UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE POSTGRADO

MAESTRÍA EN INGENIERÍA MECATRÓNICA

"IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN POR VISIÓN ARTIFICIAL EN LA ETAPA DE RECOLECCIÓN DEL CULTIVO DE FRESAS."

Tesis de Maestría presentada en cumplimiento parcial de los requisitos de la Maestría en Mecatrónica: Mención Procesos Industriales

AUTOR: Ing. Dany Orlando Orbes Padilla

DIRECTOR: Ing. Marco Antonio Ciaccia Sortino, PhD. MSc.

IBARRA - ECUADOR

2022

CERTIFICACIÓN DE DIRECTOR DE TESIS

Como director del trabajo de investigación con el tema: **"IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN POR VISIÓN ARTIFICIAL EN LA ETAPA DE RECOLECCIÓN DEL CULTIVO DE FRESAS",** trabajo que fue realizado por Dany Orlando Orbes Padilla, previo a la obtención del título de Magister en mecatrónica mención procesos industriales, doy fe de que el trabajo mencionado cumple con los requisitos y méritos suficientes para ser apoyado públicamente en los tribunales para ser seleccionado oportunamente.

Ibarra, 24 de mayo 2022

Es todo lo que puedo certificar

Firmado electrónicamente por:
MARCO ANTONIO
CIACCIA SORTINO

Ing. Marco Antonio Ciaccia Sortino, PhD. MSc.

C.C.: 1756778252

Director de Tesis

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:	1002383402		
APELLIDOS Y NOMBRES:	Dany Orlando Orbes Padilla		
DIRECCIÓN:	Sánchez y Cifuentes 617 y Grijalva		
EMAIL:	doorbesp@utn.edu.ec		
TELÉFONO FIJO:	062953552	**TELÉFONO MÓVIL:**	0988223616

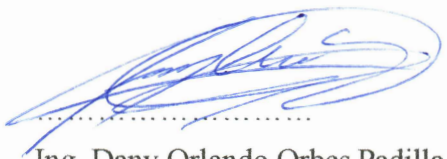
DATOS DE LA OBRA	
TÍTULO:	"IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN POR VISIÓN ARTIFICIAL EN LA ETAPA DE RECOLECCIÓN DEL CULTIVO DE FRESAS"
AUTOR:	Dany Orlando Orbes Padilla
FECHA:	24/05/2022
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	☐ PREGRADO ■ POSGRADO
TÍTULO POR EL QUE OPTA:	Magíster en Mecatrónica Mención Procesos Industriales
ASESOR / DIRECTOR:	Ing. Marco Antonio Ciaccia Sortino, PhD. MSc.

CONSTANCIAS

El autor manifiesta 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 el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 24 días del mes de mayo de 2022.

EL AUTOR:



Ing. Dany Orlando Orbes Padilla

REGISTRO BIBLIOGRÁFICO

Guía: POSGRADO – UTN

Fecha: Ibarra, 24 de mayo de 2022

Dany Orlando Orbes Padilla: "IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN POR VISIÓN ARTIFICIAL EN LA ETAPA DE RECOLECCIÓN DEL CULTIVO DE FRESAS." / **GRADO MAGISTER EN MECATRÓNICA MENCIÓN PROCESOS INDUSTRIALES.**

DIRECTOR: Ing. Marco Antonio Ciaccia Sortino, PhD. MSc.

El objetivo general de esta tesis fue: Implementar un sistema de detección y categorización con visión artificial en el proceso de cosecha del cultivo de fresas en un cultivo local de la región norte de Ecuador.

Entre los objetivos específicos se encuentran: Investigar sobre los diferentes procesos de cultivo que existen en Ecuador.

Diseñar un algoritmo de visión artificial para la detección y categorización de las fresas, tomando en cuenta los requerimientos de cosecha.

Validar el algoritmo de acuerdo con los estándares de calidad de cultivo de fresas.

Implementar este algoritmo en una placa informática equipada con un GPU, conectada a un dispositivo físico de recolección de fresas.

Firmado electrónicamente por:
MARCO ANTONIO CIACCIA SORTINO

Ing. Marco Antonio Ciaccia Sortino, PhD. MSc.

Director

Ing. Dany Orlando Orbes Padilla

Autor

DEDICATORIA

Este trabajo de investigación se la dedico a toda mi familia y amigos, que me acompañaron en el transcurso de esta nueva meta y en especial a mis hijos “los amo”.

Dany

AGRADECIMIENTOS

Agradezco de todo corazón, a mi Dios, por darme esta oportunidad de seguir con mis estudios, y a mi familia por apoyarme al 100% a seguir esta nueva meta, a mis amigos de la empresa INGENIOUS WORKS, que estuvieron acompañándome en todo el proceso de la maestría, a mi tutor y profesores que me han impartido diferentes conocimientos a lo largo de todos los módulos con paciencia y dedicación.

No quedan más palabras para decir todo lo que siento, “la gratitud es la memoria del corazón”.

Dany

ÍNDICE

ÍNDICES DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En el presente trabajo se implementa un sistema de detección y categorización con visión artificial en el proceso de cosecha del cultivo de fresas en la región norte de Ecuador, específicamente en una comunidad de la parroquia Andrade Marín. Para ello se realizó una recopilación de fotos de las fresas en el cultivo, generando un dataset en el cual se aplicó redes neuronales con un modelo de entrenamiento "*MobilenetV2*", para posteriormente cumplir con un procesamiento de imágenes en el cual se determina el centroide de la fruta por medio de programación gráfica. El funcionamiento del sistema de visión artificial inicia con la detención gráfica de la fresa, obteniendo su centroide y enviando las diferentes coordenadas de cada una de las fresas con un porcentaje de madurez mayor al 60%. Se define la primera coordenada como la más cercana al borde izquierdo de la imagen, y se envía este dato por comunicación serial a un dispositivo embebido que requiera esta información, y posteriormente se espera la respuesta del dispositivo, el cual informa que la fresa ya ha sido cosechada; este procedimiento se dará repetitivamente hasta que no existan más fresas en cuadro de la imagen. Como resultado se obtuvo un algoritmo con una perdida mínima del 1,7705599 y una velocidad de detección de 0,123 segundos, con una exactitud del 82,91%, precisión de un 80%, sensibilidad del 87% y especificidad del 79%, lo cual lo cataloga como un algoritmo de visión artificial confiable, que contribuirá a la detección de fresas en tiempo real.

Palabras clave: visión, Procesamiento, algoritmo, fresas, dataset

ABSTRACT

This work is focus on the detection and categorization system in the process of harvesting strawberries using artificial vision in the northern region of Ecuador, specifically ina community of the Andrade Marín sector. For this, we built a dataset with a compilation of photos of the strawberries in the crop, after that, we applied a neural network such as: "MobilenetV2" model, finally the image captured, was processed through artificial vision to find the centroid of the fruit. The artificial vision system begins with the graphic detection of the strawberry, obtaining its centroid and sending the different coordinates of each one of the strawberries with a maturity percentage over the 60%. The first coordinate is defined as the closest to the left border of the image, once processing this data is sent by serial communicationto an embedded device that requires this information, to make the fruit harvest; This procedurewill be repeated until there are no more strawberries in the image frame. the results were: the algorithm was a minimum loss of 1.7705599 and a detection speed of 0.123 seconds, with an accuracy of 82.91%, precision of 80%, sensitivity of 87% and specificity of 79%, in conclusionthe artificial vision algorithm is reliable, this system contributes to the detection of strawberriesin real time.

Keywords: vision, Processing, algorithm, strawberries, dataset

CAPÍTULO 1

INTRODUCCIÓN

En este capítulo se presenta el planteamiento del problema, los objetivos y la justificación del trabajo a desarrollar, además de los antecedentes relacionados con la investigación.

1.1 Problema de investigación

La fresa o fragaria es considerada como un fruto que viene de una planta prematura de alta producción, la cual posee un exquisito sabor y alto valor nutricional, además de ser muy solicitada en el mercado (Sandino et al., 2016). Este producto de alta demanda en Ecuador y el mundo, representa un gran reto para los productores, ya que se requiere una gran producción, alta calidad y por supuesto una rentabilidad aceptable.

La alta demanda exige un esfuerzo considerable que afecta significativamente la salud de los trabajadores y jornaleros involucrados en los procesos de cosecha, debido a las características que presenta las etapas de cultivo y recolección utilizada en Ecuador, las cuales se realizan de forma manual. Esto genera problemas de ergonomía en los trabajadores, quienes se movilizan con grandes cestos cargados, a lo largo de la zona de siembra, mientras permanecen en posición de flexión durante toda la jornada de recolección.

En el proceso de cosecha de esta fruta, hay puntos cruciales que deben ser considerados para cualquier evaluación aplicada, como por ejemplo su corto periodo de conservación, ya que es altamente perecible, por lo cual los cuidados que se deben aplicar para mantener su calidad y frescura son exhaustivos, demandando mucho más tiempo de cosecha, y requiriendo mayor cantidad de jornaleros dependiendo de la extensión del terreno, para satisfacer los tiempos de cosecha y mantenerse en el mercado. Como consecuencia se obtiene un índice de producción bajo, con una mayor inversión de dinero y baja rentabilidad.

En la actualidad los frutos a seleccionar dependen del nivel de maduración aceptado comercialmente, que se define por el color que ha alcanzado la superficie del fruto. En cuanto a ello para la comercialización, el estándar del fruto en Latinoamérica permite hasta 1/3 del

fruto coloreado, no obstante, esto puede cambiar de acuerdo con el país en el que se comercialice (Soto, 2018).

El proceso de cosechado cambia constantemente; la primera cosecha se realiza al tercer mes luego de la siembra y se continua con las mismas semanas durante 3 meses. A partir del cuarto mes se realizan dos cosechas por semana, ya que la maduración es mucho más rápida. La cosecha se realiza manualmente en las primeras horas de la mañana, para lo cual los trabajadores deben prepararse y utilizar guantes de látex, con la finalidad disminuir los posibles daños mecánicos que se pueda hacer en la fruta, además deben tener cuidado de tomar el fruto desde el péndulo del cáliz, y al retirarlo, girarlo para desprender la fruta (Faura, 2010).

Teniendo en cuenta todo lo expuesto anteriormente, el manejo tanto en la manipulación, empaque, transporte y almacenamiento de las fresas desde la cosecha, debe ser cuidadoso, ya que durante la etapa de comercialización se pueden producir algunos daños importantes como la reducción de firmeza, daño mecánico, pudrimiento y fermentación (Flores y Mora, 2015).

Actualmente no existen una implementación completa de un sistema de cosecha automatizado de fresa en Ecuador, ya sea por el tema de costos o por el tiempo de estudio que requiere su implementación. En el siguiente trabajo se desarrollará y se implementará un sistema de visión artificial que permita detectar y seleccionar la fresa madura, según los estándares de selección utilizados en Ecuador, el cual pudiera ser una de las etapas de una propuesta de equipo de cosechado automatizado adaptado a las condiciones de cultivo en Ecuador, que puede ser desarrollada a futuro.

1.2 Antecedentes

Durand - Petiteville et al., (2017) presentan un algoritmo de procesamiento de imágenes que extrae automáticamente las áreas carnosa o pulpa, a partir de imágenes de fresas, las cuales son capturadas por una cámara incluida en una máquina destapadora de esta fruta. El tipo de algoritmo propuesto fue RITHM, el cual procesa una imagen obtenida de la cámara, en un entorno normal de uso.

En el algoritmo se controla la iluminación basándose en una segmentación en el color y a la vez en varios pasos de selección de manchas. El objetivo de este trabajo era extraer tantos

píxeles de carne y cáliz como fuera posible, mientras se rechazan los píxeles pertenecientes a los anteriores, dando un enfoque propuesto basado en la segmentación del color de la imagen, en un color en el espacio bidimensional del área del objeto y seguido de una etapa de detección y selección de manchas. Utilizaron un conjunto de 250 imágenes para analizar la sensibilidad del algoritmo con respecto a los parámetros definidos por el usuario, y evaluar el rendimiento del proceso.

El trabajo evidencia la precisión con la que es empleada la visión artificial con un algoritmo de procesamiento de imágenes, utilizando un nuevo enfoque de segmentación del espacio de color bidimensional, seguido de detección y selección de machas y teniendo mejores resultados.

Por otro lado, Constante et al. (2016), aplicaron diferentes técnicas de visión artificial para la detección de características de las fresas utilizadas en la industria alimentaria, basándose en redes neuronales artificiales y arquitectura profunda. El algoritmo fue entrenado con aprendizaje compensado por ruido, siendo una combinación para una fuerte relación red – objeto, con el fin de reconocer las características complejas de la fresa bajo condiciones cambiantes de tamaño, iluminación y orientación. Aparte de esta modalidad utilizó la librería de open CV y un Data sheet, para entrenar la red neuronal artificial, detectando desde el inicio los bordes y la región. Hicieron pruebas en tiempo real con imágenes reales vía Stream. La investigación mencionada se enfoca en la detección de imágenes por aprendizaje compensado con redes neuronales, utilizando una red neuronal por Entrenamiento y Agentes Neuronales (Entrenables Aislamente, ANEA´s), creando fuertes relaciones entre las imágenes y los agentes y haciendo posible la clasificación de la fruta bajo diferentes condiciones morfológicas y de entorno.

En el caso de Khort et al., (2020), aplicaron un algoritmo de control automático, implementado el lenguaje orientado a objetos de alto nivel en Python 3.7.2, incluyendo operaciones en el cual determinan las coordenadas X e Y de una baya, su grado de madurez, así como el cálculo de la distancia desde la baya, encontrando la eficiencia de detección de bayas, su área y bordes con una cámara y la iluminación de 300 lux de la librería de OpenCV alcanzando el 94,6% de asertividad. Obtuvieron como resultado un algoritmo eficaz para cosechar fresas utilizando un manipulador robótico ubicado en un carro de exploración. Los

estudios experimentales realizados en este algoritmo demuestran que busca las fresas y al mismo tiempo determina su grado de madurez con un alto grado de precisión.

Asimismo, Saenz et al., (2013) realizaron un prototipo de robot recolector de fresas en sistemas hidropónicos de invernadero, donde dichos sistemas optimizan los recursos para obtener una mayor densidad de siembra y el mayor rendimiento por unidad de superficie, en este caso de la fresa. En tal sentido, la aplicación del prototipo facilita la detección de fresas en estado de madurez a partir de la tonalidad rojiza de las mismas, ayudando a optimizar los recursos y disminuir los costos de producción.

La manera en que se realizó este trabajo lleva a distintos procesamientos de imágenes, que permite establecer estándares de calidad en los productos. Por otro lado, ayuda al recolector de fresas en el proceso de selección, usando software gratuito como OpenCV y hardware gratuito como plataforma de Arduino, que permite aplicaciones en el campo agrícola sin necesidad de licencias de software o hardware costosos. El procesamiento mencionado no solo puede aplicarse para fresas sino también a otros cultivos de invernadero.

Los sistemas de visión artificial son herramientas poderosas para la inspección automática de frutas, así lo denominan Cubero et al., (2011), quienes realizaron un trabajo presentando avances en aplicaciones de visión artificial para sistemas automáticos del control de calidad de frutas y verduras. Las aplicaciones de tales sistemas incluyen clasificación, identificación y estimación de calidad a partir de parámetros externos o características internas, seguimiento de los procesos de la fruta durante el almacenamiento o la evaluación de tratamientos experimentales.

Las capacidades de un sistema de visión artificial van más allá de la limitada capacidad humana para evaluar los procesos a largo plazo de manera objetiva, o para apreciar eventos que tienen lugar fuera del campo electromagnético visible a su espectro. Tal es el caso de los espectros ultravioletas o infrarrojos, que permiten explorar defectos o características que el humano no puede ver. Los sistemas hiperespectrales proporcionan información sobre componentes individuales o daños que se puede percibir, sólo en determinadas longitudes de onda, y se puede utilizar como herramienta para desarrollar nuevos sistemas de visión por computadora adaptado a objetivos particulares.

1.3 Objetivos

1.3.1 Objetivo general

Implementar un sistema de detección y categorización con visión artificial en el proceso de cosecha del cultivo de fresas en un cultivo local de la región norte de Ecuador.

1.3.2 Objetivos específicos

- Investigar sobre los diferentes procesos de cultivo que existen en Ecuador.
- Diseñar un algoritmo de visión artificial para la detección y categorización de las fresas, tomando en cuenta los requerimientos de cosecha.
- Validar el algoritmo de acuerdo con los estándares de calidad de cultivo de fresas.
- Implementar este algoritmo en una placa informática equipada con un GPU, conectada a un dispositivo físico de recolección de fresas.

1.4 Justificación

Este trabajo se realizará con base a los requerimientos investigados en los diferentes tipos de procesos de cosecha de fresas a nivel nacional e internacional, asimismo implementado con las normativas de calidad de procesos industriales en productos alimenticios, lo cual garantiza su calidad en este proceso de detección para las cosechas en el cultivo de fresas en el norte del país.

En vista de que los mayores riesgos de daños de la fruta, así como los problemas de salud de los trabajadores se originan durante el procesos de cosecha, es de especial interés iniciar la automatización del cultivo de fresas en esta etapa, por lo cual la propuesta de un sistema de detección de la fruta madura representará un avance importante dentro del proceso de automatización de cosechas, ya que identificará de manera autónoma y en tiempo real, el fruto que posee las condiciones adecuadas de madurez para ser recolectado, disminuyendo de esta manera el tiempo de cosecha y obteniendo una mayor producción y un producto de mejor calidad.

CAPÍTULO II

MARCO REFERENCIAL

2.1 Marco Teórico

2.1.1 Generalidades de la fresa Albión

La fresa o llamada *Fragria x anassa Duch* en una especie de hortícola de reproducción vegetativa, es decir que se reproduce por eslones o por hijuelos, es originada por el cruce de dos especies del mismo género. (Soto, 2018).

