuales influirán directamente en el desempeño del sistema.

El diseño basado en una arquitectura IoT que permite la obtención de datos acerca de las frutas y su procesamiento mediante Deep Learning, presenta una solución para el monitoreo del grado de madurez que puedan tener las fresas y los plátanos. La combinación de tecnologías IoT junto con Deep Learning posibilita la recopilación, análisis y procesamiento automatizado de los datos, ofreciendo de esta manera información precisa acerca del estado de madurez de las frutas monitoreadas.

La construcción del prototipo tomando en cuenta los requerimientos levantados durante la etapa de investigación y fundamentación teórica, conlleva a la implementación de hardware y software dentro de un mismo sistema, de tal manera que tengan un funcionamiento integrado, solventando de esta forma la medición del estado de madurez de las frutas, y además previniendo el desperdicio de las mismas y evitando algún tipo de enfermedad relacionada con el consumo de alimentos en mal estado.

La arquitectura YOLOv5 es superior a MobileNet-v2 para el entrenamiento de modelos que tienen como tarea la detección y clasificación de imágenes, esto debido a que YOLOv5 utiliza un conjunto capas especializadas y más eficientes para la extracción de características de las imágenes.

En base a al umbral de confianza que se ha determinado para el sistema, se puede determinar que este proyecto es capaz de identificar plátanos y fresas en sus diferentes estados de madurez, teniendo un porcentaje de 79% de precisión, considerando así, que tiene un rendimiento aceptable para evitar el desperdicio de la mayor parte de la fruta monitoreada

RECOMENDACIONES

El apartado de investigación teórica del proyecto debe estar enfocado principalmente a tipos redes neuronales, tipos de arquitecturas e hiperparámetros, debido a que, se requiere de una arquitectura neuronal suficientemente óptima para la extracción de características en imágenes, además considerando el uso adecuado de recursos computacionales para que el proceso de entrenamiento del modelo no sea muy complejo.

Se debe tomar en cuenta también el proceso de maduración de las frutas que se va a monitorear, puesto que existen frutas climatéricas y no climatéricas, las cuales van a tener un comportamiento distinto durante este proceso de maduración; este factor puede ser considerado al momento de la programación del sistema para determinar las veces que se va a monitorear por día o cada cierto tiempo.

Los datasets adquiridos deben ser revisados de forma manual, debido a que pueden carecer de muestras de una clase en específico. Por tal motivo, si las imágenes para el dataset bajo cierto criterio se consideran insuficientes, es recomendable extraer más imágenes de otros datasets o construir un conjunto de muestras a partir fotografías de plátanos y fresas durante todo el proceso de maduración; a partir de esto se puede aumentar el nivel de precisión del modelo que se desea entrenar.

Durante la construcción del sistema de monitoreo, se debe procurar que el hardware a utilizar tenga una disposición adecuada sobre el sistema, además, tomando en cuenta otros factores para la correcta captura de imágenes como: iluminación y color de fondo.

REFERENCIAS

Abobakr, A., Mohsen, M., Said, L. A., Madian, A. H., Elwakil, A. S., & Radwan, A. G. (2019). Banana ripening and corresponding variations in bio-impedance and glucose levels. *NILES 2019 - Novel Intelligent and Leading Emerging Sciences Conference*, 130–133. https://doi.org/10.1109/NILES.2019.8909322

ArduCam. (2021). *Arducam MINI High Quality Camera with M12 mount lens, 12.3MP 1/2.3" 477P HQ Camera Module for NVIDIA® Jetson Nano/Xavier NX/AGX Orin/Orin Nano/Orin NX - Arducam*. https://www.arducam.com/product/arducam-high-quality-camera-for-jetson-nano-and-xavier-nx-12mp-m12-mount/

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. https://github.com/AlexeyAB/darknet.

Bonilla, J. & Prieto F. (2016). Determinación del estado de maduración de frutos de feijoa mediante un sistema de visión por computador utilizando información de color. Revista de Investigación Desarrollo e Innovación, 7(1), 111-126. doi: 10.19053/20278306.v7.n1.2016.5603

Boufeloussen, O. (2020). *Simple Explanation of Recurrent Neural Network (RNN)*. https://medium.com/swlh/simple-explanation-of-recurrent-neural-network-rnn-1285749cc363

Brezmes, J. (2001). Diseño de una nariz electrónica para la determinación no destructiva del grado de maduración de la fruta. [Tesis doctoral, Universitat Politécnica de Catalunya]. UPCommons. https://upcommons.upc.edu/bitstream/handle/2117/94188/CAPITOL1.pdf

Capino, A., & Farcuh, M. (2022). *El etileno y la regulación de la maduración de los frutos | Extensión de la Universidad de Maryland*. https://extension.umd.edu/resource/ethylene-and-regulation-fruit-ripening

CDC. (16 de Julio de 2021). *Centers for Disease Control and Prevention*. Obtenido de Centers for Disease Control and Prevention: https://www.cdc.gov/foodsafety/es/foodborne-germs-es.html

Chica, A., Serra, J., & Botti, V. (2001). *Aplicación de una red neuronal para la predicción de la reacción catalítica isomerización del n-Octano*. https://www.researchgate.net/publication/228815505_Aplicacion_de_una_red_neuronal_para_la_prediccion_de_la_reaccion_catalitica_isomerizacion_del_n-Octano/download

FAO. (2019). El estado mundial de la agricultura y la alimentación. Progresos en la lucha contra la pérdida y el desperdicio de alimentos. Roma.