Es resistente a la marchitez de Verticilium y la pudrición de la corona de phytophthora, contiene potentes antioxidantes y son altamente nutricionales, es muy rica en fibra, vitamina C y ácido fólico. Posee ciertas propiedades que ayudan a exfoliar la piel, aportan un extra de hidratación y protegen del sol. Se caracteriza por ser un fruto firme y de color rojo oscuro por dentro y por fuera.

Descripción botánica y morfológica. Esta planta se considera herbácea, es decir que sus hojas y órganos se forman en la parte leñosa de la corona y se le puede considerar como una planta resistente de vida corta. En la Figura 1 se presentan las partes de la planta de fresa.

Características fisicoquímicas. En sus propiedades tenemos su peso que varía entre 16,53 y 6,65g, y su concentración de azúcar está entre los 6,7 y 7,28 grados.

Características organolépticas. Las características se dan en los siguientes puntos:

- Contiene una serie de azucares y ácidos, por lo cual su sabor contiene un balance de azúcar y acidez.
- Su forma, por lo general son alargadas y cónicas.
- Tiene un brillo intenso y un color rojizo oscuro y uniforme, aunque puede ser más rosado o anaranjado dependiendo de la variedad.
- Su pulpa es de color blanco o rojizo de acuerdo con la variedad.
- Su textura es suave con firmeza moderada a alta (Flores & Mora, 2015).

Figura 1 : Partes esenciales de la planta fresa

Nota. Este grafico tomado de (Bonilla, 2010) , donde podemos detallar las partes esenciales de la planta de fresa.

Ciclo fenológico del cultivo. Esta planta se puede cultivar en cualquier época del año, pero se recomienda en época de lluvia ya que puede asegurar una adaptación del cultivo y garantizar el desarrollo inicial (Cortés, 2011). De este mismo modo la recolección se debe de dar en una época seca, por lo que el productor debe de tener una distribución de actividades como se lo puede ilustrar en la Figura 2.

Figura 2: Calendario de siembra, labores y cosecha del cultivo de la fresa en Colombia (Faura, 2010a)

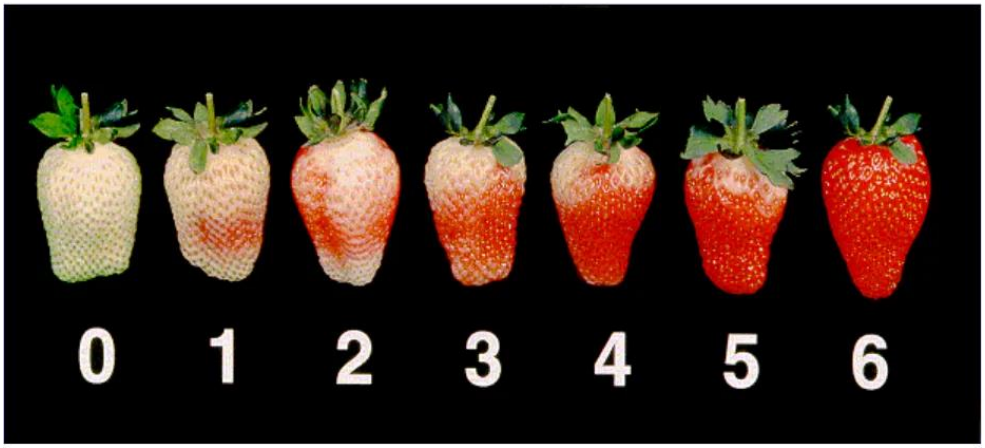
Nota. Calendario seccionado por épocas con sus respectivos procesos de cultivo.

Cosecha. Es necesaria la capacitación en los jornaleros para evitar las pérdidas de calidad ya que, por lo general, no se toman las precauciones necesarias causando daños mecánicos y aumentando las posibilidades de entrada de patógenos en la fruta (Faura, 2010).

Esta actividad se realiza en las primeras horas del día, y se recomienda el uso de guantes de látex para disminuir el daño. En la cosecha se debe de tener en cuenta los siguientes puntos:

- **Recolección:** El momento adecuado de recolección se debe dar de acuerdo con el propósito del cultivo. En cuanto a la calidad de coloración para la comercialización cada país tiene su norma, pero el estándar para el producto al nivel de Latinoamérica es de 1/3 del fruto coloreado como se observa en la Figura 3.

Figura 3: Índice de maduración de la fresa (Acuña, 2021)



Nota. Este índice es evaluado por expertos en la exportación de la fresa.

- **Índice de madurez para fresa:** Este parámetro tiene en cuenta el color del fruto para la determinación de la madurez.

- **Clasificación:** Es ideal no llevar más de 5 a 8 kg en cada canastilla (las cuales han de ser preferiblemente plásticas pues causan menos daños mecánicos y son de fácil limpieza).

- **Empaque:** Se utilizan empaques de plástico, tanto para consumo en fresco como industrial. Otro tipo de empaques pueden ser los de cartón comprimido o madera que se elaboran de acuerdo al peso contenido (empaques para 250, 500, 1000 y 2000 gr).

Postcosecha. Bajo patrones muy buenos de almacenamiento difícilmente logra mantenerse en buenas condiciones durante 5 días ya que su eficiencia en esta etapa es baja.

El manejo del fruto desde la cosecha debe ser muy cuidadoso ya que se pueden producir daños importantes, entre ellos la reducción de la firmeza, daño mecánico, pudriciones y fermentaciones.

La fruta debe ser empacada en el mismo momento de la cosecha y se ha de colocar en enfriamiento inmediatamente. La cadena de frio debe mantenerse hasta la entrega al consumidor final.

La primera selección que se le hace al fruto ya cosechado es retirar todo el producto contaminado por alguna plaga o enfermedad para no contaminar todo el lote de producción. La clasificación por color y tamaño se hace conforme a los requerimientos de calidad e índices de maduración permitida del lugar en el cual se hará la comercialización del producto.

Para mantener por más tiempo la calidad de la fresa se requiere entre 0 a 1°C y 90 a 95% de humedad relativa. Otra razón importante para realizar el enfriamiento es que los agentes patógenos siguen actuando aún después de cosechada y en esta etapa de la cadena ya no es posible aplicar ningún tipo de control químico (Flores & Mora, 2015).

2.1.2 Entornos de desarrollo

Anaconda Navigator. Software que permite crear ambientes de trabajos y realizar ciencia de datos, aprendizaje automático de Python / R, Anaconda fue construida por científicos de datos, para científicos de datos. Haciendo posible acceder a un conjunto de herramientas para trabajo con miles de paquetes y bibliotecas de código abierto.

Este permite hacer funcionar aplicaciones y administrar fácilmente distintos paquetes. Así, Anaconda Navigator puede buscar paquetes en Anaconda Cloud o en otros repositorios, y está disponible para ambientes Windows, macOS y Linux (Anaconda, 2019).

Google Colab. Proyecto de investigación de Google diseñado con el propósito de ayudar a difundir la educación y la investigación en el aprendizaje automático.

Colab permite conectarnos a un servidor de Jupyter que se encuentre configurado en forma local. Esto permite acceder al sistema de archivos local, con las versiones que instaladas en localmente.

Colab recrea cuadernos que permiten combinar código ejecutable y texto enriquecido en un mismo documento, además de imágenes, HTML, LaTeX y mucho más. Los cuadernos que se crean en Colab se almacenan en tu cuenta de Google Drive. Siendo posible compartir

cuadernos de Colab fácilmente con compañeros de trabajo, lo que les permite comentarlos o incluso editarlos (Colab Google, 2020).

Docker Hub. Docker Hub es la biblioteca y comunidad más grande del mundo para imágenes de contenedores, este recrea archivos Docker Image que ayuda a utilizar código en un contenedor de "Docker", con un tipo de imagen que se ejecuta como un grupo de instrucciones para generar una plantilla.

Docker ayuda a los desarrolladores a dar vida a sus ideas al conquistar la complejidad del desarrollo de aplicaciones. Simplificando y acelerando los flujos de trabajo de desarrollo con un canal de desarrollo integrado y mediante la consolidación de los componentes de la aplicación. Utilizados activamente por millones de desarrolladores en todo el mundo (Docker, 2020a).

JetPack 5.0 Developer Preview, es el Sistema operative Linux base con SDK de NVIDIA, que cuenta con un cargador de arranque, kernel de linux, entorno de escritorio Ubuntu y un conjunto completo de bibliotecas para la aceleración de computación Grafica GPU, gráficos, multimedia y visión artificial, a su vez incluye CUDA 11.4, que es básicamente los núcleos o procesadores paralelos que procesan los datos de entrada y salida, incluso tiene un modelo de programación que permite incrementos dramáticos en el rendimiento del computador (NVIDIA, 2021a).

PyTorch. PyTorch es una biblioteca de tensores optimizada para el aprendizaje profundo mediante GPU y CPU, se utiliza para aplicaciones donde se aplica visión artificial y procesamiento de imágenes, además, esta se basada en la biblioteca de Torch (Pytorch, 2021).

Las funciones descritas en esta documentación están clasificadas por estado de versión:

- **Estable:** estas características se mantendrán a largo plazo y, por lo general, no debería haber grandes limitaciones de rendimiento o lagunas en la documentación. También se espera mantener la compatibilidad con versiones anteriores (aunque pueden ocurrir cambios importantes y se notificará una versión antes).

- **Beta:** estas funciones están etiquetadas como Beta porque la API puede cambiar en función de los comentarios de los usuarios, porque el rendimiento debe mejorar o

porque la cobertura entre los operadores aún no está completa. Para las funciones Beta, se compromete a llevar la función a la clasificación Estable.

- **Prototipo:** estas características generalmente no están disponibles como parte de distribuciones binarias como PyPI o Conda, excepto a veces detrás de indicadores de tiempo de ejecución, y se encuentran en una etapa temprana para comentarios y pruebas (Pytorch, 2021).

SentiSight.ai – LabelImg. Este Software etiqueta imágenes para la etapa vital del entrenamiento de modelos de reconocimiento de imágenes de aprendizaje profundo. Esta herramienta de etiquetación de imágenes se ha diseñado para acelerar este proceso al ofrecer una gama de capacidades asistidas por IA que se pueden personalizar para cumplir con los requisitos del usuario. La anotación de imágenes es una parte importante del proceso para crear y entrenar sus propios modelos de reconocimiento de imágenes, que incluyen detección de objetos y clasificación de imágenes (Sentisight, 2020).

2.1.3 Inteligencia artificial

La IA es un conjunto de algoritmos desarrollados para brindar a una máquina las mismas capacidades que la de un humano. Permite a un robot tomar sus propias decisiones, interactuar con las personas y reconocer objetos. (Morena et al., 2019).

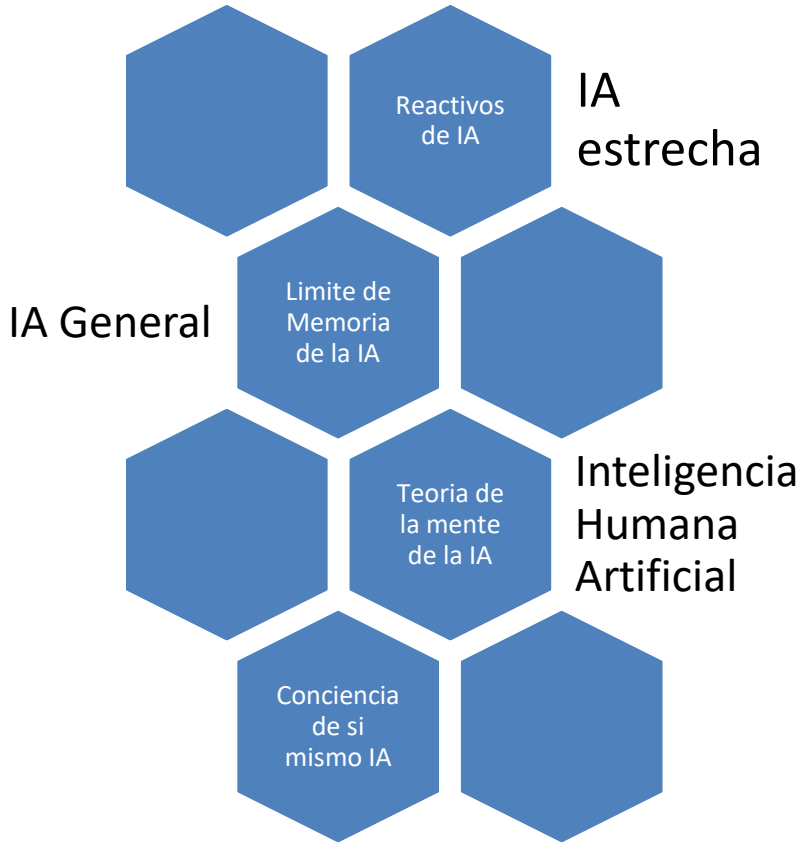
Existen muchos productos del mundo real que utilizan este tipo de tecnología, tales como:

- **Siri:** Asistente de voz de Apple, incluido en sus teléfonos y tabletas, que nos permite buscar datos al instante, enviar mensajes, verificar el clima y mucho más.
- **Netflix:** Servicio de cine y televisión en línea, que se ejecuta en un sistema de recomendación desarrolla con IA, para recomendar películas a los usuarios en función de su historial de visualización.
- **Spotify:** Servicio de música en línea que utiliza un sistema de para recomendar canciones a los usuarios, considerando las que ha escuchado previamente y el tipo de música agregada a su biblioteca.

- **Los autos autónomos de Tesla:** Construidos con IA para detectar obstáculos, personas e incluso señales de tráfico, garantizando que los pasajeros tengan un viaje seguro.

Tipos de Inteligencia Artificial. Existen diferentes tipos de sistemas de IA categorizados en función de su propósito principal (Figura 4). Dando paso hacia la construcción de sistemas más inteligentes.

Figura 4: Tipos de IA (Gollapudi, 2019a)



Nota. Segmentación de la inteligencia artificial según su aplicación.

IA reactiva. Este tipo de máquinas no tienen memoria y no utilizan información de experiencias pasadas. El contexto actual se percibe directamente tal como es, esto hace que la máquina se comporte de la misma manera cada vez que se encuentra con una situación, obteniendo un resultado confiable y consistente.

Las máquinas de IA de memoria limitada. Miran al pasado y lo usan como una representación preprogramada del mundo, para aplicarlo al conjunto de datos actual. Un ejemplo, son los automóviles autónomos.

Las máquinas de IA autoconscientes. Son una extensión de la teoría de la IA mental. Pueden configurar representaciones, lo que significa que se tiene máquinas que son conscientes dado un contexto.

- **La inteligencia artificial estrecha (ANI)** se trata de resolver un problema contra una solicitud dada con un rango estrecho de habilidades. Esto también se llama IA débil.
- **La inteligencia artificial general (AGI)** se conoce como IA fuerte y se refiere a una máquina que es tan capaz como los humanos.
- **La súper inteligencia artificial (ASI)** se trata de máquinas que pueden realizar tareas más allá de lo que los humanos son capaces de hacer.

La mayor parte de la IA existente hoy en día es ANI. AGI y ASI todavía se están desarrollando.

En la Machine Learning **(Aprendizaje Automático).** Es un campo de investigación en la intersección de la estadística, la inteligencia artificial y la informática, también se conoce como análisis predictivo o aprendizaje estadístico. La aplicación de métodos de aprendizaje automático se ha vuelto omnipresente en los últimos años en la vida cotidiana, ya que prácticamente se trata de extraer conocimiento de los datos.

Figura 5 representa las funciones y características básicas de un sistema de IA en el centro y los subcampos relacionados que respaldan la implementación de estas funciones (Gollapudi, 2019a).

Machine Learning (Aprendizaje Automático). Es un campo de investigación en la intersección de la estadística, la inteligencia artificial y la informática, también se conoce como análisis predictivo o aprendizaje estadístico. La aplicación de métodos de aprendizaje automático se ha vuelto omnipresente en los últimos años en la vida cotidiana, ya que prácticamente se trata de extraer conocimiento de los datos.

Figura 5: Inteligencia y visión artificial (Gollapudi, 2019)

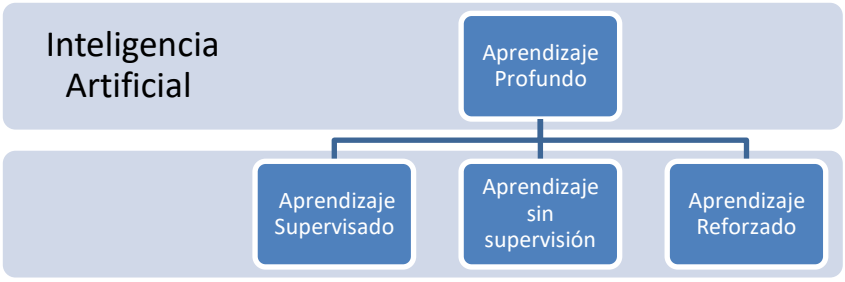
Nota. Aplicaciones con sus características definidas

Desde recomendaciones automáticas para ver películas, predicciones de gustos personales de comida, o qué productos comprar, opciones de emisoras de radio en línea personalizada, etiquetación de amigos en sus fotos en redes sociales, sitios web y dispositivos modernos tienen algoritmos de aprendizaje automático en su núcleo (Müller & Guido, 2017).

En otras palabras, es un mecanismo de búsqueda de patrones que filtra los detalles relevantes del entorno, para ir aprendiendo con el tiempo y hacerlo mejor utilizando su propia experiencia.

Los algoritmos de aprendizaje automático se construyen para generar inteligencia, con el objetivo de obtener un resultado en forma de una regla precisa en la máxima medida (Gollapudi, 2019b). Los subcampos de este tipo de aprendizaje se presentan en la Figura 6.

Figura 6: Subcampos del aprendizaje automático (Gollapudi, 2019b)



- **El aprendizaje supervisado:** Trabaja según una expectativa conocida, realizando un análisis a partir de los datos previamente definidos.

- **Aprendizaje no supervisado:** Es prácticamente cuando no existe un objetivo claro o un problema específico que resolver, es decir, que tiene como objetivo descifrar la estructura en los datos y luego identificar posibles atributos de salida.

- **Aprendizaje por refuerzo:** Es una metodología de aprendizaje que se centra en maximizar las recompensas del resultado.

2.1.4 Neural Networks (Red Neuronal).

Una Red Neuronal se la puede definir como un modelo matemático concerniente al manejo y procesamiento de la información, siendo una buena manera descriptiva de un algoritmo de aprendizaje autónomo, dado que el procesamiento de la información ocurre en su forma más simple, sobre elementos llamados neuronas, y estas están conectadas e intercambian señales entre ellas a través de enlaces de conexión.

Estas conexiones crean vínculos, generando un estado interno determinado por las conexiones entrantes, dando una función de activación diferente, que se calcula sobre su estado, determinando su señal de salida.

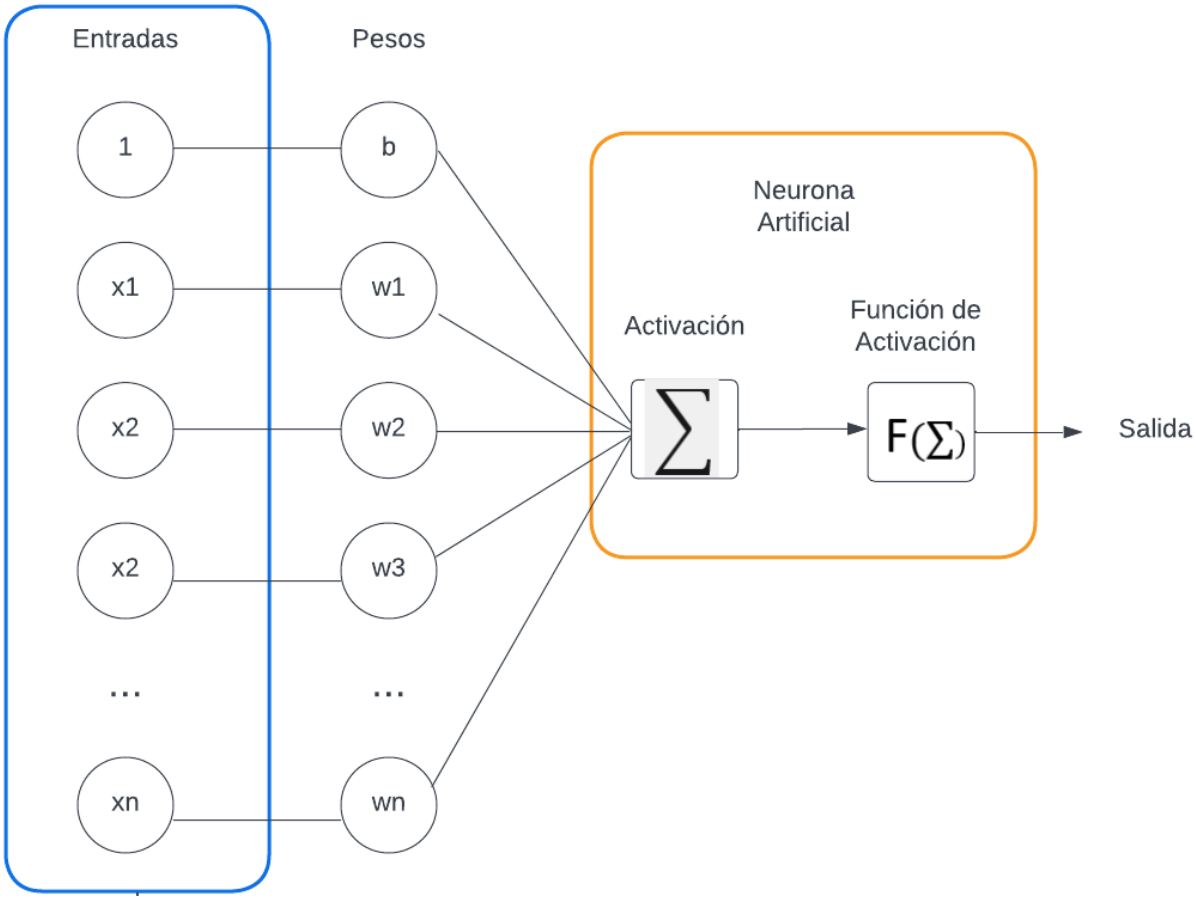
Las características principales para una red Neuronal, se destaca por la arquitectura a la red neuronal, que describe el conjunto de conexiones, determinando si es multi o de una sola capa, definiendo el feedforward que es la topología del arreglo de neuronas y sus interconexiones.

Otra característica esencial es el aprendizaje, lo que en otras palabras se puede definir como el entrenamiento; la forma más común de entrenar una red neuronal es con el descenso de gradiente y la retro – propagación; podemos definir a la gradiente, como un cálculo que permite como ajustar los parámetros de red de tal forma que se minimice su desviación a la salida (Vasilev, 2019a). En la Figura 7 e presenta una representación de una red neuronal que representa dicha función matemática estructurada con sus entradas, pesos, capas, neurona artificial y su salida.

La Neurona se puede representar matemáticamente como se indica en la siguiente ecuación (8):

$$y = f\left(\sum_{i}^{n} xi.wi + b\right) \quad (1)$$

Figura 7: Representación gráfica de una Red Neuronal Artificial (Vasilev, 2019).



Nota: Una neurona es una función matemática que toma uno o más valores de entrada y emite un solo valor numérico.

En donde se calcula la suma ponderada de las entradas xi que son valores numéricos que representa los datos de entrada y los pesos wi que representa valores numéricos que presentan la fuerza de las entradas o la fuerza de las conexiones entre las neuronas.

Una vez realizada la primera operación, se toma el resultado como entrada a la función de activación f, conocida también como función de transferencia, cumpliendo con el requisito de ser no lineales; no obstante, el valor de activación definido anteriormente puede interpretarse como el producto de punto entre el vector w y el vector x. La salida de la neurona se convierte entonces en y = wx + b, que es la ecuación lineal. Esto muestra que, en el espacio de entrada unidimensional, la neurona define una línea.

Para finalizar, hay que tener en cuenta que en el Machine learning el perceptrón solo funciona con clases linealmente separables y se pueden definir en un Hiperplano organizando las neuronas en una red neuronal.

Redes Neuronales Convolucionales. Es un tipo de red neuronal artificial similar a las neuronas biológicas en la corteza visual primaria ya que corresponden a campos receptivos.

Fundamentos de las redes neuronales convolucionales

Núcleos y filtros:

Convolution: Función encargada de filtrar los valores que pueden ayudar a aumentar la nitidez, difuminar, detectar ejes y realces basados en kernel de una imagen, dentro de los valores de cada píxel. Además, permite procesar por partes los pixeles de una imagen, para generar una nueva abstracción de la imagen, con una profundidad mayor.

Pooling: Filtro encargado de reducir el tamaño de la imagen, con el fin de agrupar varios píxeles en uno, y así reducir la carga computacional necesaria en la próxima capa de una imagen más reducida.

Se emplean dos métodos de Pooling, los cuales son:

Max Pooling: Filtro que recorre bloques de pixeles en la imagen, agrupándolos, y tomando el valor más alto sobre el bloque de imagen.

Average Pooling: Recorre bloques de pixeles para obtener el promedio de todos los pixeles en un rango determinado, para después generarlo como un nuevo píxel.

Deep Learning. Hoy en día el aprendizaje profundo, pue ser la definición de una realidad a través de las representaciones de sus rasgos. Una red neuronal profunda no simplemente reconoce lo que hace que un gato sea un gato, o una ardilla una ardilla, sino que entiende qué características están presentes en un gato y una ardilla respectivamente. Aprende a diseñar un gato o una ardilla utilizando esas características (Vasilev, 2019a).

Una red profunda que aprende representaciones básicas de su salida puede hacer clasificaciones utilizando las suposiciones que ha hecho. Por ejemplo, si no hay cola peluda, probablemente no será una ardilla, sino más bien un gato. De esta manera, la cantidad de información que aprende la red es mucho más completa y robusta, y la parte más emocionante es que las redes neuronales profundas aprenden a hacerlo automáticamente.

Para comprender la importancia del Deep Learning y las redes convolucionales, se puede apreciar en el trabajo de Krizheysky, Sutskever y Hilton, titulado "*ImageNet Classification with Deep Convolutional Neural Networks"* mencionan claramente la importancia del número de capas ocultas presentes en las redes profundas.

En el trabajo anteriormente mencionado es notable que el rendimiento de la red neuronal disminuye, cuando se elimina una capa convolucional dando como resultado la perdida de aproximada el 2%, evidenciando esto es deseable que la profundidad es realmente importante para obtener buenos resultados (Krizhevsky et al., 2014).

2.1.5 Sistemas de visión artificial

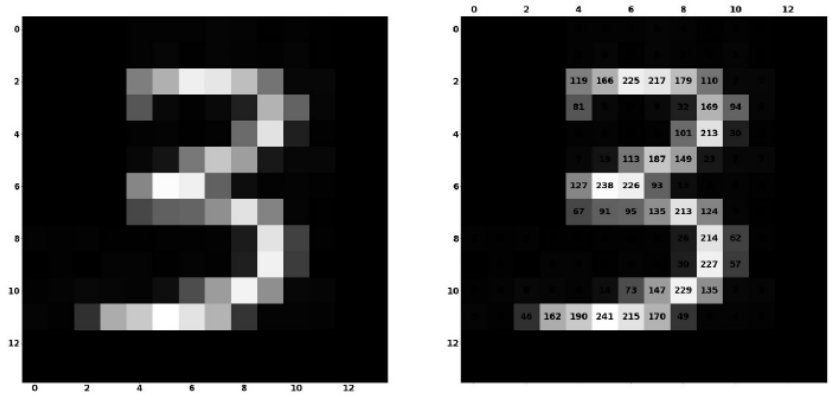
La sección de IA encargada de imágenes se llama visión por computadora, y es un campo científico interdisciplinario que trata de imitar los ojos humanos, obtenido mayor nivel de comprensión de una imagen específica al realizar tareas automatizadas y emplear algoritmos.

La visión artificial o visión por computadora es un conjunto de métodos utilizados para adquirir, analizar, procesar imágenes y transformarlas en información que puede ser valiosa para una computadora. Transformando la información recopilada en datos numéricos, para su posterior uso en el computador (Morena et al., 2019).

Algoritmos básicos en computadora

- **Terminología de la imagen:** Una computadora entiende una imagen como un conjunto de números agrupados (matriz bidimensional que contiene valores de 0 a 255 y en imágenes en escala de grises, 0 es para negro y 255 para blanco) que representan los valores de píxeles de una imagen, como se muestra en la Figura 8.

Figura 8: Representación de imágenes sin y con valores de píxeles (Morena et al., 2019)



En la Figura 8, en el lado izquierdo, el número 3 se muestra en una resolución baja. En el lado derecho, se muestra la imagen junto con el valor de cada píxel. A medida que este valor aumenta, se muestra un color más brillante y, si el valor disminuye, el color se oscurece.

Las imágenes coloreadas o imágenes RGB tienen tres capas de matrices bidimensionales aplicadas juntas. Cada capara representa un color y al juntarlas todas, forma la imagen coloreada **(**Morena *et al.,* 2019).

OpenCV. Es una biblioteca de código abierto que incluye algoritmos de visión por computadora. El documento describe la llamada API OpenCV 2.x, que es esencialmente una API C ++, a diferencia de la API OpenCV 1.x basada en C (la API C está obsoleta y no se ha probado con el compilador "C" desde las versiones de OpenCV 2.4)

- OpenCV tiene una estructura modular, lo que significa que el paquete incluye varias bibliotecas compartidas o estáticas. Dispone de los siguientes módulos:
- Funcionalidad principal (núcleo): módulo compacto que define las estructuras de datos básicas, incluida la densa matriz multidimensional Mat y funciones básicas utilizadas por los demás módulos.

- Procesamiento de imágenes (imgproc): módulo de procesamiento de imágenes que incluye filtrado de imágenes lineales y no lineales, transformaciones geométricas de imágenes, conversión de espacio de color, histogramas, etc.
- Análisis de video (video): módulo de análisis de video con algoritmos de estimación de movimiento, sustracción de fondo y seguimiento de objetos.
- Calibración de cámara y reconstrucción 3D (calib3d): algoritmos básicos de geometría de múltiples vistas, calibración de cámara única y estéreo, estimación de pose de objeto, algoritmos de correspondencia estéreo y elementos de reconstrucción 3D.
- Marco de características 2D (features2d): detectores de características destacadas, descriptores y comparadores de descriptores.
- Detección de objetos (objdetect): detección de objetos e instancias de las clases predefinidas (caras, ojos, personas, automóviles, etc.).
- GUI de alto nivel (highgui): interfaz fácil de usar para funciones de IU simples.
- E / S de video (videoio): interfaz para captura de video y códecs de video.
- Algunos otros módulos auxiliares, como los contenedores de prueba de FLANN y Google, los enlaces de Python y otros.

2.1.6 Modelos de una red de detección

Los modelos de detección de objetos tienen un único archivo comprimido, que contiene una estructura de directorios y archivos. En este se soportan algoritmos de detección de objeto como los siguientes:

ResNet-50 (Deep Residual Networks)

ResNet 50 es un modelo de clasificación de imágenes previamente entrenado en el conjunto de datos ImageNet. Esta es la implementación de PyTorch basada en la arquitectura descrita en el documento “Aprendizaje profundo residual para el reconocimiento de imágenes” en el paquete TorchVision. La entrada del modelo es un blob que consta de una sola imagen de en orden.1, 3, 224, 224RGB.

El resultado del modelo es un clasificador de objetos típico para las 1000 clasificaciones diferentes que coinciden con las de la base de datos ImageNet.

- **Especificación**

Tabla 1: Especificaciones caracterizadas del ResNet-50

Métrico	Valor
Escribe	Clasificación
GFLOP	10.8148
MParams	27.4493
Marco de origen	PyTorch *

Fuente: Recuperado de la página oficial de OPENVINO de la empresa Intel (Intel Corporation, 2022)

- **Precisión**

Tabla 2: Referencias de precisión del ResNet-50

Métrico	Valor
Top 1	81,11%
Top 5	95,36%

Fuente: Recuperado de la página oficial de OPENVINO de la empresa Intel (Intel Corporation, 2022)

Ssd_mobilenet_v2_coco

El ssd_mobilenet_v2_cocomodelo es una red de detección de casilla múltiple de disparo único (SSD) destinada a realizar la detección de objetos. El modelo ha sido entrenado a partir del conjunto de datos de imágenes de objetos comunes en contexto (COCO). La entrada del modelo es un blob que consta de una sola imagen de en orden.1, 3, 300, 300RGB

La salida del modelo es un vector típico que contiene los datos del objeto rastreado, como se describió anteriormente. Tenga en cuenta que los class_iddatos ahora son significativos y deben usarse para determinar la clasificación de cualquier objeto detectado.

- **Especificación**

Tabla 3: Especificaciones caracterizadas del ssd_mobilnet_v2_coco

Métrico	Valor
Tipo	Detección
GFLOP	3.775
MParams	16.818
Marco de origen	TensorFlow *

Fuente: Recuperado de la página oficial de OPENVINO de la empresa Intel (Intel Corporation, 2022)

- **Precisión**

Tabla 4: Referencias de precisión del ssd_mobilnet_v2_coco

Métrico	Valor
coco_precision	24,9452%

Fuente: Recuperado de la pagina oficial de OPENVINO de la empresa Intel (Intel Corporation, 2022)

Tiny YOLO V3

YOLO v3 Tiny es un modelo de detección de objetos en tiempo real implementado con Keras * desde este repositorio y convertido al marco TensorFlow *. Este modelo se entrenó previamente en un conjunto de datos de objetos comunes en contexto (COCO) con 80 clases.

- **Especificación**

Tabla 5: Especificaciones caracterizadas del Tiny YOLO V3

Métrico	Valor
Escribe	Detección
GFLOP	5.582
MParams	8.848
Marco de origen	Keras *

Fuente: Recuperado de la pagina oficial de OPENVINO de la empresa Intel (Intel Corporation, 2022)

- **Precisión**

Tabla 6: Métricas de precisión obtenidas en el conjunto de datos de validación de Common Objects in Context (COCO) para el modelo convertido

Métrico	Valor
mapa	**35,9%**
MAPA DE COCO	**39,7%**

Fuente: Recuperado de la pagina oficial de OPENVINO de la empresa Intel (Intel Corporation, 2022)

2.2 Marco Legal

En este punto se analiza un conjunto de leyes, normas y reglamentos que le dan fundamento a esta investigación.

2.2.1 Artículos de constitución

Art. 276.- El régimen de desarrollo tendrá los siguientes objetivos:

1. Mejorar la calidad y esperanza de vida, y aumentar las capacidades y potencialidades de la población en el marco de los principios y derechos que establece la Constitución.

2. Construir un sistema económico, justo, democrático, productivo, solidario y sostenible basado en la distribución igualitaria de los beneficios del desarrollo, de los medios de producción y en la generación de trabajo digno y estable.

3. Fomentar la participación y el control social, con reconocimiento de las diversas identidades y promoción de su representación equitativa, en todas las fases de la gestión del poder público.

4. Recuperar y conservar la naturaleza y mantener un ambiente sano y sustentable que garantice a las personas y colectividades el acceso equitativo, permanente y de calidad al agua, aire y suelo, y a los beneficios de los recursos del subsuelo y del patrimonio natural.

5. Garantizar la soberanía nacional, promover la integración latinoamericana e impulsar una inserción estratégica en el contexto internacional, que contribuya a la paz y a un sistema democrático y equitativo mundial.