FAO. (2020). *Organización de las Naciones Unidas para la Alimentación y la Agricultura*. Obtenido de Organización de las Naciones Unidas para la Alimentación y la Agricultura: https://www.fao.org/news/story/es/item/1310444/icode/

Gao, J., Zhang, Y., Li, Z., & Liu, M. (2020). Role of ethylene response factors (ERFs) in fruit ripening. *Food Quality and Safety*, *4*(1), 15–20. https://doi.org/10.1093/FQSAFE/FYZ042

Gershenson, C. (2003). *Artificial Neural Networks for Beginners*. https://www.researchgate.net/publication/1956697_Artificial_Neural_Networks_for_Beginners

González, F., & Martínez, J. A. (2001). Evaluación estadística del comportamiento de líneas aéreas de distribución frente a sobretensiones de origen externo. *TDX (Tesis Doctorals en Xarxa)*. https://www.tdx.cat/handle/10803/6281

Hernández, L. (2019). *Organización de las Naciones Unidas para la Alimentación y la Agricultura*. Obtenido de Organización de las Naciones Unidas para la Alimentación y la Agricultura: https://www.fao.org/3/x4390T/x4390t06.html

IBM. (2020). *What is Deep Learning? | IBM*. https://www.ibm.com/cloud/learn/deep-learning#toc-how-deep-l-vLJwLmX4

IBM Cloud Education. (2020). *What are Neural Networks? | IBM*. https://www.ibm.com/cloud/learn/neural-networks

Ihab, M. (2017). *(PDF) Detection and Tracking of Pallets using a Laser Rangefinder and Machine Learning Techniques*. https://www.researchgate.net/publication/324165524_Detection_and_Tracking_of_Pallets_using_a_Laser_Rangefinder_and_Machine_Learning_Techniques

Klodd, A., Tepe, E., & Hoover, E. (2021). *Harvesting strawberries | UMN Extension*. https://extension.umn.edu/strawberry-farming/harvesting-strawberries

Krishnan Sandhya. (2021). *How to determine the number of layers and neurons in the hidden layer? | by Sandhya Krishnan | Geek Culture | Medium | Geek Culture*. https://medium.com/geekculture/introduction-to-neural-network-2f8b8221fbd3

Lawaniya, H. (2020). Computer Vision. *IET Computer Vision*.

Lema, E. (2021). SISTEMA ELECTRÓNICO DE DETECCIÓN DE RESIDUOS DE PLAGUICIDAS ORGANOCLORADOS EN CULTIVOS DE FRUTILLA PARA ESTABLECER UN MANEJO INTEGRADO DE PLAGAS (MIP). [Tesis de pregrado, Universidad Técnica del Norte]. Repositorio Digital Universidad Técnica del Norte. http://repositorio.utn.edu.ec/handle/123456789/11676

Llumaquinga, G. (2018), DESARROLLO E IMPLEMENTACIÓN DE UNA NARIZ ELECTRÓNICA USANDO SENSORES SEMICONDUCTORES DE MADUREZ DE FRUTAS COMO PLÁTANO Y FRUTILLA. [Tesis de Pregrado, Universidad de las Fuerzas Armadas]. Repositorio Institucional de la Universidad de las Fuerzas Armadas ESPE. http://repositorio.espe.edu.ec/xmlui/bitstream/handle/21000/14912/T-ESPE-040268.pdf?sequence=1&isAllowed=y

Mazon-Olivo, B. & Pan, A. "Internet of Things: State-of-the-art, Computing Paradigms and Reference Architectures," in IEEE Latin America Transactions, vol. 20, no. 1, pp. 49-63, Jan. 2022, doi: 10.1109/TLA.2022.9662173

Ministerio de Salud Pública. (21 de Enero de 2021). *Ministerio de Salud Pública*. Obtenido de Ministerio de Salud Pública: https://www.salud.gob.ec/wp-content/uploads/2021/01/Etas-SE-03.pdf

MongoDB. (2022). *MongoDB*. Obtenido de MongoDB: https://www.mongodb.com/cloud-explained/iot-architecture

Naeem, M., Rizvi, S. T. H., & Coronato, A. (2020). A Gentle Introduction to Reinforcement Learning and its Application in Different Fields. *IEEE Access*, *8*, 209320–209344. https://doi.org/10.1109/ACCESS.2020.3038605

Nieto, B., & Rangel, J. (2022). *Vista de Sistema de visión artificial para gestión de calidad del Banano Cavendish en etapa de postcosecha*. https://revistas.utp.ac.pa/index.php/ric/article/view/3670/4248

Noticias ONU. (7 de Junio de 2022). *Noticias ONU*. Obtenido de Noticias ONU: https://news.un.org/es/story/2022/06/1509842

Nyuytiymbiy, K. (2020). *Parameters, Hyperparameters, Machine Learning | Towards Data Science*. https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac

OpenGenus IQ. (2023). *MobileNetV2 architecture*. https://iq.opengenus.org/mobilenetv2-architecture/#google_vignette

Osman, A., Abid, U., Gemma, L., Perotto, M., & Brunelli, D. (2021). *TinyML Platforms Benchmarking*.

Raspberry. (2021). *Raspberry Pi*. https://raspberrypi.cl/

Romero, B. (2020). Una introducción a los modelos de Machine Learning. En *Exploraciones, intercambios y relaciones entre el diseño y la tecnología*. https://doi.org/10.16/CSS/JQUERY.DATATABLES.MIN.CSS

Seidaliyeva, U., Akhmetov, D., Ilipbayeva, L., & Matson, E. (2020). *(PDF) Real-Time and Accurate Drone Detection in a Video with a Static Background*. https://www.researchgate.net/publication/342856036_Real-Time_and_Accurate_Drone_Detection_in_a_Video_with_a_Static_Background

Sik-Ho, T. (2020). *Reading: PANet — Path Aggregation Network, 1st Place in COCO 2017 Challenge (Instance Segmentation) | by Sik-Ho Tsang | Becoming Human: Artificial Intelligence Magazine*. https://becominghuman.ai/reading-panet-path-aggregation-network-1st-place-in-coco-2017-challenge-instance-segmentation-fe4c985cad1b

Soto, F., Tramón, C., Aqueveque, P., & De Bruijn, J. (2018). MICROORGANISMOS ANTAGONISTAS QUE INHIBEN EL DESARROLLO DE PATÓGENOS EN POST-COSECHA DE LIMONES (Citrus limon L.) . *CHILEAN JOURNAL OF AGRICULTURAL & ANIMAL SCIENCES*, 174-180.