6. Promover un ordenamiento territorial equilibrado y equitativo que integre y articule las actividades socioculturales, administrativas, económicas y de gestión, y que coadyuve a la unidad del Estado.

7. Proteger y promover la diversidad cultural y respetar sus espacios de reproducción e intercambio; recuperar, preservar y acrecentar la memoria social y el patrimonio cultural.

Art. 277.- Para la consecución del buen vivir, serán deberes generales del Estado:

1. Garantizar los derechos de las personas, las colectividades y la naturaleza.

2. Dirigir, planificar y regular el proceso de desarrollo.

3. Generar y ejecutar las políticas públicas, y controlar y sancionar su incumplimiento.

4. Producir bienes, crear y mantener infraestructura y proveer servicios públicos.

5. Impulsar el desarrollo de las actividades económicas mediante un orden jurídico e instituciones políticas que las promuevan, fomenten y defiendan mediante el cumplimiento de la Constitución y la ley.

6. Promover e impulsar la ciencia, la tecnología, las artes, los saberes ancestrales y en general las actividades de la iniciativa creativa comunitaria, asociativa, cooperativa y privada.

Art. 281.- La soberanía alimentaria constituye un objetivo estratégico y una obligación del Estado para garantizar que las personas, comunidades, pueblos y nacionalidades alcancen la autosuficiencia de alimentos sanos y culturalmente apropiado de forma permanente. Para ello, será responsabilidad del Estado:

1. Impulsar la producción, transformación agroalimentaria y pesquera de las pequeñas y medianas unidades de producción, comunitarias y de la economía social y solidaria.

2. Adoptar políticas fiscales, tributarias y arancelarias que protejan al sector agroalimentario y pesquero nacional, para evitar la dependencia de importaciones de alimentos.

3. Fortalecer la diversificación y la introducción de tecnologías ecológicas y orgánicas en la producción agropecuaria.

4. Promover políticas redistributivas que permitan el acceso del campesinado a la tierra, al agua y otros recursos productivos.

5. Establecer mecanismos preferenciales de financiamiento para los pequeños y medianos productores y productoras, facilitándoles la adquisición de medios de producción.

6. Promover la preservación y recuperación de la agro biodiversidad y de los saberes ancestrales vinculados a ella; así como el uso, la conservación e intercambio libre de semillas.

7. Precautelar que los animales destinados a la alimentación humana estén sanos y sean criados en un entorno saludable.

8. Asegurar el desarrollo de la investigación científica y de la innovación tecnológica apropiadas para garantizar la soberanía alimentaria.

9. Regular bajo normas de bioseguridad el uso y desarrollo de biotecnología, así como su experimentación, uso y comercialización.

10. Fortalecer el desarrollo de organizaciones y redes de productores y de consumidores, así como las de comercialización y distribución de alimentos que promueva la equidad entre espacios rurales y urbanos.

11. Generar sistemas justos y solidarios de distribución y comercialización de alimentos. Impedir prácticas monopólicas y cualquier tipo de especulación con productos alimenticios.

12. Dotar de alimentos a las poblaciones víctimas de desastres naturales o antrópicos que pongan en riesgo el acceso a la alimentación. Los alimentos recibidos de ayuda internacional no deberán afectar la salud ni el futuro de la producción de alimentos producidos localmente.

13. Prevenir y proteger a la población del consumo de alimentos contaminados o que pongan en riesgo su salud o que la ciencia tenga incertidumbre sobre sus efectos.

14. Adquirir alimentos y materias primas para programas sociales y alimenticios, prioritariamente a redes asociativas de pequeños productores y productoras.

En el Art. 256 de la constitución del ecuador detalla la obligación del régimen de desarrollo para mejorar la calidad de vida, teniendo de la mano el desarrollo económico, cultural y productivo en ello, consecuentemente en el Art. 277 inciso 6, da a conocer como deberes de estado el apoyo a la innovación tecnológica entre otros factores esenciales para el buen vivir, y por último el Art. 281 donde el estado se responsabiliza en impulsar la producción y fortalecer la diversificación e introducción de tecnologías ecológicas para el desarrollo de la producción agropecuaria.

2.2.2 Convenios nacionales

Acuerdo MDT-2015-0233 expídanse las normas que regulan las relaciones de trabajo especiales en el sector agropecuario

Que, el artículo 33 de la Constitución de la República del Ecuador establece que el trabajo es un derecho, un deber social y un derecho económico, fuente de realización personal y base de la economía, siendo el Estado el que garantizará a las personas trabajadoras el pleno respeto a su dignidad, una vida decorosa, remuneraciones y retribuciones justas y el desempeño de un trabajo saludable y libremente escogido o aceptado;

Que, de acuerdo con lo establecido en el numeral 6 del artículo 284 de la Constitución de la República del Ecuador, la política económica del Estado ecuatoriano tiene el objetivo de impulsar el pleno empleo y valorar todas las formas de trabajo, con respeto a los derechos laborales;

Que, el artículo 325 de la Constitución de la República del Ecuador establece que el Estado garantizará el derecho al trabajo. Se reconocen todas las modalidades de trabajo, en relación de dependencia o autónomas, con inclusión de labores de auto sustento y cuidado humano; y como actores sociales productivos a todos los trabajadores;

Ley de Fomento y Desarrollo Agropecuario

Art. 4 La Política de Investigación Agropecuaria será determinada por el Ministerio de Agricultura y Ganadería y ejecutada por el Instituto Nacional de Investigaciones Agropecuarias, observando las siguientes prioridades:

1. Productos alimenticios básicos de alto contenido nutritivo;

2. Productos destinados a la exportación;

3. Productos destinados a la sustitución de importaciones, y,

4. Materia Prima para la industria nacional.

Art. 5 Titulo III

De la Orientación y de los Beneficios de la Ley Artículos 4 a 53

*2.2.3 **CAPÍTULO I-** De la Investigación Agropecuaria Artículos 4 a 8*

La investigación agropecuaria se orientará a elevar la productividad de los recursos humanos y naturales mediante la generación y adopción de tecnologías de fácil difusión y aplicación a fin de incrementar la producción de los renglones señalados en el artículo anterior.

El Gobierno Nacional atenderá en forma prioritaria la asignación de recursos destinados a la investigación agropecuaria que realicen el Instituto Nacional de Investigaciones Agropecuarias y otras entidades del sector público.

*2.2.4 **CAPÍTULO VII-** De la Sanidad Agropecuaria*

Artículos 43 a 45

Es obligación de los productores velar por la salud de sus animales y la sanidad de sus plantas, así como participar en las campañas de sanidad emprendidas por el Gobierno.

Art. 44 El Ministerio de Agricultura y Ganadería establecerá programas y aplicará medidas de prevención de enfermedades y plagas que afecten a la vida vegetal y animal del país; y, en caso de presentarse éstas, organizará de inmediato campañas de erradicación.

Art. 45 Las campañas de sanidad serán financiadas con fondos fiscales y con la participación de los productores beneficiados, para lo cual el Ministerio de Agricultura fijará las tazas de servicio correspondientes.

2.3 Convenios internacionales

2.3.1 La Convención Internacional de Protección Fitosanitaria

La Convención Internacional de Protección Fitosanitaria (CIPF) establece normas (llamadas NIMF) para el transporte inocuo y seguro de plantas y productos vegetales, con la finalidad de prevenir la extensión internacional de plagas y enfermedades vegetales. El cumplimiento de las obligaciones de la CIPF y las NIMF es esencial para que los países puedan comerciar a escala internacional y para la seguridad alimentaria. Estas normas son importantes ya que permiten proteger a los consumidores, los productores y el medio ambiente de los países de los riesgos que representan las plagas introducidas, y porque ayudan a los exportadores a demostrar que sus productos son inocuos. El Portal Fitosanitario Internacional contiene toda la

documentación de relevancia para el programa de trabajo de la CIPF, incluidas las normas aprobadas y propuestas, y supervisa la aplicación de la Convención en los países. La FAO, junto con la Secretaría de la CIPF, ayuda a los países en desarrollo a mejorar la capacidad de sus servicios fitosanitarios para que puedan aplicar las normas acordadas y los procedimientos de la CIPF.

2.3.2 El Convenio de Rotterdam

El Convenio de Rotterdam, hospedado conjuntamente por AGP y la División de Productos Químicos del PNUMA, se ocupa del comercio internacional de ciertos productos químicos peligrosos con la finalidad de proteger la salud humana y el medio ambiente. Los plaguicidas representan una categoría principal de los productos químicos peligrosos objeto de comercio. El Convenio también contribuye al uso de dichos productos químicos de manera que respeten el medio ambiente y asegura el intercambio de información sobre sus características mediante el establecimiento de un proceso de toma de decisiones sobre las exportaciones y las importaciones nacionales y la difusión de estas decisiones a las partes.

2.3.3 La Reunión Conjunta sobre Residuos de Plaguicidas

AGP colabora estrechamente con la Organización Mundial de la Salud (OMS) para reducir los riesgos derivados del uso de plaguicidas y, particularmente, en el seno del principal órgano asesor de la Comisión del Codex Alimentarius: la Reunión Conjunta FAO/OMS sobre Residuos de Plaguicidas, con el objeto de recomendar los niveles máximos de residuos en alimentos y piensos y de proporcionar orientaciones sobre el establecimiento de límites máximos para residuos, y la Reunión Conjunta FAO/OMS sobre las Especificaciones de Plaguicidas, con el objeto de desarrollar parámetros de calidad de los plaguicidas para fines reglamentarios y comerciales.

AGP participa principalmente en la Reunión Conjunta FAO/OMS sobre Residuos de Plaguicidas en la revisión de los patrones de utilización de plaguicidas, datos sobre las características químicas y la composición de los plaguicidas, el destino ambiental, el metabolismo en el ganado y los cultivos, los métodos de análisis de los residuos de plaguicidas y los estudios de elaboración. Los expertos de la OMS se ocupan principalmente de la evaluación de los datos toxicológicos asociados.

2.3.4 Las comisiones de lucha contra la langosta de la FAO

Las comisiones regionales de lucha contra la langosta de la FAO representan una red de colaboración internacional fundamental para el intercambio de datos sobre los brotes presentes y potenciales entre países vecinos. La facilitación de las comisiones corre a cargo de AGP, específicamente el programa sobre plagas y enfermedades vegetales transfronterizas del Sistema de prevención de emergencia (EMPRES).

2.3.5 La Comisión de Recursos Genéticos para la Alimentación y la Agricultura

La Comisión de Recursos Genéticos para la Alimentación y la Agricultura (CRGAA) y su Grupo de trabajo técnico intergubernamental sobre los recursos fitogenéticos para la alimentación y la agricultura, y la Secretaría del Tratado Internacional sobre los Recursos Fitogenéticos para la Alimentación y la Agricultura son órganos importantes que promueven la conservación y el uso sostenible de los recursos genéticos. La función de AGP consiste en proporcionar sus conocimientos técnicos especializados sobre aspectos de la producción vegetal y prestar apoyo a procesos internacionales.

CAPÍTULO III

MARCO METODOLÓGICO

3.1 Descripción del área de estudio / Grupo de estudio

El estudio se llevará a cabo en un cultivo de fresas perteneciente a la microempresa “LA SELECTA” perteneciente a la Sra. María Lucrecia Guerra Andrango, dicho cultivo está ubicado en el sector de Santa Isabel de Pilascacho, comunidad de la parroquia de Andrade Marín, que pertenece a cantón Antonio Ante provincia de Imbabura.

La señora Guerra es uno de los proveedores de fresas de los mercados adyacentes de Ibarra y Cayambe. En la Figura 9 se indica el destino ubicado a 17,1 Km de la Universidad Técnica del Norte, a un tiempo de 31 min y fue tomada de Google Maps en la Web, por el autor. También se muestra en dicha imagen la ubicación de la empresa donde se realizará el estudio.

Figura 9: Localización de la microempresa la Selecta. Esta imagen indica el destino ubicado a 17,1 Km de la Universidad Técnica del Norte, a un tiempo de 31 min.

3.2 Enfoque de investigación / Tipo de investigación

En este trabajo se aplicó un método de inducción – deducción, ya que se basa en lo lógico, sus razonamientos son estructurados y empleados en las siguientes partes: observación,

deducción y experimentación, teniendo en cuenta que se genera un enfoque ingenieril, ya que se debe pensar en términos analíticos y objetivos que permitan enfocar los problemas de manera metódica ofreciendo la solución de una problemática, que sea concretada a través de las prácticas de los conocimientos de ingeniería, determinando sus restricciones, funcionalidades, criterios de diseño y la optimización de soluciones.

Se optó por el tipo de investigación científica, ya que constituye una estrategia de observación y reflexión sistemáticamente sobre realidades, obteniendo resultados para la base del desarrollo de la creación científica, por lo tanto, se inició desde la identificación del problema y su análisis, haciendo trabajo de campo y palpando la realidad, y tomando sus requerimientos.

En este sentido se procedió a la búsqueda de posibles soluciones, utilizando el trabajo documental que es un tipo de investigación cualitativa que se encarga de recopilar y seleccionar información a través de la lectura de documentos, libros, revistas, grabaciones, filmaciones, periódicos, bibliografías, entre otras fuentes de información (Fuentes-Doria et al., 2020).

Luego de realizar la revisión del estado del arte correspondiente, se procedió a la evaluación de las diferentes alternativas y la selección de la solución óptima.

3.3 Procedimientos

Fase 1: Investigación:

Se realizó un proceso de revisión de literatura y una investigación de campo, para la recopilación de información con respecto al proceso de cultivo y cosecha de la fresa, incluyendo las propiedades del producto, las malezas y enfermedades. Dando a conocer los puntos característicos visibles para el algoritmo de visión artificial, adjuntando los requerimientos del sistema.

- **Actividad 1:** "Revisión de literatura": Se procedió a la investigación de diferentes conceptos relacionados con el cultivo de fresa Albión, como sus características morfológicas y procesos de cosecha.
- **Actividad 2:** "Trabajo de campo en el proceso de cultivo y cosecha de la fresa, para recopilación de información": Se hizo trabajo de campo en el cultivo de estudio, para

recopilar información de dicho producto, de acuerdo con la experiencia vivida de los trabajadores, utilizando rubricas o entrevistas.

- **Actividad 3:** "Examinar los parámetros de calidad de producto que maneja la empresa": Fueron obtenidos los parámetros de cosecha en el cultivo para la categorización de la fresa.

Fase 2: Diseño del algoritmo

Se examinaron los factores a emplear en algoritmo para la categorización del estado de la fresa, que se trabaja en la tarjeta informática de selección, junto a las librerías correspondientes y con los programas adyacentes que se debe usar para la selección del algoritmo.

- **Actividad 4:** "Recopilación de fotos para la creación del Dataset": Se la recolectaron las imágenes o fotos de fresas maduras en buenas condiciones, para la creación de la data sheet, con la cual se entrenó del algoritmo.

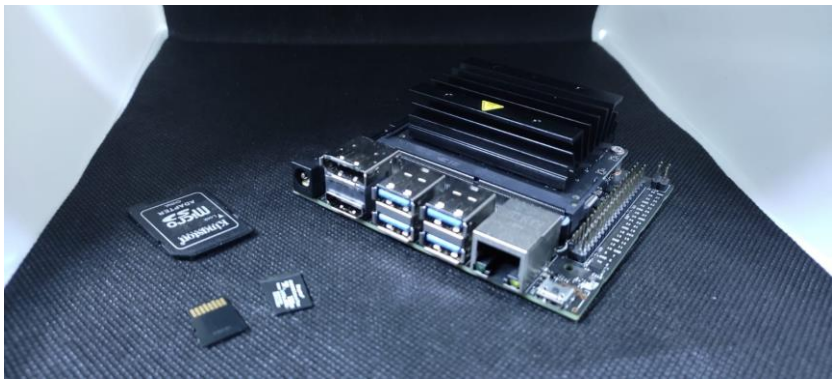
- **Actividad 5:** "Instalación de software": Se instaló el sistema operativo que utiliza la JETSON nano y los paquetes, librerías y programas adjuntos que sirvieron para el desarrollo de visión artificial.

En ello se recomienda instalar los siguientes programas:

a) **Instalación JetPack 4.0**: para la instalación se puede guiar en la página oficial de NVIDIA, 2021 , donde encontramos una guía rápida de instalación.

   Una vez finalizado se procede a colocar esta memoria en la tarjeta JETSON nano de la Figura 10, para encenderla conectando los periféricos cotidianos, como son un mouse, teclado y monitor, y proceder a instalar los repositorios.

Figura 10: Representación de la tarjeta JETSON con las tarjetas de memoria necesarias



b) Instalación de Dóker: para descargar este bloque de contenedores, es necesario descargarnos y seguir las instrucciones dadas de las pagina oficial de Docker, 2020b, la instalación ocupa el l4t-pytorch que es el contenedor base, ya que incluye un soporte para la transferencia de aprendizaje/entrenamiento. Una vez realizado esto lanzamos un contenedor, ejecutándolo con los siguientes comandos de la Figura 11.