Sutton, R. S., & Barto, A. G. (2015). *Reinforcement Learning: An Introduction Second edition, in progress*.

TinyML (2022). *TinyML*. Obtenido de TinyML: https://www.tinyml.org/

Trigos, A., Ramírez K., & Salinas, A. (2008). Presencia de hongos fitopatógenos en frutas y hortalizas y su relación en la seguridad alimentaria. *Revista mexicana de micología*, *28*(spe), 125-129. Recuperado en 21 de julio de 2022, de http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0187-31802008000300015&lng=es&tlng=es.

Venegas, V. (2019). DISEÑO DE UN SISTEMA DE DESPENSA INTELIGENTE PARA REFRIGERADORA QUE PERMITA PRIORIZAR LA COMPRA DE ALIMENTOS EN EL HOGAR. [Tesis de pregrado, Universidad Técnica del Norte]. Repositorio Digital Universidad Técnica del Norte. http://repositorio.utn.edu.ec/handle/123456789/8867

Villalba, E. (2022). *Metodologías y procesos de análisis de software*. http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/175/A5%20Cap%C3%ADtulo%202.pdf

Xu, R., Lin, H., Lu, K., Cao, L., & Liu, Y. (2021). A forest fire detection system based on ensemble learning. *Forests*, *12*(2), 1–17. https://doi.org/10.3390/F12020217

Yousuf, T., Mahmoud, R., Aloul, F., & Zualkernan, I. (2015). Internet of Things (IoT) Security: Current Status, Challenges and Countermeasures. *International Journal for Information Security Research*, *5*(4), 608–616. https://doi.org/10.20533/IJISR.2042.4639.2015.0070

ANEXOS

Anexo 1. Código de la aplicación script del proceso de detección de imagen.

```
# Erazo Carlos - Script de detecciòn de imagenes

import sys
sys.path.insert(0, './yolov5')
import argparse
import os
import platform
import sys
from pathlib import Path
from os import remove # Librebria para borrar archivos y directorios en Python
from os import system #Librería para ejecutar comandos de consola mediante el uso de Python
import torch
import datetime
from datetime import datetime
from pyrebase import pyrebase

system("rm -r /home/carlos/yolo/runs/detect/exp") # cambiar cuando se pase a raspberry
# Variables de estado de madurez y tipo de fruta


# Obtener la fecha y hora actual
fecha_hora_actual = datetime.now()

# Formatear la fecha y la hora por separado
fecha_formateada = fecha_hora_actual.strftime("%Y-%m-%d")
hora_formateada = fecha_hora_actual.strftime("%H:%M:%S")

# Concatenar la fecha y la hora
fecha_hora = hora_formateada + " - " + fecha_formateada

# Credenciales de Firebase
firebaseConfig = {
  'apiKey': "API_KEY",
  'authDomain': "fir-prueba-4ad44.firebaseapp.com",
  'databaseURL': "https://fir-prueba-4ad44-default-rtdb.firebaseio.com",
  'projectId': "fir-prueba-4ad44",
  'storageBucket': "fir-prueba-4ad44.appspot.com",
  'messagingSenderId': "393050977459",
  'appId': "1:393050977459:web:c22eee44bfedfbf804ecc6",
  'measurementId': "G-7X32Q0V2VF"
}

# Conexión e inicialización con la base de datos
firebase = pyrebase.initialize_app(firebaseConfig)
db = firebase.database()
```

```
storage = firebase.storage()


FILE = Path(__file__).resolve()
ROOT = FILE.parents[0]  # YOLOv5 root directory
if str(ROOT) not in sys.path:
   sys.path.append(str(ROOT))  # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative

# Importación de librerías para el uso de modelos YOLO
from models.common import DetectMultiBackend
from utils.dataloaders import IMG_FORMATS, VID_FORMATS, LoadImages, LoadScreenshots,
LoadStreams
from utils.general import (LOGGER, Profile, check_file, check_img_size, check_imshow,
check_requirements, colorstr, cv2,
                   increment_path, non_max_suppression, print_args, scale_boxes, strip_optimizer,
xyxy2xywh)
from utils.plots import Annotator, colors, save_one_box
from utils.torch_utils import select_device, smart_inference_mode


# Definición de parámetros de la detección
@smart_inference_mode()
def run(
      weights=ROOT / 'best.pt',  # Ruta del modelo
      source=ROOT / 'imagen.jpg',  # Archivo de imagen
      data=ROOT / 'data/coco128.yaml',  # Ruta del archivo dataset.yaml
      imgsz=(640, 640),  # Tamaño de la imagen para inferencia (alto, ancho)
      conf_thres=0.4,  # Umbral de confianza (confianza mínima) 0.25
      iou_thres=0.45,  # Umbral de IoU
      max_det=1000,  # Detecciones máximas por imagen
      device='',  # Dispositivo de procesamiento (cuda, por ejemplo, 0 para GPU o 'cpu' para CPU)
      view_img=False,  # Mostrar resultados
      save_txt=False,  # Guardar resultados en un archivo *.txt
      save_conf=False,  # Guardar confianzas en --save-txt etiquetas
      save_crop=False,  # Guardar recortes de predicciones
      nosave=False,  # No guardar imágenes/videos
      classes=None,  # Filtrar por clase: --class 0, o --class 0 2 3
      agnostic_nms=False,  # NMS (Supresión de No Máximo) // detección de objetos redundantes
      augment=False,  # Inferencia aumentada
      visualize=False,  # Visualizar características
      update=False,  # Actualizar todos los modelos
      project=ROOT / 'runs/detect',  # Guardar resultados en la carpeta del proyecto/nombre runs/detect
      name='exp',  # Nombre del proyecto/nombre para guardar resultados #exp
      exist_ok=False,  # Si el proyecto/nombre existe, no incrementarlo
      line_thickness=3,  # Grosor de línea del cuadro delimitador (píxeles)
      hide_labels=False,  # Ocultar etiquetas
      hide_conf=False,  # Ocultar confianzas
      half=False,  # Usar inferencia en media precisión FP16
):
```

```
# Carga de imagenes
source = str(source)
save_img = not nosave and not source.endswith('.txt')
is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
if is_url and is_file:
    source = check_file(source)

# Directorios
save_dir = increment_path(Path(project) / name, exist_ok=exist_ok)
(save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  # crea un directorio

# Cargar modelo
device = select_device(device)
model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data, fp16=half)
stride, names, pt = model.stride, model.names, model.pt
imgsz = check_img_size(imgsz, s=stride)  # verifica el tamaño de la imagen

# Cargar datos
bs = 1  # batch_size
dataset = LoadImages(source, img_size=imgsz, stride=stride, auto=pt, vid_stride=vid_stride)
vid_path, vid_writer = [None] * bs, [None] * bs



# Inferencia
model.warmup(imgsz=(1 if pt or model.triton else bs, 3, *imgsz))  # warmup
seen, windows, dt = 0, [], (Profile(), Profile(), Profile())
estadoMadurez1 = ""
estadoMadurez2 = ""
estadoMadurez3 = ""
estadoMadurez4 = ""
estadoMadurez5 = ""
estadoMadurez6 = ""
estadoMadurez7 = ""
estadoMadurez8 = ""
estadoMadurez9 = ""
estadoMadurez10 = ""
tipoFruta1 = ""
tipoFruta2 = ""

for path, im, im0s, vid_cap, s in dataset:
    with dt[0]:
        im = torch.from_numpy(im).to(model.device)
        im = im.half() if model.fp16 else im.float()
        im /= 255  # 0 - 255 to 0.0 - 1.0
        if len(im.shape) == 3:
            im = im[None]

    # Inferencia
    with dt[1]:
        visualize = increment_path(save_dir / Path(path).stem, mkdir=True) if visualize else False
```

```
            pred = model(im, augment=augment, visualize=visualize)