Figura 11: Comandos necesarios para inicializar el modelo y descargar recursos

```
$ git clon --recursivo https://github.com/dusty-nv/jetson-inference
$ cd jetson-inferencia
$ docker/ejecutar.sh
```

En este caso es recomendable usar el Docker/run.sh, que es simplemente un script para ejecutar el contenedor, a la vez extrayendo la etiqueta de DockerHub en función del JetPack-L4T, montando todas las carpetas, incluso los modelos DNN a utilizar con las siguientes rutas:

- **JETSON-inference/data** (almacena los modelos de red, motores TensorRT serializados e imágenes de prueba)
- **JETSON-inference / python/ training / classification / data** (almacena conjuntos de datos de entrenamiento de clasificación)
- **JETSON-inference / python / training / classification / models** (modelos de clasificación de tiendas entrenados por PyTorch)
- **JETSON-inference / python / training / detection / ssd / data** (almacena conjuntos de datos de entrenamiento de detección)

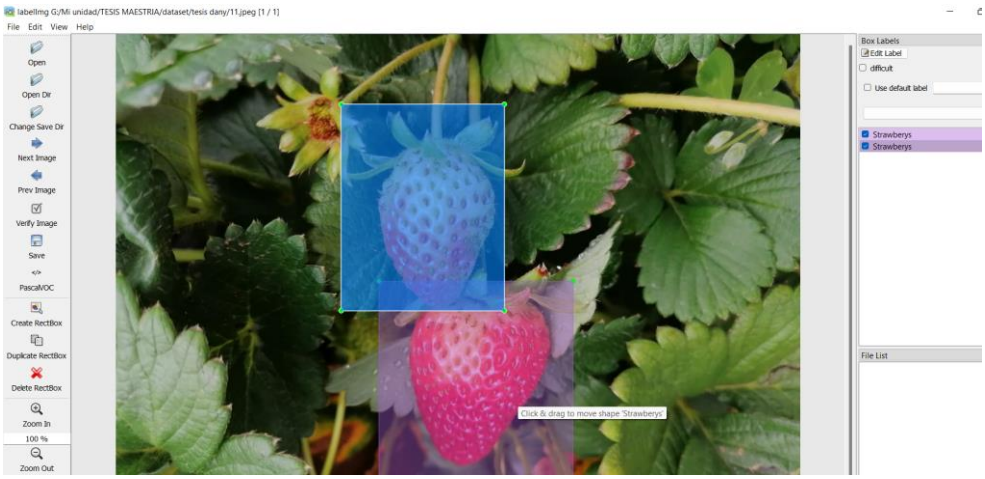
- **JETSON-inference / python / training / detection / ssd / models** (almacenas modelos de detección entrenados por PyTorch)

Fase 3: Preparación de la Data para Diseño del algoritmo de visión artificial

Teniendo en cuenta la información que ha sido recopilada sobre la fruta a seleccionar, como sus características, enfermedades etc., se obtuvieron imágenes que fueron utilizadas para definir los parámetros de selección.

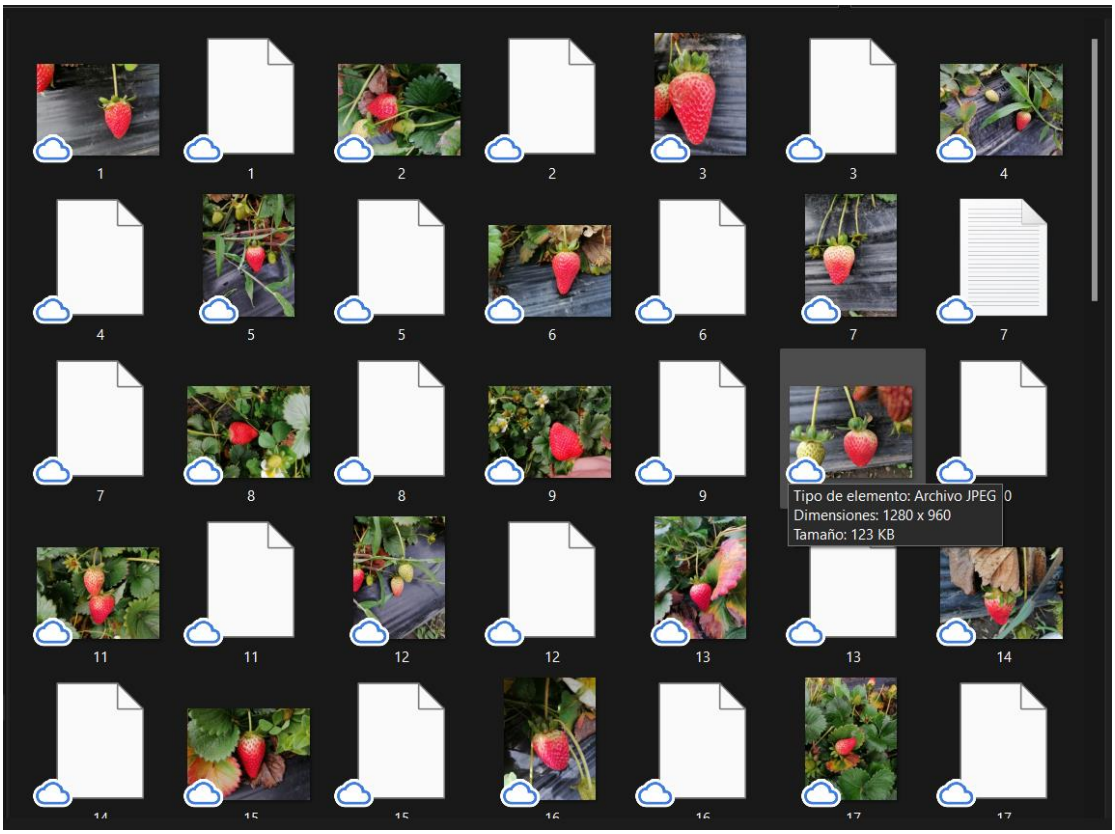
- **Actividad 6:** "Etiquetado y transformación con la compresión de datos": Para realizar esta actividad es necesario tener a la mano un software que permita etiquetar las imágenes obtenidas, y crear una marca en un set de entrenamiento de las coordenadas de los objetos que se deben detectar, obteniendo un archivo de coordenadas en un formato XML (Figura 12).

Figura 12: Proceso de etiquetado de fresas para entrenar el modelo



Este proceso puede tardar un tiempo ya que analiza una cantidad considerable de imágenes, teniendo en cuenta que entre más imágenes se utilicen, mejor será la precisión del algoritmo. Una vez hecho, se debe crear un archivo global con todas coordenadas de todas las imágenes, para convertir los datos de XML a un archivo Pasca VOC o CSV (valores separados por comas) o un formato de compresión, obteniendo una recopilación de todos los XML. Finalmente se transforma a un formato que permita el entrenamiento del algoritmo (Figura 13).

Figura 13: Carpeta contenedora de las imágenes con sus respectivos archivos XML



- **Actividad 7:** "Pre-entrenamiento: Antes de empezar, se descarga el modelo "SSD-Mobilenet-v1" predefinido y usaremos las redes "ResNet -18" que son redes neuronales profundas con un alto nivel y son escalarmente complicadas de entrenar, pero que en el modelo escogido facilita su entrenamiento reformulando, en otras palabras, se utilizan las capas como funciones residuales de aprendizaje con referencia a las entradas de las capas externas, lo cual brinda eficiencia, que sean mucho más fáciles de optimizar y además se obtiene una mayor precisión a partir de un punto ya definido o entrenado. Se obtiene entonces un error mínimo del 3,57% según lo descrito en He *et al.*, (2019).

  Este preentrenamiento se realizará mediante Pytorch y el dataset o conjunto de imágenes, en el sistema operativo Jetpack 4.5, comenzando a direccionar en la carpeta de "training", luego a detection en la carpeta del modelo "ssd", como se indica en los siguientes comandos de la Figura 14, los cuales descargarán automáticamente el modelo base donde comenzará el entrenamiento, ya que se modificará tanto al número de etiquetas, el nombre de la etiqueta y se direccionará el entrenamiento a la carpeta *data* donde estará el *data set* (Figura 15).

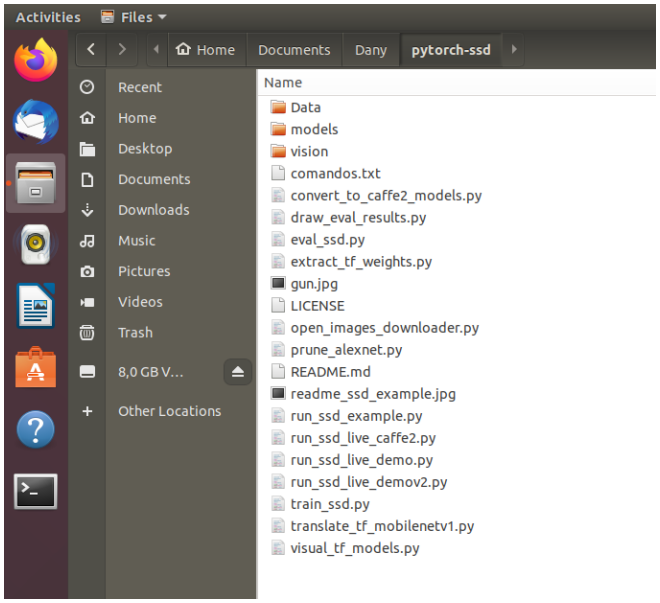
Figura 14: Comandos necesarios para pre entrenar el modelo

```
$ cd jetson-inference/python/training/detection/ssd

$ wget https://nvidia.box.com/shared/static/djf5w54rjvpqocsiztzaandq1m3avr7c.pth -O models/mobilenet-v1-ssd-mp-0_675.pth

$ pip3 install -v -r requisitos.txt
```

Figura 15: Conjunto de archivos obtenidos después de realizar el pre entrenamiento



En la *data set* se aumenta el número de imágenes mediante un repositorio adyacente de NVIDIA con respecto a fresas teniendo un numero de 1490 fotografías. En la práctica los desarrolladores de NVIDIA dicen que se reduce el tiempo de acuerdo al número de imágenes que se tiene, y que es mejor mantener un *data set* menor a 10000. Aunque depende mucho de la aplicación, ya que entre más imágenes más preciso será el modelo.

Estas imágenes serán distribuidas en 3 carpetas que son: *test*, *train* y *validation* por concepto de *Machine Learning*, en las cuales cada una determina un punto muy importante en el entrenamiento. A continuación de describe cada una de ellas:

Train data set: permite entrenar al algoritmo determinando el valor de los parámetros, encontrando los valores de manera automática para cada uno de los pesos (pesos), los cuales se utilizan instantáneamente en el entrenamiento, ya que el algoritmo debe tener acceso a todos estos datos.

Validation data set: permite aprender los hiper parámetros de nuestros algoritmos, los

cuales son requerimientos establecidos manualmente antes de comenzar el entrenamiento y son utilizados para encontrar los valores ideales de los parámetros del modelo, obteniendo otros valores de referencia para corroborar que el aprendizaje de nuestro algoritmo se está realizando de buena manera determinando el *lost* o *pérdida de precisión*.

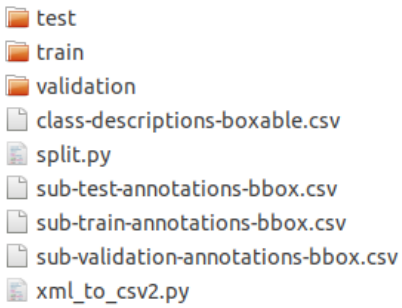
Test set: es el conjunto de imágenes de prueba que ofrece una manera de corroborar la selección de híper parámetros, así como los parámetros que fueron aprendidos, y estos funcionan en nuevos datos, además de utilizar el mismo conjunto de datos nos ayuda a tener un terreno justo para comprobar el desempeño de diversos algoritmos.

A simple vista el *Test set* y el *Validation set* parecen similares, pero no lo son, ya que, si se modifican los hiperparámetros de acuerdo solo al segundo, podemos incurrir en una situación de *overbooking*, que sucede cuando el modelo funciona bastante bien con los datos de entrenamiento, pero tiene un desempeño relativamente pobre al momento de la validación.

En este caso de estudio se distribuyó del número total de imágenes de la siguiente forma: 60% a *Train*, 20% a *Validation* y 20% a *Test*.

Dada la organización de los datos, es importante establecer un archivo .csv (valores separados por comas) que básicamente comprime todo el conjunto de .xml de cada carpeta de la data. Cada uno de estos archivos deben tener la nomenclatura de la Figura 16, lo cual se realiza ejecutando en Python el Scrip llamado xml_to_csv2.py (Anexo 2.1), ubicado en la carpeta *data*, donde se convertirá carpeta por carpeta obteniendo los archivos csv.

Figura 16: Obtención de archivos CSV que permiten entrenar el modelo



Una vez realizados los archivos csv se activamos el ambiente generado previamente con Anaconda, y se ubica la dirección de la carpeta donde está la carpeta principal de ejecución y posteriormente se inicia el entrenamiento, según lo descrito en la Figura 17.

Figura 17: Proceso de activación del entorno de ejecución del modelo

```
$ conda activate jetson

$ cd /home/user/Documents/Dany/pytorch-ssd

$python train_ssd.py --dataset_type open_images --datasets /home/user/Documents/Dany/pytorch-ssd/Data/fresas --net mb1-ssd --pretrained_ssd /home/user/Documents/Dany/pytorch-ssd/models/mobilenet-v1-ssd-mp-0_675.pth --scheduler cosine --lr 0.01 --t_max 100 --validation_epochs 5 --num_epochs 200 --base_net_lr 0.001 --batch_size 11
```

En el último comando tenemos una línea con algunas especificaciones que son el *dataset_type, datasets, net, pretrained_ssd, scheduler, lr, t_max, Validation_epochs, num_epochs, base_net_lr y Bach size,* los cuales se describen en la siguiente Tabla 7, y con estos parámetros enviamos el código a consola para comenzar el entrenamiento.

Tabla 7: Descripción de términos implementados en el código.

Argumento	Valor	Descripción
dataset_type	open_images.	El tipo de *dataset* donde son abiertas las imágenes etiquetadas.
datasets	/home/user/Documents/Dany/pytorch-ssd/Data/fresas.	Este parámetro es la ruta para leer el *dataset*.
net	mb1-ssd.	Tipo de diseño de red.
pretrained_ssd	/home/user/Documents/Dany/pytorch-ssd/models/mobilenet-v1-ssd-mp-0_675.pth	Este parámetro es la ruta para cargar el modelo preentrenado, que es donde se descarga y descomprime *ssd_mobilenet*.
scheduler	cosine	Establece la tasa de aprendizaje de cada grupo de parámetros utilizando coseno, y este se establece de acuerdo *lr* y *num_epochs*.
lr	0,0001	Tasa de aprendizaje mínima, tomada a 0,0001, ya que según los desarrolladores de Pytorsh si los epoch ≥ 80 la tasa de aprendizaje debe ser de 0,005 (Pytorch, 2021)

t_max	200	Número máximo de iteraciones, por defecto se usa 200.
Validation_epochs	5	Este parámetro da el número de cada ciclo en el que se realizará cada validación, por lo general se establece en 1 (uno), pero en este caso se corroboró 5 veces los datos para exactitud.
num_epochs	200	200 es el número total de pasos de iteración, ya que este sistema se define estocástico, debido a que todas las descripciones y el número de parámetros determinan algunos valores que no se pueden predecir con exactitud.
base_net_lr	0,001	Tasa de aprendizaje inicial para la red base, la cual depende básicamente de la red *mb1-ssd* con resolución de 0,001.
Bach_size	2	Argumento para aumentar el ritmo del proceso dependiendo de la memoria disponible.

Una vez generado el comando se observará el proceso que da el dato de pérdida de cada ciclo, y así continuará hasta que complete el número máximo de *epoch*, como se muestra en la Figura 18 y Figura 19 .

Figura 18: Proceso de entrenamiento del modelo

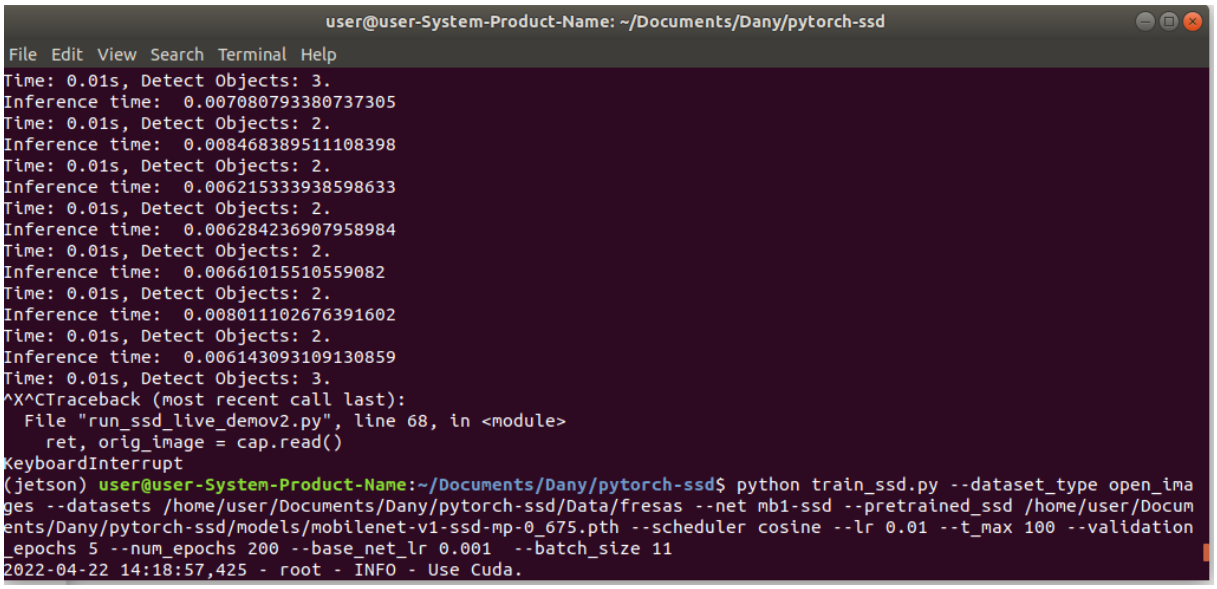
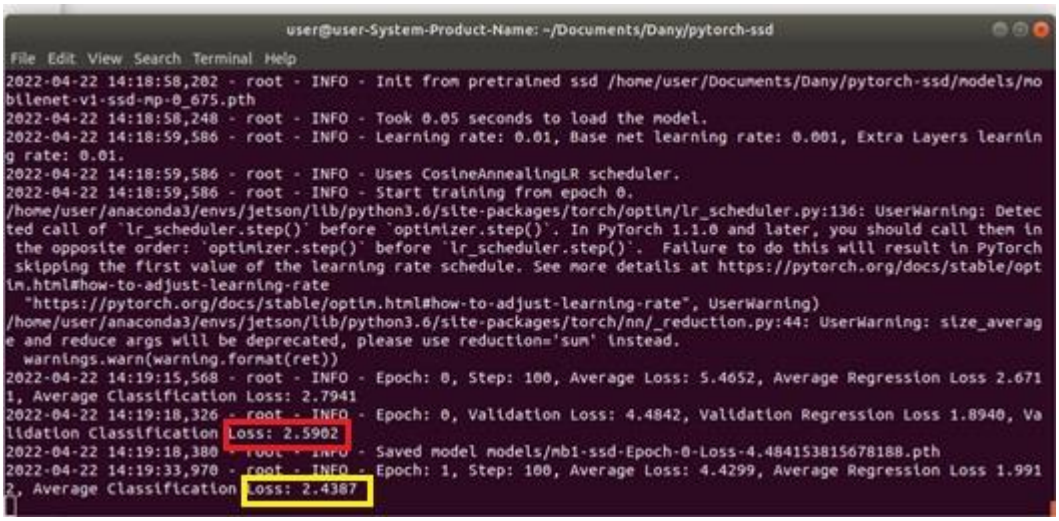


Figura 19: Comprobación del funcionamiento del modelo a través del parámetro "lost"



- **Actividad 8:** "Prueba del algoritmo y reaprendizaje": Antes de probar el primer algoritmo se revisaron los modelos generados, y se seleccionó el que tenía menor perdida, ejecutándose con el comando de la Figura 20.