        # NMS (Reducción de detecciones redundantes)
        with dt[2]:
            pred = non_max_suppression(pred, conf_thres, iou_thres, classes, agnostic_nms,
max_det=max_det)

        # Proceso de detección
        for i, det in enumerate(pred):  # per image
            seen += 1
            p, im0, frame = path, im0s.copy(), getattr(dataset, 'frame', 0)

            p = Path(p)  # to Path
            save_path = str(save_dir / p.name)
            txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mode == 'image' else f'_{frame}')
            s += '%gx%g ' % im.shape[2:]
            gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]
            imc = im0.copy() if save_crop else im0
            annotator = Annotator(im0, line_width=line_thickness, example=str(names))

            if len(det):
                # Rescalamiento de las cajas delimitadoras
                det[:, :4] = scale_boxes(im.shape[2:], det[:, :4], im0.shape).round()

                # Impresión de los resultados
                for c in det[:, 5].unique():
                    n = (det[:, 5] == c).sum()  # Detecciones por clase
                    s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # Adición del nombre de la clase

                # Trazado de las cajas delimitadoras
                for *xyxy, conf, cls in reversed(det):
                    if save_txt:  # Condicional para la escritura de las cajas delimitadoras
                        xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()
                        line = (cls, *xywh, conf) if save_conf else (cls, *xywh)  # Formato de etiqueta
                        with open(f'{txt_path}.txt', 'a') as f:
                            f.write(('%g ' * len(line)).rstrip() % line + '\n')

                    if save_img or save_crop or view_img:  # Escritura en el archivo de imagen

                        c = int(cls)
                        label = None

                        # Condicionales según el estado de madurez
                        if names[c]== "unripe" or names[c] == "freshunripe" or names[c] == "freshripe"
                            or names[c]== "ripe" or names[c] == "overripe" or names[c]== "rotten":
                            tipoFruta1 = "Platanos"
                            db.child("datos/tipoFruta1").set(tipoFruta1)
                            if names[c] == "unripe":
                                annotator.box_label(xyxy, label, (255, 0, 255))
                                estadoMadurez1 = "Inmaduro"
                            elif names[c] == "freshunripe":
```

```
                        annotator.box_label(xyxy, label, (0, 100, 0))
                        estadoMadurez2 = "Fresco inmaduro"
                    elif names[c] == "freshripe":
                        annotator.box_label(xyxy, label, (128, 0, 128))
                        estadoMadurez3 = "Fresco"
                    elif names[c] == "ripe":
                        annotator.box_label(xyxy, label, (0, 102, 255))
                        estadoMadurez4 = "Maduro"
                    elif names[c] == "overripe":
                        annotator.box_label(xyxy, label, (255, 255, 0))  #(235, 206, 135)
                        estadoMadurez5 = "Maduracion excesiva"
                    elif names[c]== "rotten":
                        annotator.box_label(xyxy, label, (0, 0, 255))
                        estadoMadurez6 = "Podrido"
                if names[c] == "inmaduro" or names[c]== "maduro" or names[c]== "demasiadoM" or
names[c] == "podrido":
                    tipoFruta2 = "Fresas"
                    #db.child("datos/tipoFruta2").set(tipoFruta2)
                    print(tipoFruta2)
                    if names[c]== "inmaduro":
                        annotator.box_label(xyxy, label, (140, 0, 179))
                        estadoMadurez7 = "Inmaduro"
                    elif names[c]== "maduro":
                        annotator.box_label(xyxy, label, (128, 0, 0))
                        estadoMadurez8 = "Maduro"
                    elif names[c] == "demasiadoM":
                        annotator.box_label(xyxy, label, (42, 42, 165))
                        estadoMadurez9 = "Demasiado maduro"
                    elif names[c] == "podrido":
                        annotator.box_label(xyxy, label, (51, 51, 51))
                        estadoMadurez10 = "Podrido"

            if save_crop:
                save_one_box(xyxy, imc, file=save_dir / 'crops' / names[c] / f'{p.stem}.jpg', BGR=True)

        #
        im0 = annotator.result()

        # Guardar resultados de la detección
        if save_img:
            if dataset.mode == 'image':

                # Actualizar los datos en Firebase
                datos = {"estadoMadurez1": estadoMadurez1, "estadoMadurez2": estadoMadurez2,
"estadoMadurez3": estadoMadurez3,
                         "estadoMadurez4": estadoMadurez4,"estadoMadurez5": estadoMadurez5,
"estadoMadurez6": estadoMadurez6,
                         "estadoMadurez7": estadoMadurez7, "estadoMadurez8": estadoMadurez8,
"estadoMadurez9": estadoMadurez9,
                         "estadoMadurez10": estadoMadurez10, "tipoFruta1": tipoFruta1, "tipoFruta1":
tipoFruta1, "fecha_hora": fecha_hora}
```

```
                # Escritura en la base de datos
                db.child("datos").update(datos)
                # Almacenamiento de la imagen con las detecciones
                cv2.imwrite(save_path, im0)

# Directorios de las imagenes
imagen = '/home/pi/yolo/runs/detect/exp/imagen.jpg'
dir_imagen = '/images/imagen.jpg'


run()

# Subir imagen a Firebase Storage
storage.child(dir_imagen).put(imagen)
```

Anexo 2. Código de la aplicación móvil

```
package com.example.firebaseprueba;
import android.app.ProgressDialog;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.NotificationCompat;
import androidx.core.content.ContextCompat;
import android.Manifest;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.os.Build;
import android.text.SpannableString;
import android.text.Spanned;
import android.text.style.BackgroundColorSpan;
import android.view.View;
import android.widget.Toast;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import androidx.annotation.NonNull;
import androidx.navigation.ui.AppBarConfiguration;
import com.example.firebaseprueba.databinding.ActivityMainBinding;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
```

```
import com.google.firebase.storage.FileDownloadTask;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.database.DatabaseReference; //Yo la agregue
import android.widget.Button;
import android.widget.TextView;

import java.io.File;
import java.io.IOException;

public class MainActivity extends AppCompatActivity {

    //Variables de la actividad
    ActivityMainBinding binding;
    StorageReference storageReference;
    ProgressDialog progressDialog;
    DatabaseReference mDatabase;

    TextView report_TextViewData;
    TextView textPlatanos;
    TextView textFresas;
    TextView recomendacion;

    //Variables de ejecución de la aplicación
    boolean isFirstRun = true;
    String KEY_FIRST_RUN = "firstRun";

    int REQUEST_CODE = 200;
    @RequiresApi(api = Build.VERSION_CODES.TIRAMISU)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        verificar();
        if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
            NotificationChannel channel = new NotificationChannel("channel1", "Channel1",
NotificationManager.IMPORTANCE_HIGH);
            NotificationManager notificationManager = getSystemService(NotificationManager.class);
            notificationManager.createNotificationChannel(channel);

        }

        //fruta_TextViewData = (TextView) findViewById(R.id.etimageId);
        mDatabase = FirebaseDatabase.getInstance().getReference();
        //mBtnActualizarDatos = (Button) findViewById(R.id.getImage);
        report_TextViewData = (TextView) findViewById(R.id.date_report);
        //estadoM_TextViewData = (TextView) findViewById(R.id.estadoMadurez);
        textPlatanos = (TextView) findViewById(R.id.textPlat);
```

```
        textFresas = (TextView) findViewById(R.id.textFres);
        recomendacion = (TextView) findViewById(R.id.recomendacion);

        if (savedInstanceState != null) {
            isFirstRun = savedInstanceState.getBoolean(KEY_FIRST_RUN);
        }

        if (isFirstRun) {
            iniciarAplicacion();
            isFirstRun = false;
        }

    }


    public void notificationRotten(View view){
        //final String CHANNEL_ID = "channel1";

        NotificationCompat.Builder builder = new NotificationCompat.Builder(this, "channel1")
                .setSmallIcon(R.drawable.baseline_notifications_24)
                .setContentTitle("Sistema de Visión Artificial")
                .setContentText("Estado de madurez de las frutas")
                .setAutoCancel(true)
                .setColor(Color.RED)
                .setStyle(new NotificationCompat.BigTextStyle()
                        .setBigContentTitle("Estado de madurez de las frutas")
                        .bigText("Alerta el estado de las frutas es podrido"));
        NotificationManager notificationManager = getSystemService(NotificationManager.class);

        Intent intent = new Intent(getApplicationContext(), MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);
        //intent.putExtra("data","Hola como estas");

        PendingIntent pendingIntent = PendingIntent.getActivity(getApplicationContext(),
                0,intent,PendingIntent.FLAG_MUTABLE);
        builder.setContentIntent(pendingIntent);
        //NotificationManager notificationManager1 = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);


        notificationManager.notify(1,builder.build());

    }

    public void notificationOverripe(View view){
        //final String CHANNEL_ID = "channel1";

        NotificationCompat.Builder builder = new NotificationCompat.Builder(this, "channel1")
                .setSmallIcon(R.drawable.baseline_notifications_24)
                .setContentTitle("Sistema de Visión Artificial")
                .setContentText("Estado de madurez de las frutas")
```

```
            .setAutoCancel(true)
            .setColor(Color.GREEN)
            .setStyle(new NotificationCompat.BigTextStyle()
                .setBigContentTitle("Estado de madurez de las frutas")
                .bigText("Alerta el estado de las frutas es demasiado maduro"));
        NotificationManager notificationManager = getSystemService(NotificationManager.class);

        Intent intent = new Intent(getApplicationContext(), MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);

        PendingIntent pendingIntent = PendingIntent.getActivity(getApplicationContext(),
            0,intent,PendingIntent.FLAG_MUTABLE);
        builder.setContentIntent(pendingIntent);



        notificationManager.notify(1,builder.build());

    }

    @RequiresApi(api = Build.VERSION_CODES.TIRAMISU)
    public void verificar() {
        int permisosNotificaciones = ContextCompat.checkSelfPermission(this,
Manifest.permission.POST_NOTIFICATIONS);

        if (permisosNotificaciones == PackageManager.PERMISSION_GRANTED){
            Toast.makeText(this, "Permisos de notificaciones concedidas", Toast.LENGTH_SHORT).show();
        } else {
            requestPermissions(new String[]{Manifest.permission.POST_NOTIFICATIONS},
REQUEST_CODE);
        }
    }


    @Override
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);

        outState.putBoolean(KEY_FIRST_RUN, isFirstRun);
    }

    @Override
    protected void onRestart() {
        super.onRestart();

        iniciarAplicacion();
    }

    public void iniciarAplicacion(){

        progressDialog = new ProgressDialog(MainActivity.this);
```

```
        progressDialog.setMessage("Obteniendo reporte... Por favor espere.");
        progressDialog.setCancelable(false);
        progressDialog.show();

        //String imageID = binding.etimageId.getText().toString();
        String imageID = "imagen";

        storageReference = FirebaseStorage.getInstance().getReference("images/"+imageID+".jpg");


        //mDatabase.child("datos").child("tipoFruta").addListenerForSingleValueEvent(new
ValueEventListener() {
        mDatabase.child("datos").addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {

                if (snapshot.exists()){

                    // definición de variables para los 10 diferentes estados
                    String tipoFruta1 = snapshot.child("tipoFruta1").getValue().toString();
                    String tipoFruta2 = snapshot.child("tipoFruta2").getValue().toString();
                    String estadoMadurez1 = snapshot.child("estadoMadurez1").getValue().toString();
                    String estadoMadurez2 = snapshot.child("estadoMadurez2").getValue().toString();
                    String estadoMadurez3 = snapshot.child("estadoMadurez3").getValue().toString();
                    String estadoMadurez4 = snapshot.child("estadoMadurez4").getValue().toString();
                    String estadoMadurez5 = snapshot.child("estadoMadurez5").getValue().toString();