Figura 20: Comando necesario para ejecutar el modelo en el proceso de selección de fresa

```
$ python run_ssd_live_demov2.py mb1-ssd models/mb1.pth models/labels.txt 0.5
```

Fase 4: Pre-entrenamiento de las Redes Neuronales del modelo generado con la Data para Diseño del algoritmo de visión artificial

En esta fase se generó el comprimido de la *data* con el formato que el modelo puede entrenar. Un pre-entrenamiento es el inicio de aprendizaje del algoritmo, aplicando distintos comandos por consola.

- **Actividad 9:** "Reentrenamiento del modelo generado con la data para diseño del algoritmo de visión artificial": En esta fase en particular se repite la actividad 7, hasta obtener el resultado deseado, que es un alto porcentaje de asertividad en la detección del producto y un *lost* bajo.

- **Actividad 10:** "Realizar el procesamiento de imágenes y conversión de coordenadas gráficas a coordenadas reales": En este caso se utilizó la detección del objeto para hallar el centroide del fruto, evidenciando las coordenadas gráficas que se aplican en la imagen.

Para determinar que las características de cultivo sean ideales en la visión artificial, se utiliza diferentes tratamientos de imágenes, donde se hará la programación gráfica de acuerdo con un mapeo y calibración según los objetos en la realidad.

- **Actividad 11:** "Combinación del algoritmo con el procesamiento de imágenes, estableciendo y comparaciones coordenadas": Una vez finalizado el proceso de obtener las coordenadas del objeto se procedió a las pruebas.

- **Actividad 12:** "Convertir los archivos de datos en archivos de información ONIX": Una vez que obtenidas las coordenadas, los archivos fueron convertido a formato ONIX, el cual permite la gestionar la información de manera más rápida y eficaz.

Fase 5: Implementación del sistema artificial en el actuador.

Culminando el desarrollo del algoritmo se pone a prueba el sistema que enviará las coordenadas, a un actuador guiado con mecanismos.

- **Actividad 13:** "Pruebas de Campo": Se llevó el equipo ensamblado al campo donde se elabora la cosecha de las fresas, procediendo a realizar pruebas de campo.

3.4 Consideraciones bioéticas

El trabajo se desarrolló en un entorno natural, donde el equipamiento no tiene impacto con el medio ambiente, ya que utiliza electrónica digital, y está equipado con un cargador solar, cumpliendo el ciclo de la sustentabilidad.

Hay que tener en cuenta que este trabajo ayudará en el trabajo de los jornaleros, cuidando así la salud de todos los trabajadores, incluso en la parte de la manipulación directa con el cultivo por los pesticidas o químicos que se emplea en el terreno.

CAPÍTULO IV

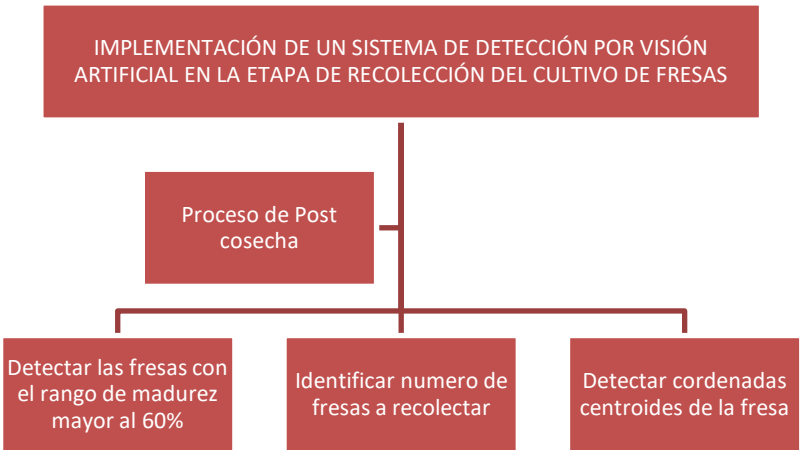
RESULTADOS Y DISCUSIÓN

4.1 Especificaciones del sistema a diseñar

Para lograr la solución considerando las cualidades que tiene una fresa madura lista para su cosecha, se requiere que cumpla con ciertos requisitos para su posterior clasificación, selección y cosecha. Por tal motivo se ha determinado que las características y condiciones esperadas para este trabajo sean:

- Las coordenadas de posición de las fresas deben ser precisa.
- El rango de madurez de la fresa sea mayor al 60%.
- El modelo que va a ser entrenado debe procesar un alto nivel de fotogramas por segundo (FPS).
- El modelo debe tener un buen equilibrio entre precisión y velocidad en el procesamiento de información.
- El modelo debe tener un alto nivel de mapeo en el procesamiento en tiempo real.
- La precisión del detector no debe ser afectada en el momento de extraer las características de las fresas.
- El tamaño de objetos que puede detectar el modelo debe ser el adecuado al tamaño de las fresas.
- La cámara de detección debe proporcionar una buena resolución de imagen.
- La memoria del controlador debe tener como mínimo una capacidad de 1Gb.

- *Figura 21: Proceso de selección de las fresas a través del modelo.*



Para la implementación del sistema se identificaron tres variables de salida en el modelo de visión artificial, como se explica en la Figura 21.

4.1.1 Funcionamiento

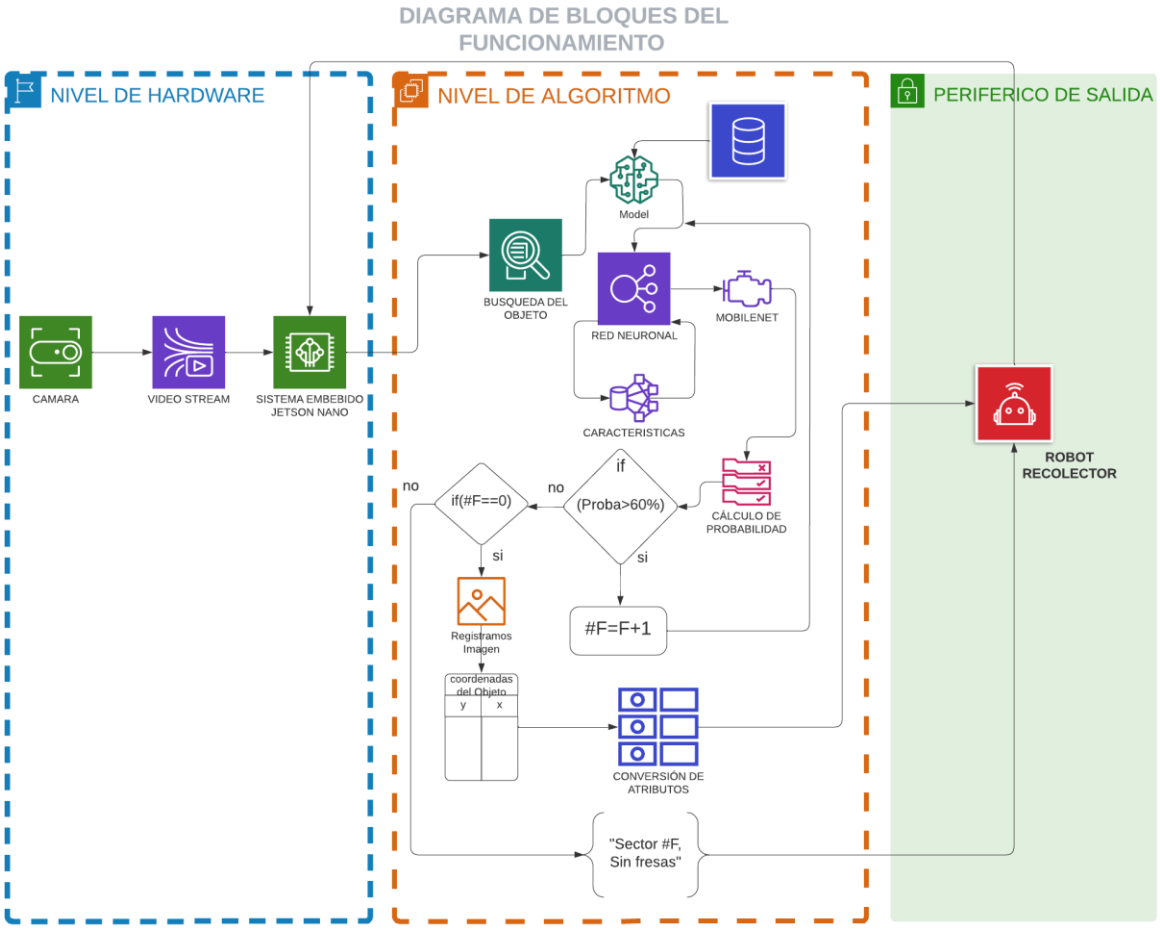
El funcionamiento del algoritmo es a nivel de software, no obstante, tanto periféricos de entrada como de salida apropian su propio hardware y software, es decir que en el proceso de funcionamiento se tienen: nivel de hardware, de algoritmo y periférico de salida, como se observa en la Figura 22.

- **Nivel de hardware:** En este punto tenemos la funcionalidad de la cámara, tomando imágenes aproximadas al tiempo real es decir a 144HZ, no obstante, por el stream y el procesamiento de imagen se puede decir que baja a 50 fps lo cual se considera buena jugabilidad. Toda esta información de fotogramas es enviada a la tarjeta embebida "JETSON Nano" para entrar a nivel del algoritmo.

- **Nivel de algoritmo:** En este nivel el procesamiento de la imagen por *frame* donde se encarga de detectar varios elementos de una imagen y clasificarlos, a través del modelo entrenado anteriormente, donde se recolecto de forma automática los parámetros e hiperparametros, para poder diferenciarlos de pixel a pixel, comparando sus características, generando una dinámica según el modelo Mobilenet.

  Este modelo calcula la probabilidad del objeto de detección, este a su vez entra una aprobación que debe de ser mayor al 60%; si ese es el caso aumentará el número de objetos o de fresas detectadas, generando un ciclo hasta que todas las fresas de la imagen sean contadas, después registra la imagen general con todas las fresas, dando un tratamiento de imágenes para hallar su centroide, que es el punto exacto de coordenadas. La organización de comunicación comenzará en un barrido de izquierda a derecha, mandando al periférico de salida las coordenadas de la fresa más cercana al punto de origen de la matriz de la imagen; una vez que el dispositivo recoja la fresa de las coordenadas enviadas, recibirá un comando para poder realizar el mismo ciclo, hasta que no haya más fresas.

Figura 22: Esquema de funcionamiento del modelo.



- **Periférico de salida:** el periférico de salida es el robot cartesiano, que se utilizará para la recolección de la fresa.

Cada una de las etapas del sistema integral de recolección de fresas, debe ser normado en una comunicación *Half duplex*, para que tenga un trabajo de coordinación y sincronización con cada uno de los elementos del sistema.

4.2 Resultados y análisis

Para resolver las necesidades de la empresa, en primera instancia, se realizó un análisis y se determinó los subprocesos o procedimientos necesarios para satisfacer los objetivos planteados. Se seleccionó el hardware necesario, se diseñó y desarrolló el modelo de programación y detección del producto, posteriormente se implementó y se realizaron pruebas de funcionamiento para verificar los resultados que se estimaron.

4.2.1 Limitación de la cantidad de datos

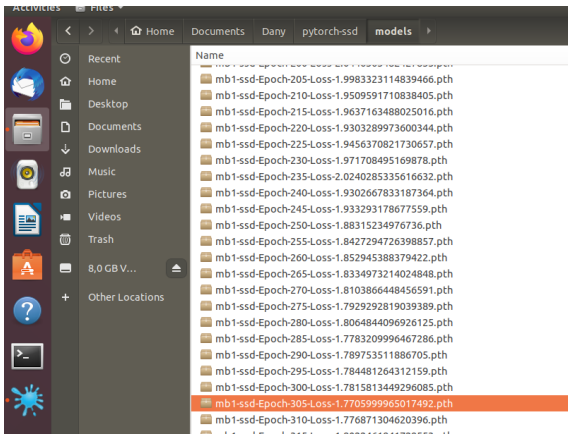
La funcionalidad del modelo depende en gran parte de la cantidad de imágenes, entre más imágenes tenga un modelo, más preciso será. Pero también aumenta el tiempo de entrenamiento y disminuye el espacio disponible en el disco, por ello es necesario delimitar la cantidad de datos que se van a procesar, con lo que se llevó a 1490 fotografías en su *dataset*.

4.2.2 Entrenamiento del modelo

Con el primer entrenamiento, comenzamos con un error métrico o pérdida que fue de 12,767890%.

Partiendo de lo establecido en la actividad 7 del capítulo anterior, se realizaron 219 entrenamientos almacenadas en la carpeta "*models"* (Figura 23), los cuales tomaron aproximadamente dos semanas, producto de ello se obtuvo un modelo con una pérdida de 1,770559%.

Figura 23: Proceso de repetición del entrenamiento del modelo.

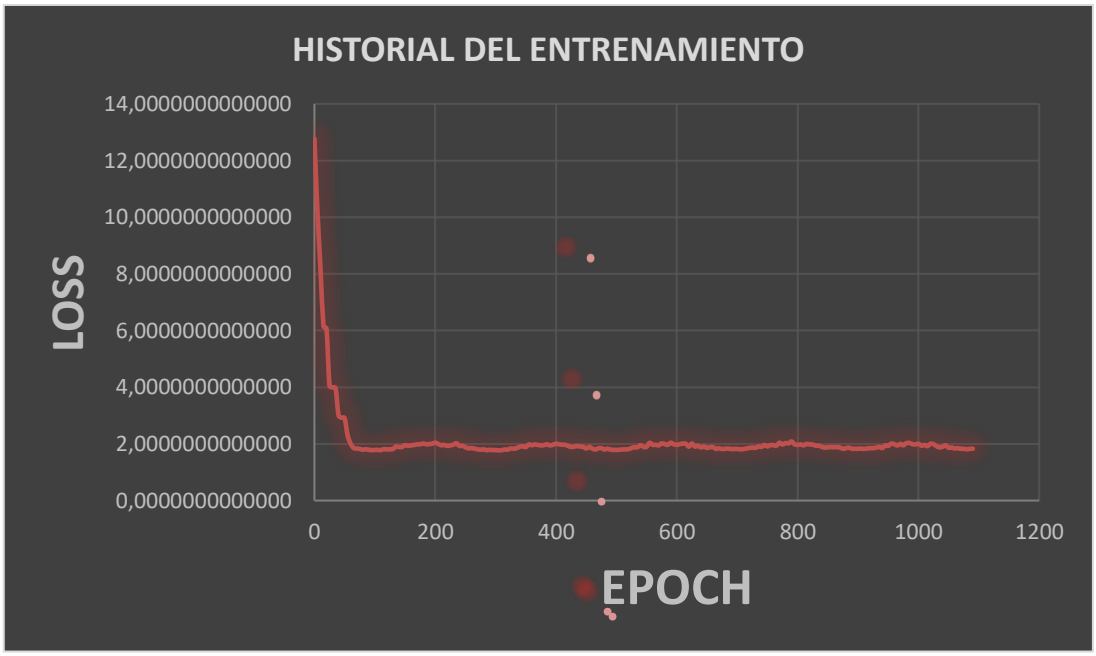


Analizando el historial del entrenamiento del modelo, según la Figura 24, se puede deducir que a partir del *epoch* 100 hasta el 1100 se encuentra básicamente en un punto muerto, lo que significa que por más entrenamientos que se realicen no baja de la perdida mínima, es decir no bajará de 1,770559% que es el valor mínimo. A este valor se conoce como el termino de *Best Value de la gradiente descendente*, ya que es el más cercano al 0% de error que sería la perfección.

4.2.3 Configuración del dispositivo y conexión de tarjetas JETSON y ARDUINO

En este caso para implementar el algoritmo se recomienda poner el dispositivo en modo ejecución, es decir que se deshabilitará el modo gráfico de escritorio, para configurar un comando de ejecución automático al momento que se encienda la JETSON. Esto no solo ayuda a generar el proceso automático, sino también a mejorar la rapidez del proceso, ya que no se ocupará procesamiento ni memoria en la parte gráfica.