                    String estadoMadurez6 = snapshot.child("estadoMadurez6").getValue().toString();
                    String estadoMadurez7 = snapshot.child("estadoMadurez7").getValue().toString();
                    String estadoMadurez8 = snapshot.child("estadoMadurez8").getValue().toString();
                    String estadoMadurez9 = snapshot.child("estadoMadurez9").getValue().toString();
                    String estadoMadurez10 = snapshot.child("estadoMadurez10").getValue().toString();
                    String fecha_hora = snapshot.child("fecha_hora").getValue().toString();
                    report_TextViewData.setText(fecha_hora);
                    textPlatanos.setText("  **"+tipoFruta1+"**"+"\n");
                    textFresas.setText("  **"+tipoFruta2+"**"+"\n");


                    // Condicionales para cada uno de los estados de madurez
                    if(estadoMadurez1.equals("Inmaduro")){

                        String frase1 = "Inmaduro\n";
                        String color1 = "#00FF00"; // Color en formato hexadecimal

                        // Agrega dos espacios en blanco antes de la frase
                        SpannableString spannableString = new SpannableString("  " +  frase1);
                        BackgroundColorSpan backgroundColorSpan = new
BackgroundColorSpan(Color.parseColor(color1));
                        // Se agrega el color respectivo junto al estado de madurez respectivo
                        spannableString.setSpan(backgroundColorSpan, 0, 1,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
                        textPlatanos.append(spannableString);
```

```
                recomendacion.setText("Estado de la fruta óptimo");

            }
            if(estadoMadurez2.equals("Fresco inmaduro")){
                String frase2 = "Fresco inmaduro\n";
                String color2 = "#006400"; // Color en formato hexadecimal

                SpannableString spannableString = new SpannableString("  " +  frase2); // Agrega dos
espacios en blanco antes de la frase
                BackgroundColorSpan backgroundColorSpan = new
BackgroundColorSpan(Color.parseColor(color2));

                spannableString.setSpan(backgroundColorSpan, 0, 1,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE); // El círculo se aplica al primer carácter (espacio en
blanco)
                textPlatanos.append(spannableString);
                recomendacion.setText("Estado de la fruta óptimo");
            }
            if(estadoMadurez3.equals("Fresco")){
                String frase3 = "Fresco\n";
                String color3 = "#800080"; // Color en formato hexadecimal

                SpannableString spannableString = new SpannableString("  " +  frase3); // Agrega dos
espacios en blanco antes de la frase
                BackgroundColorSpan backgroundColorSpan = new
BackgroundColorSpan(Color.parseColor(color3));

                spannableString.setSpan(backgroundColorSpan, 0, 1,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE); // El círculo se aplica al primer carácter (espacio en
blanco)
                textPlatanos.append(spannableString);
                recomendacion.setText("Estado de la fruta óptimo");
            }
            if(estadoMadurez4.equals("Maduro")){
                String frase4 = "Maduro\n";
                String color4 = "#FF6600"; // Color en formato hexadecimal

                SpannableString spannableString = new SpannableString("  " +  frase4); // Agrega dos
espacios en blanco antes de la frase
                BackgroundColorSpan backgroundColorSpan = new
BackgroundColorSpan(Color.parseColor(color4));

                spannableString.setSpan(backgroundColorSpan, 0, 1,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE); // El círculo se aplica al primer carácter (espacio en
blanco)
                textPlatanos.append(spannableString);
                recomendacion.setText("Estado de la fruta óptimo");
            }

            if (estadoMadurez7.equals("Inmaduro")) {
                //recomend_TextViewData.setText("RECOMENDACIÓN: Desechar la o las frutas
```

```
encerradas en el recuadro rojo ⚠️ ⚠️ ");
                String frase7 = "Inmaduro\n";
                String color7 = "#B3008C"; // Color en formato hexadecimal

                SpannableString spannableString = new SpannableString("  " +  frase7); // Agrega dos
espacios en blanco antes de la frase
                BackgroundColorSpan backgroundColorSpan = new
BackgroundColorSpan(Color.parseColor(color7));

                spannableString.setSpan(backgroundColorSpan, 0, 1,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE); // El círculo se aplica al primer carácter (espacio en
blanco)
                textFresas.append(spannableString);
                recomendacion.setText("Estado de la fruta óptimo");


            }
            if (estadoMadurez8.equals("Maduro")) {
                //recomend_TextViewData.setText("RECOMENDACIÓN: Desechar la o las frutas
encerradas en el recuadro rojo ⚠️ ⚠️ ");
                String frase8 = "Maduro\n";
                String color8 = "#000080"; // Color en formato hexadecimal

                SpannableString spannableString = new SpannableString("  " +  frase8); // Agrega dos
espacios en blanco antes de la frase
                BackgroundColorSpan backgroundColorSpan = new
BackgroundColorSpan(Color.parseColor(color8));

                spannableString.setSpan(backgroundColorSpan, 0, 1,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE); // El círculo se aplica al primer carácter (espacio en
blanco)
                textFresas.append(spannableString);
                recomendacion.setText("Estado de la fruta óptimo");



            }

            if (estadoMadurez5.equals("Maduracion excesiva")) {

                String frase5 = "Maduración excesiva\n";
                String color5 = "#00FFFF"; // Color en formato hexadecimal
                // Agrega dos espacios en blanco antes de la frase
                SpannableString spannableString = new SpannableString("  " +  frase5);
                BackgroundColorSpan backgroundColorSpan = new
BackgroundColorSpan(Color.parseColor(color5));
                // Se agrega el color respectivo junto al estado de madurez respectivo
                spannableString.setSpan(backgroundColorSpan, 0, 1,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
                textPlatanos.append(spannableString);
```

```
                recomendacion.setText("ALERTA: Los plátanos señalados con el recuadro cian, tienen
una maduración excesiva\n" +
                        "RECOMENDACIÓN: ¡No desperdicies! Puedes hacer batidos o pacakes de plátano
con estas frutas");
                notificationOverripe(null);

            }
            if (estadoMadurez6.equals("Podrido")) {
                String frase6 = "Podrido\n";
                String color6 = "#FF0000"; // Color en formato hexadecimal
                // Agrega dos espacios en blanco antes de la frase
                SpannableString spannableString = new SpannableString("  " +  frase6);
                BackgroundColorSpan backgroundColorSpan = new
BackgroundColorSpan(Color.parseColor(color6));
                // Se agrega el color respectivo junto al estado de madurez respectivo
                spannableString.setSpan(backgroundColorSpan, 0, 1,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
                textPlatanos.append(spannableString);
                recomendacion.setText("ALERTA: Los plátanos señalados con el recuadro rojo, se