Figura 24. Grafica sobre el historial del entrenamiento del modelo SSD Mobilenet V1

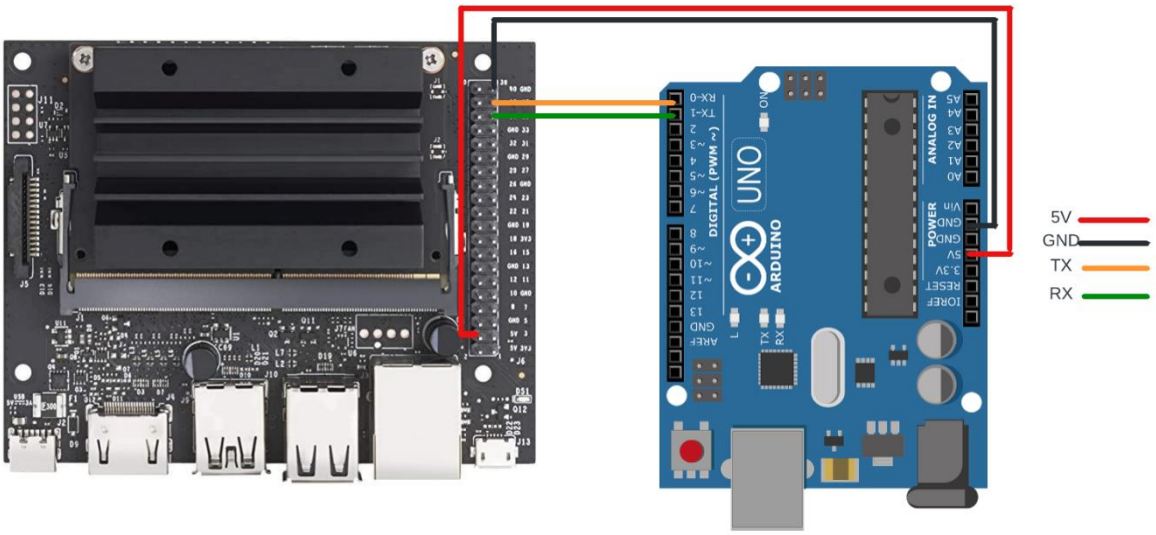


Una vez que el modelo ha sido entrenado, se procede a realizar las conexiones de los controladores, los cuales permiten el procesamiento de las imágenes que son recibidas por la cámara, y la recepción de las coordenadas y ejecución de procedimientos de cosecha de las fresas. No obstante, es importante realizar la comunicación serial de la JETSON y el Arduino, conectando sus pines de transmisión y recepción de datos, y su respectiva alimentación como se puede ver en la Figura 25.

Al realizar la configuración de la comunicación entre la JETSON y Arduino se debe realizar de manera *half dúplex*, de forma que no exista ninguna colisión de información y el proceso de intercambio de información sea eficiente.

Arduino cumplirá la función del robot recolector, el cual obtiene las coordenadas de las fresas para el manejo de los actuadores en el proceso de cosecha.

Figura 25: Diagrama de conexiones entre la JETSON y ARDUINO 1



4.2.4 Análisis y valoración del modelo

Para el análisis y evaluación del rendimiento del modelo de aprendizaje aplicado en el algoritmo de visión artificial, se utiliza *la Matriz de Confusión y la curva de ROC*.

La curva de ROC(Receiver Operating Charteristic, o Característica Operativa del Receptor), una gráfica que indica la relación grafica entre la tasa de Verdaderos Positivo que |la tasa de Falsos Positivos son el número de casos que la prueba declara positivos y que en realidad son negativos.

Para recrear a la curva de ROC se realizó una matriz de confusión, que se utilizó para probar el modelo de clasificación, teniendo en cuenta una muestra positiva, que son un numero de imágenes que contienen el objeto a detectar, y una muestra negativa que son imágenes randomicas sin el objeto (Vasilev, 2019).

De estas imágenes se cataloga 4 estados que son:

a) **Verdaderos Positivos:** son respuestas afirmativas de detección en imágenes con el objeto a detectar.

b) **Falsos Negativos:** son respuestas negativas de detección en imágenes sin el objeto a detectar.

c) **Falsos Positivos:** son respuestas negativas de detección en imágenes con el objeto a detectar.

d) **Verdaderos Negativos:** son respuestas afirmativas de detección en imágenes sin el objeto a detectar.

Con el numero de cada uno de los estados, proporcionaremos las propiedades básicas del algoritmo que son:

a) **Exactitud:** es la capacidad de acercarse al valor real.

b) **Precisión:** es cuando se da el mismo resultado en medidas diferentes.

c) **Recuperación / Sensibilidad:** es la fracción de instancias relevantes que se han obtenido sobre las instancias relevante, enfocando los parámetros de compresión y relevancia.

d) **Especificidad (Specificity):** son el numero de casos negativos acertados.

e) **F1 SCORE:** es la medida de precisión que tiene la prueba.

Todos estos valores se miden en el porcentaje, en donde el 100% identifica la perfección del propiedad

- **La matriz de confusión**

Es una representación de parámetros, como se presenta en la Tabla 8, en la cual se trabajó con 200 imágenes de muestras, donde 100 de estas tienen una fresa o varias fresas, por ende, las otras 100 no contendrán ninguna fresa.

Tabla 8. Tabla de Abreviaturas para la matriz de confusión

Términos	Abreviaturas	Descripción
Verdaderos positivos	TP	Predichos positivos y realmente positivos.
Falsos Positivos	FP	Predichos positivos y en realidad son negativos.
Negativos verdaderos	NT	negativos previstos y realmente negativos.
Falsos negativos	FN	Predicho negativo y en realidad es positivo.

El registro fotográfico de la prueba se presenta en el Anexo 3. En esta prueba detallamos cuales de las imágenes fueron verdaderos positivos, verdaderos negativos, falsos positivos y

falsos negativos, en manera de clasificación binaria, donde se evidenció si existió o no la detección de la fresa, y a la vez con estos datos podemos calcular la precisión, recuperación, especificidad, *F1 Score* y la *Curva de ROC* para validar.

En la Tabla 9 tenemos los valores de cada termino, como 86 verdaderos positivos, 13 falsos negativos, 21 falsos positivos y 79 verdaderos negativos.

Tabla 9 Matriz de Confusión

		Predicción	
		Positivo	**Negativo**
Actual	**0**	86	13
	1	21	79

- **Exactitud (Accuracy)**

Esta propiedad del algoritmo se calcula para estimar la exactitud de probabilidad de un modelo, pero este no es un indicador claro de rendimiento. Para el cálculo de precisión se aplica la ecuación (8):

$$\boldsymbol{Exactitud} = \frac{\boldsymbol{TP} + \boldsymbol{TN}}{\boldsymbol{TP} + \boldsymbol{FP} + \boldsymbol{TN} + \boldsymbol{FN}} \quad \textbf{\textit{(2)}}$$

$$Exactitud = \frac{86 + 79}{86 + 21 + 79 + 13} = 0{,}829145729 = 82{,}9\%$$

En este punto sale 82,9% lo cual cataloga el modelo de entrenamiento dentro de un panorama muy bueno con respecto a la probabilidad.

- **Precisión**

Es un porcentaje entre instancias positivas del total de instancias positivas previstas. El denominador de la ecuación (8) es la predicción positiva de la modelo realizada a partir de todo el conjunto de datos dados, afirmando en cuanto acierta el modelo.

$$\boldsymbol{Precisión} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FP}} \quad \textbf{\textit{(3)}}$$

$$Precisión = \frac{86}{86 + 13} = 0{,}803738318 = 80{,}33\%$$

En este punto tenemos el 80,33% del número total de predicciones correctas, lo cual da un panorama positivo ya que se acerca al 100% que es prácticamente la perfección.

- **Recuperación / Sensibilidad / Tasa de verdaderos positivos (Recall/Sensitivity/ True Positive Rate)**

Es el porcentaje de instancias positivas del total de instancias positivas reales, por lo tanto, el denominador de la ecuación (4), es el número real de instancias positivas presentes en este conjunto de datos.

$$\boldsymbol{Sensibilidad = \frac{TP}{TP + FN}} \quad \textit{(4)}$$

$$Sensibilidad = \frac{86}{86 + 21} = 0{,}868686869 = 86{,}86\%$$

El valor obtenido es de 87%, lo cual es aceptable, pero no asegura que sea un buen modelo. Sin embargo, complementando con la información aportada por la precisión, se evidencia que el modelo presenta una sensibilidad de respuesta aceptable.

- **Especificidad (Specificity)**

La especificidad da el porcentaje de instancias negativas del total de instancias negativas reales, donde el denominador de la ecuación (5), es el número real de instancias negativas presentes en ese grupo de datos, es decir que es una especie de medida para saber que tan separadas están las clases.

$$\mathbf{Especificidad} = \frac{\boldsymbol{TN}}{\boldsymbol{TN + FP}} \quad \textit{(5)}$$

$$\text{Especificidad} = \frac{79}{79 + 13} = 0{,}79 = 79\%$$

Este resultado de 80% identifica, una buena distinción de etiqueta, o de selección de clase, es decir que tiene muy poca probabilidad de equivocarse con algo semejante.

- **F1 SCORE**

Es la relación que existe entre la precisión y la recuperación, como se evidencia en la ecuación (6).

$$\boldsymbol{F1SCORE = \frac{2}{\frac{1}{precisión} + \frac{1}{Recuperación}}} \quad \textit{(6)}$$

$$F1SCORE = \frac{2}{\frac{1}{0{,}803738318} + \frac{1}{0{,}868686869}} = 0{,}816244356 = 81{,}62\%$$

El resultado del 0,816244356 demuestra que la puntuación general de rendimiento es del 81,62%, teniendo un algoritmo de buena calidad.

- **Curva ROC**

En la curva ROC se tiene una representación gráfica de la sensibilidad de un sistema clasificador binario, en cual se varía el umbral de discriminación. Este esencialmente se utiliza para evaluar el rendimiento de los algoritmos de clasificación binaria.

Los parámetros de esta gráfica se obtienen calculando la *tasa de verdaderos positivos* (TPR) y la *tasa de falsos positivos* (FPR), con lo cual se obtiene un solo clasificador en una variedad de umbrales. El TPR y FPR se calculan según la ecuación (7) y (8) respectivamente.

$$\boldsymbol{TPR = Sensitivity = \frac{TP}{TP + FN}} \quad \textit{(7)}$$

$$TPR = \frac{86}{86 + 13} = 0{,}868686869$$

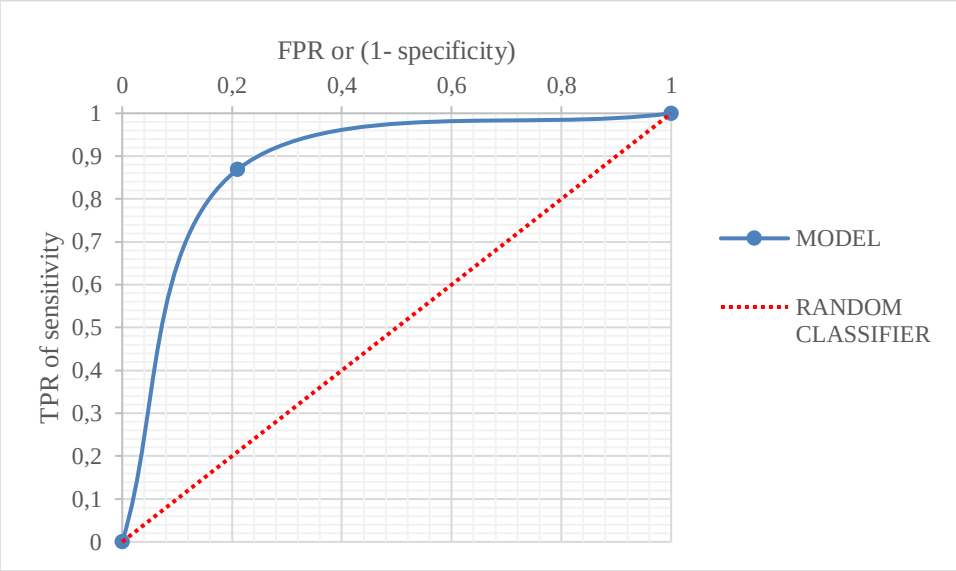
$$\boldsymbol{FPR = 1 - Specificity = \frac{FP}{FP + TN}} \quad \textit{(8)}$$

$$FPR = \frac{21}{21 + 79} = 0{,}21$$

Con estos resultados se ubicó un punto vector en la Figura 26, teniendo en cuenta la línea roja de puntos llamada "*Random Classifier*", que representa un clasificador aleatorio. Se trazó una curva desde el punto de origen hacia el punto de coordenada con los valores anteriormente calculados culminando en el punto (1,1). La curva se analiza de la siguiente manera:

- Un resultado de (1,0): el gráfico lo designa como un modelo de **clasificación perfecta**, y esto se debe a que en este punto existe una máxima sensibilidad, o podría decirse que la tasa de verdaderos positivos está al 100% con respecto a una tasa de falsos positivos de 0%, esto se estima en referencia a (Ericmelillanca, 2017)
- Todo modelo que este debajo de la línea "*Random Classifier*", se considera un **modelo clasificador no apto o no confiable**.

- *Figura 26: Curva de ROC*



Con respecto al modelo que se presenta en este trabajo de investigación, se puede decir que en la perspectiva del vector del modelo (0,21;0,86), tiene un desempeño superior al aleatorio, y que además desempeña un buen rendimiento en su funcionalidad.

4.2.5 Ejecución del programa

Una vez que el programa pudo aprender del modelo, está listo para generar las predicciones de las nuevas imágenes que se le van a suministrar. Para ello es necesario crear un modelo congelado que permita obtener otro modelo ya entrenado con el algoritmo que permite generar las predicciones. Al ejecutar el comando del modelo congelado, este detectará las fresas en las nuevas imágenes que se le suministran al modelo.

Según la Figura 27, el modelo si puede detectar las fresas en las imágenes que fueron captadas a través de *streaming* (en tiempo real), lo cual indica que el algoritmo cumple su objetivo.

Figura 27: Comprobación del funcionamiento del modelo en el campo de cosecha

Al finalizar el proceso de detención de las fresas fue posible determinar que el programa funciona con eficiencia y acorde a las necesidades del proyecto, pero es necesario evidenciar las fallas y complicaciones que se presentaron en el proceso de ejecución, teniendo en respuesta los siguientes puntos:

- **Entrenamiento:** Fue un proceso largo y repetitivo debido a la cantidad de imágenes que se debían analizar y al modelo pre entrenado que se utilizó. Este proceso fue realizado varias veces hasta que se obtuvo un valor de *lost* satisfactorio y que el modelo tuviera un funcionamiento satisfactorio.

- **Condiciones ambientales:** Debido a las condiciones del clima, fue complejo entrenar el modelo ya que los equipos no pueden ser afectados por la lluvia y fue necesario esperar un ambiente climático adecuado. Tomando en consideración que el proceso de cosecha de las fresas se realiza en temporadas con condiciones climáticas similares, no se espera ninguna afectación a la funcionalidad del sistema.

- **Entorno:** Este factor juega un papel importante en el proceso de detección y análisis de objetos cuando se implementa la visión artificial. La iluminación y el fondo que se presentan en las imágenes a procesar afectan el entrenamiento del modelo debido al análisis de píxeles.

- **Tiempo de ejecución:** El proceso no puede ser ejecutado mientras no se realice con luz natural debido a la afectación de la imagen. Si se realiza en la noche, la cámara no puede detectar el objeto, y si se utiliza una cámara nocturna el análisis de imágenes no podría reconocer los objetos.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- La mayoría de los cultivos de fresa de la parte norte del país, utiliza la modalidad de suelo de terrenos sueltos, con una profundidad superior a 80 cm, y la recolección de las fresas se la realiza de forma manual con jornaleros en diferentes horarios del día, razón por la cual todo el trabajo de visión artificial se centró en estas características de cultivos de esta localidad.

- Se diseño un algoritmo de visión artificial para la detección y categorización de la fresa tomando en cuenta los requerimientos de cosecha de la zona norte del país, obteniendo un modelo determinante con una pérdida de 1,72% aproximado y una velocidad de detección de 0,123 segundos, obteniendo un modelo confiable.

- La validez del algoritmo se determinó a través de la *matriz de confusión*, de la cual se obtuvo un 82,99% de exactitud, 80,33% de precisión, 86,6% de recuperación, 79% de especificidad y 81,62% de F1Score, que son valores que permiten concluir que el algoritmo es confiable y apto para la detección.

- Se determinó a partir de las pruebas realizadas que la tarjeta embebida JETSON NANO especializada en visión artificial, es apta para esta implementación del algoritmo de visión artificial, ya que dentro de su sistema operativo tiene elementos de propiedades gráficas que ayudan a tener una buena respuesta del modelo, como fue el caso de CUDA, la cual permitió obtener mejoras en el procesamiento de información.

5.2 Recomendaciones

- En investigaciones posteriores se recomienda hacer la evaluación en cultivos aéreos, ya que en este caso la fresa cuelga de la planta, pudiendo aplicarse un tipo de fotometría básica de cuatro perspectivas, para tener una exactitud mayor al porcentaje de madurez y a la calidad de la fresa.

- Se recomienda al momento de entrenar con el modelo SSD-Movilenet V1, tener un segmento de perdida bajo el 2,0.

- Es preciso decir que la precisión de la detección puede mejorar con una cámara, de acuerdo con los FPS que lleve o al tipo ISO que ocupe de acuerdo al ambiente.

- Al momento de poder probar en el robot cartesiano se recomienda una calibración externa de acuerdo con los límites de cuadro de imagen, para la transformación de pixeles a medidas reales.

- Para trabajos futuros, se recomienda utilizar tecnología IOT, que permita implementar el algoritmo en un servidor remoto en un servidor bien equipado con recursos de hardware.