encuentran podridos\n" +
                        "RECOMENDACIÓN: Revisar manualmente cada uno de los plátanos señalados y
desecharlos de ser necesario");
                notificationRotten(null);
            }

            if (estadoMadurez9.equals("Demasiado maduro")) {
                //recomend_TextViewData.setText("RECOMENDACIÓN: Desechar la o las frutas
encerradas en el recuadro rojo ⚠️ ⚠️ ");
                String frase9 = "Maduración excesiva\n";
                String color9 = "#A52A2A"; // Color en formato hexadecimal

                SpannableString spannableString = new SpannableString("  " +  frase9); // Agrega dos
espacios en blanco antes de la frase
                BackgroundColorSpan backgroundColorSpan = new
BackgroundColorSpan(Color.parseColor(color9));

                spannableString.setSpan(backgroundColorSpan, 0, 1,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE); // El círculo se aplica al primer carácter (espacio en
blanco)
                textFresas.append(spannableString);
                recomendacion.setText("ALERTA: Las fresas señaladas con el recuadro marrón, tienen
una maduración excesiva\n" +
                        "RECOMENDACIÓN: ¡No desperdicies! Puedes hacer mermelada o batidos de fresa
con estas frutas");
                notificationOverripe(null);


            }
            if (estadoMadurez10.equals("Podrido")) {
                //recomend_TextViewData.setText("RECOMENDACIÓN: Desechar la o las frutas
```

```
encerradas en el recuadro rojo ⚠️ ⚠️ ");
                String frase10 = "Podrido\n";
                String color10 = "#333333"; // Color en formato hexadecimal

                SpannableString spannableString = new SpannableString("  " +  frase10); // Agrega dos
espacios en blanco antes de la frase
                BackgroundColorSpan backgroundColorSpan = new
BackgroundColorSpan(Color.parseColor(color10));

                spannableString.setSpan(backgroundColorSpan, 0, 1,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE); // El círculo se aplica al primer carácter (espacio en
blanco)
                textFresas.append(spannableString);

                recomendacion.setText("ALERTA: Las fresas señaladas con el recuadro gris oscuro, se
encuentran podridas\n" +
                        "RECOMENDACIÓN: Revisar manualmente cada una de las fresas señaladas y
desecharlas de ser necesario");
                notificationRotten(null);

            }

            if(estadoMadurez6.equals("Podrido") && estadoMadurez10.equals("Podrido")){
                recomendacion.setText("ALERTA: Los plátanos y las fresas señaladas con los recuadros
rojo y gris oscuro respectivamente, se encuentran podridos\n" +
                        "RECOMENDACIÓN: Revisar manualmente cada uno de las frutas señaladas y
desecharlas de ser necesario");
            }
            if(estadoMadurez5.equals("Maduracion excesiva") && estadoMadurez9.equals("Demasiado
maduro")){
                recomendacion.setText("ALERTA: Los plátanos y las fresas señaladas con los recuadros
cian y marrón respectivamente, tienen una maduración excesiva\n" +
                        "RECOMENDACIÓN: ¡No desperdicies! Puedes hacer mermelada, batidos o pacakes
con estas frutas");
            } if(estadoMadurez5.equals("Maduracion excesiva") &&
estadoMadurez10.equals("Podrido")){
                recomendacion.setText("ALERTA: Los plátanos señalados con el recuadro cian, tienen
una maduración excesiva y las fresas señaldas con el recuadro gris" +
                        " oscuro, se encuentran podridas\n" + "RECOMENDACIÓN: ¡No desperdicies!
Separa los plátanos de las fresas podridas, y con estas frutas puedes " +
                        "hacer batidos o un pacakes");
            }
            if(estadoMadurez6.equals("Podrido") && estadoMadurez9.equals("Demasiado maduro")){
                recomendacion.setText("ALERTA: Los plátanos señalados con el recuadro rojo, se
encuentran podridos y las fresas señaladas con el recuadro gris oscuro" +
                        ", tienen una maduración excesiva\n" + "RECOMENDACIÓN: ¡No desperdicies!
Separa las fresas de los plátanos podridos, y con estas frutas puedes " +
                        "hacer mermelada o batidos");
            } if(estadoMadurez9.equals("Demasiado maduro") &&
estadoMadurez10.equals("Podrido")){
                recomendacion.setText("ALERTA: Las fresas señaladas con el recuadro marrón, tienen un
```

```
maduración excesiva y las fresas señaladas con el recuadro gris oscuro" +
                        ", se encuentran podridas\n" + "RECOMENDACIÓN: ¡No desperdicies! Separa las
fresas podridas de las fresas con maduración excesiva, y con estas frutas " +
                        "puedes hacer mermelada o batidos");
                }
                if(estadoMadurez5.equals("Maduracion excesiva") && estadoMadurez6.equals("Podrido")){
                    recomendacion.setText("ALERTA: Los plátanos señalados con el recuadro cian, tienen
una maduración excesiva y los plátanos señalados con el recuadro rojo, se" +
                        " encuentran podridos\n" + "RECOMENDACIÓN: ¡No desperdicies! Separa los
plátanos podridos de los plátanos con maduración excesiva, y con estas frutas " +
                        "puedes hacer batidos o pancakes");
                }



            }

        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });

    try {
        File localfile = File.createTempFile("tempfile", ".jpg");
        storageReference.getFile(localfile)
                .addOnSuccessListener(new OnSuccessListener<FileDownloadTask.TaskSnapshot>() {
                    @Override
                    public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {

                        if(progressDialog.isShowing())
                            progressDialog.dismiss();

                        Bitmap bitmap = BitmapFactory.decodeFile(localfile.getAbsolutePath());
                        int desiredWidth = 1000;
                        int desiredHeight = 1000;

                        // Calcula los factores de escala para redimensionar la imagen
                        float widthFactor = (float) desiredWidth / bitmap.getWidth();
                        float heightFactor = (float) desiredHeight / bitmap.getHeight();

                        // Calcula el factor de escala final basado en el factor de escala más pequeño
                        float scaleFactor = Math.min(widthFactor, heightFactor);

                        // Calcula el ancho y alto finales de la imagen redimensionada
                        int scaledWidth = Math.round(bitmap.getWidth() * scaleFactor);
                        int scaledHeight = Math.round(bitmap.getHeight() * scaleFactor);

                        // Crea el bitmap redimensionado utilizando la escala calculada
```

```
                    Bitmap resizedBitmap = Bitmap.createScaledBitmap(bitmap, scaledWidth,
scaledHeight, true);

                    // Muestra el bitmap redimensionado en el ImageView
                    binding.imageView.setImageBitmap(resizedBitmap);
                    //binding.imageView.setImageBitmap(bitmap);

                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {

                    if (progressDialog.isShowing())
                        progressDialog.dismiss();

                    Toast.makeText(MainActivity.this, "", Toast.LENGTH_SHORT).show();

                }
            });


        } catch (IOException e) {
            throw new RuntimeException(e);
        }

    }

}
```