- Es recomendable utilizar un equipo acorde a la cantidad de información que se va a procesar, debido a que a mayor cantidad de imágenes que se utilizan para entrenar el modelo, mayor es el tiempo de procesamiento.

- Para disminuir el tiempo que entrenamiento del modelo, se recomienda realizar el proceso fuera de la JETSON. Una vez finalizado el entrenamiento ya puede ser utilizado en el controlador.

REFERENCIAS

Acuña, J. (2021). *Fresa (Fragaria × ananassa Duch.)*.

Anaconda. (2019, October 14). *Anaconda states*. Https://Www.Anaconda.Com/.

Bonilla, C. (2010). *Fresa* (2nd ed.). Cartillas del Corredor .

Colab Google. (2020, November 1). *Google Colab*. Https://Colab.Research.Google.Com/#scrollTo=lSrWNr3MuFUS.

Constante, P., Chang, O., Pruna, E., & Escobar, I. (2016). Artificial Vision Techniques for Strawberry ' s Industrial Classification. *Ieee Latin America Transactions*, *14*(6), 2576–2581.

Córtes, P. (2011). Propuesta técnica-ambiental para asegurar la inocuidad de fresas cultivadas en Cartago. *San José Costa Rica 2011; Consultado 1-Sept-2014*.

Cubero, S., Aleixos, N., Moltó, E., Gómez-Sanchis, J., & Blasco, J. (2011). Advances in Machine Vision Applications for Automatic Inspection and Quality Evaluation of Fruits and Vegetables. *Food and Bioprocess Technology*, *4*(4), 487–504. https://doi.org/10.1007/s11947-010-0411-8

Dmitry Khort , Alexey Kutyrev , Igor Smirnov1 Volodymyr Osypenko, and N. K. (2020). Computer Vision System for Recognizing the Coordinates Location and Ripeness of Strawberries. *Data Stream Mining & Processing*. https://doi.org/10.1007/978-3-030-61656-4

Docker. (2020a, August 12). *Docker about*. Https://Www.Docker.Com/Company.

Docker. (2020b, August 12). *https://www.docker.com/company*. Docker About.

Durand-Petiteville, A., Vougioukas, S., & Slaughter, D. C. (2017). Real-time segmentation of strawberry flesh and calyx from images of singulated strawberries during postharvest processing. *Computers and Electronics in Agriculture*, *142*, 298–313. https://doi.org/10.1016/j.compag.2017.09.011

Ericmelillanca. (2017, December 23). *Evaluación de modelos de clasificación: Matriz de Confusión y Curva ROC*. Ericmelillanca.

Faura, F. (2010). Fresa (Fragaria x ananassa Duch) Producción y Manejo Poscosecha. *Biblioteca Agropecuaria de Colombia, (BAC)*.

Flores, F., & Mora, C. (2015). Manual de fresa. *PROGRAMA DE APOYO AGRÍCOLA Y AGROINDUSTRIAL VICEPRESIDENCIA DE FORTALECIMIENTO EMPRESARIAL CÁMARA DE COMERCIO DE BOGOTÁ*, 1–54.

Fuentes-Doria, D. D., Toscano-Hernández, A. E., Malvaceda-Espinoza, E., Díaz Ballesteros, J. L., & Díaz Pertuz, L. (2020). Metodología de la investigación: Conceptos, herramientas y ejercicios prácticos en las ciencias administrativas y contables. In *Metodología de la investigación: Conceptos, herramientas y ejercicios prácticos en las ciencias administrativas y contables*. https://doi.org/10.18566/978-958-764-879-9

Gollapudi, S. (2019a). *Learn Computer Vision Using OpenCV: With Deep Learning CNNs and RNNs*.

Gollapudi, S. (2019b). *Learn Computer Vision Using OpenCV: With Deep Learning CNNs and RNNs*.

He, K., Zhang, X., Ren, S., & Sun, J. (2019). *Deep Residual Learning for Image Recognition*. http://image-net.org/challenges/LSVRC/2015/

Intel Corporation. (2022, February 14). *Classification Models*. Https://Docs.Openvino.Ai/2021.3/Omz_models_model_resnet_50_pytorch.Html. https://docs.openvino.ai/2021.3/classification_models_public.html

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2014). *ImageNet Classification with Deep Convolutional Neural Networks*. http://code.google.com/p/cuda-convnet/

Morena, A., Molina, G., & Garay, U. (2019). Artificial Visión and Language Processing for Robotics. In *Packt* (1st ed.). Packt Publishing Ltd. https://doi.org/10.2514/5.9781600868962.0000.0000

Müller, A. C., & Guido, S. (2017). *Introduction to Machine Learning with Python* (Vol. 2).

NVIDIA. (2021a, June 22). *VISTA PREVIA PARA DESARROLLADORES DE JETPACK SDK 5.0*. Https://Developer.Nvidia.Com/Jetpack-Sdk-50dp.

NVIDIA. (2021b, June 22). *VISTA PREVIA PARA DESARROLLADORES DE JETPACK SDK 5.0*. Https://Developer.Nvidia.Com/Jetpack-Sdk-50dp. https://developer.nvidia.com/jetpack-sdk-50dp

Pytorch. (2021, January 21). *Pytorch*. Https://Pytorch.Org/.

Saenz, E., Jimenez, M., & Ramirez, A. (2013). Strawberries collecting robot prototype in greenhouse hydroponic systems. *Symposium of Signals, Images and Artificial Vision - 2013, STSIVA 2013*. https://doi.org/10.1109/STSIVA.2013.6644933

Sandino Mora, J. D., Amaya Hurtado, D., & Ramos Sandoval, O. L. (2016). Monitoreo Preliminar De Incidencia De Fisiopatías En Cultivos De Fresa Usando Procesamiento Digital De Imágenes. *Biotecnoloía En El Sector Agropecuario y Agroindustrial*, *14*(1), 45. https://doi.org/10.18684/bsaa(14)45-52

Sentisight. (2020, May 15). *Anotación de imagen*. Https://Www.Sentisight.Ai/.

Soto, J. O. (2018). El Cultivo de la Fresa en el Perú. *INIA*, *1*(1), 1–8.

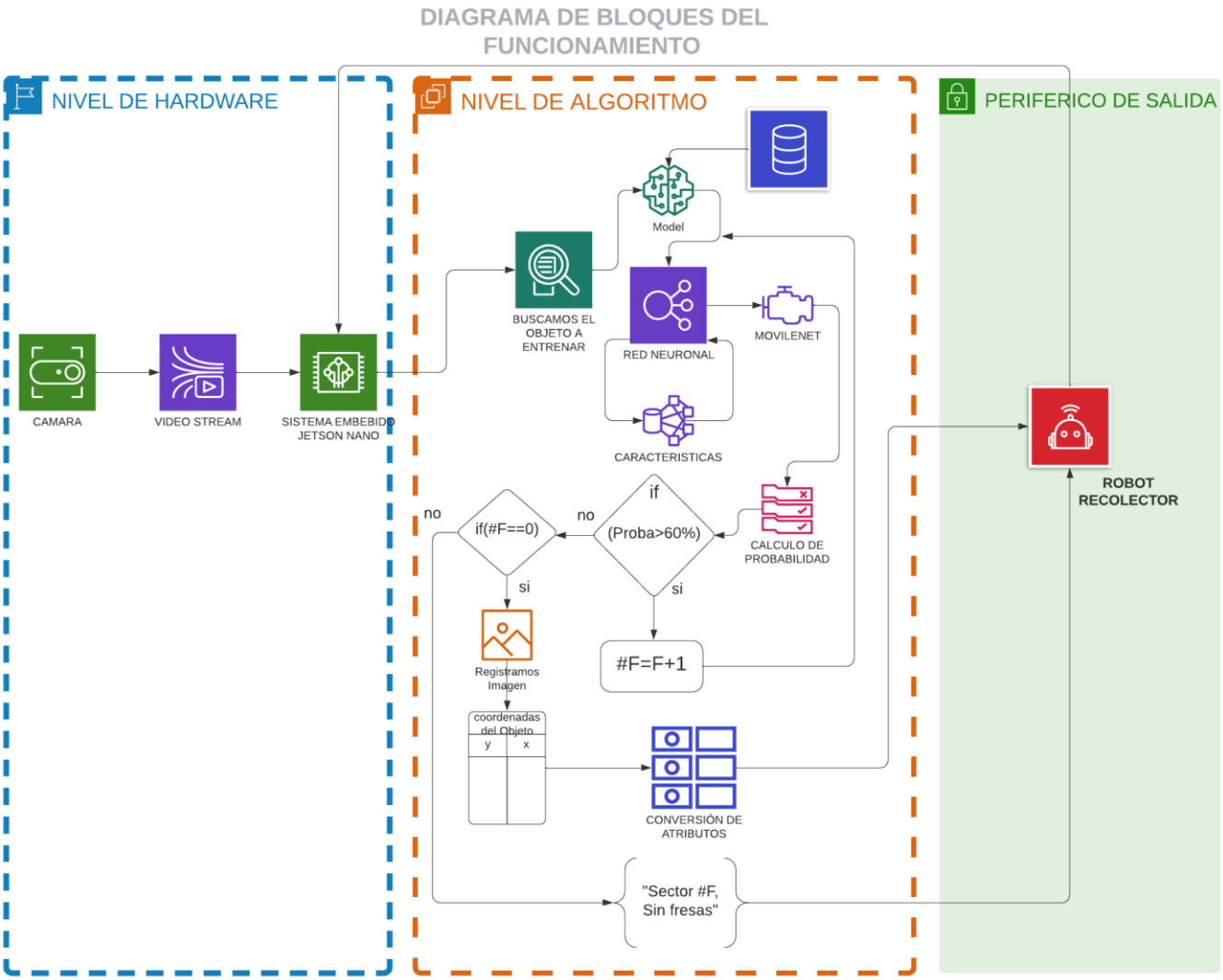
Towardsdatascience. (2019). *Varias formas de evaluar el rendimiento de un modelo de aprendizaje automático*.

Vasilev, I. (2019a). *Python deep learning : exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow*.

Vasilev, I. (2019b). *Python deep learning : exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow*.

ANEXOS

Anexo 1: Diagrama de bloques del funcionamento



Anexo 2: Script de conversión de archivos xml a csv

```
import os
import glob
import pandas as pd
import xml.etree.ElementTree as ET
def xml_to_csv(path):
  xml_list = []
  for xml_file in glob.glob(path + '/*.xml'):
    tree = ET.parse(xml_file)
    root = tree.getroot()
    for member in root.findall('object'):
      value = (root.find('filename').text,
           int(root.find('size')[0].text),
           int(root.find('size')[1].text),
           member[0].text,
           int(member[4][0].text),
           int(member[4][1].text),
           int(member[4][2].text),
           int(member[4][3].text)
           )
      xml_list.append(value)
  column_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax', 'ymax']
  xml_df = pd.DataFrame(xml_list, columns=column_name)
  return xml_df
def main():
  image_path = os.path.join(os.getcwd(), 'annotations')
```

```
  xml_df = xml_to_csv(image_path)
  xml_df.to_csv('raccoon_labels.csv', index=None)
  print('Successfully converted xml to csv.')
main()
```

Anexo 3: Script de entrenamiento del algoritmo

```
import argparse
import os
import logging
import sys
import itertools
import torch
from torch.utils.data import DataLoader, ConcatDataset
from torch.optim.lr_scheduler import CosineAnnealingLR, MultiStepLR
from vision.utils.misc import str2bool, Timer, freeze_net_layers, store_labels
from vision.ssd.ssd import MatchPrior
from vision.ssd.vgg_ssd import create_vgg_ssd
from vision.ssd.mobilenetv1_ssd import create_mobilenetv1_ssd
from vision.ssd.mobilenetv1_ssd_lite import create_mobilenetv1_ssd_lite
from vision.ssd.mobilenet_v2_ssd_lite import create_mobilenetv2_ssd_lite
from vision.ssd.mobilenetv3_ssd_lite import create_mobilenetv3_large_ssd_lite, create_mobilenetv3_small_ssd_lite
from vision.ssd.squeezenet_ssd_lite import create_squeezenet_ssd_lite
from vision.datasets.voc_dataset import VOCDataset
from vision.datasets.open_images import OpenImagesDataset
from vision.nn.multibox_loss import MultiboxLoss
from vision.ssd.config import vgg_ssd_config
from vision.ssd.config import mobilenetv1_ssd_config
from vision.ssd.config import squeezenet_ssd_config
from vision.ssd.data_preprocessing import TrainAugmentation, TestTransform
parser = argparse.ArgumentParser(
  description='Single Shot MultiBox Detector Training With Pytorch')

parser.add_argument("--dataset_type", default="voc", type=str,
        help='Specify dataset type. Currently support voc and open_images.')
parser.add_argument('--datasets', nargs='+', help='Dataset directory path')
parser.add_argument('--validation_dataset', help='Dataset directory path')
parser.add_argument('--balance_data', action='store_true',
```

```
        help="Balance training data by down-sampling more frequent labels.")
parser.add_argument('--net', default="vgg16-ssd",
        help="The network architecture, it can be mb1-ssd, mb1-lite-ssd, mb2-ssd-lite, mb3-large-ssd-lite, mb3-small-ssd-lite or vgg16-ssd.")
parser.add_argument('--freeze_base_net', action='store_true',
        help="Freeze base net layers.")
parser.add_argument('--freeze_net', action='store_true',
        help="Freeze all the layers except the prediction head.")
parser.add_argument('--mb2_width_mult', default=1.0, type=float,
        help='Width Multiplifier for MobilenetV2')
# Params for SGD
parser.add_argument('--lr', '--learning-rate', default=1e-3, type=float,
        help='initial learning rate')
parser.add_argument('--momentum', default=0.9, type=float,
        help='Momentum value for optim')
parser.add_argument('--weight_decay', default=5e-4, type=float,
        help='Weight decay for SGD')
parser.add_argument('--gamma', default=0.1, type=float,
        help='Gamma update for SGD')
parser.add_argument('--base_net_lr', default=None, type=float,
        help='initial learning rate for base net.')
parser.add_argument('--extra_layers_lr', default=None, type=float,
        help='initial learning rate for the layers not in base net and prediction heads.')
# Params for loading pretrained basenet or checkpoints.
parser.add_argument('--base_net',
        help='Pretrained base model')
parser.add_argument('--pretrained_ssd', help='Pre-trained base model')
parser.add_argument('--resume', default=None, type=str,
        help='Checkpoint state_dict file to resume training from')
# Scheduler
parser.add_argument('--scheduler', default="multi-step", type=str,
        help="Scheduler for SGD. It can one of multi-step and cosine")
# Params for Multi-step Scheduler
```

```
parser.add_argument('--milestones', default="80,100", type=str,
        help="milestones for MultiStepLR")
# Params for Cosine Annealing
parser.add_argument('--t_max', default=120, type=float,
        help='T_max value for Cosine Annealing Scheduler.')

# Train params
parser.add_argument('--batch_size', default=32, type=int,
        help='Batch size for training')
parser.add_argument('--num_epochs', default=120, type=int,
        help='the number epochs')
parser.add_argument('--num_workers', default=4, type=int,
        help='Number of workers used in dataloading')
parser.add_argument('--validation_epochs', default=5, type=int,
        help='the number epochs')
parser.add_argument('--debug_steps', default=100, type=int,
        help='Set the debug log output frequency.')
parser.add_argument('--use_cuda', default=True, type=str2bool,
        help='Use CUDA to train model')
parser.add_argument('--checkpoint_folder', default='models/',
        help='Directory for saving checkpoint models')
logging.basicConfig(stream=sys.stdout, level=logging.INFO,
        format='%(asctime)s - %(name)s - %(levelname)s - %(message)s')
args = parser.parse_args()
DEVICE = torch.device("cuda:0" if torch.cuda.is_available() and args.use_cuda else
"cpu")
if args.use_cuda and torch.cuda.is_available():
  torch.backends.cudnn.benchmark = True
  logging.info("Use Cuda.")
def train(loader, net, criterion, optimizer, device, debug_steps=100, epoch=-1):
  net.train(True)
  running_loss = 0.0
  running_regression_loss = 0.0
```

```
  running_classification_loss = 0.0
  for i, data in enumerate(loader):
    images, boxes, labels = data
    images = images.to(device)
    boxes = boxes.to(device)
    labels = labels.to(device)
    optimizer.zero_grad()
    confidence, locations = net(images)
    regression_loss, classification_loss = criterion(confidence, locations, labels, boxes) #
TODO CHANGE BOXES
    loss = regression_loss + classification_loss
    loss.backward()
    optimizer.step()
    running_loss += loss.item()
    running_regression_loss += regression_loss.item()
    running_classification_loss += classification_loss.item()
    if i and i % debug_steps == 0:
      avg_loss = running_loss / debug_steps
      avg_reg_loss = running_regression_loss / debug_steps
      avg_clf_loss = running_classification_loss / debug_steps
      logging.info(
        f"Epoch: {epoch}, Step: {i}, " +
        f"Average Loss: {avg_loss:.4f}, " +
        f"Average Regression Loss {avg_reg_loss:.4f}, " +
        f"Average Classification Loss: {avg_clf_loss:.4f}"
      )
      running_loss = 0.0
      running_regression_loss = 0.0
      running_classification_loss = 0.0
def test(loader, net, criterion, device):
  net.eval()
  running_loss = 0.0
  running_regression_loss = 0.0
```

```
  running_classification_loss = 0.0
  num = 0
  for _, data in enumerate(loader):
    images, boxes, labels = data
    images = images.to(device)
    boxes = boxes.to(device)
    labels = labels.to(device)
    num += 1
    with torch.no_grad():
      confidence, locations = net(images)
      regression_loss, classification_loss = criterion(confidence, locations, labels, boxes)
      loss = regression_loss + classification_loss
    running_loss += loss.item()
    running_regression_loss += regression_loss.item()
    running_classification_loss += classification_loss.item()
  return running_loss / num, running_regression_loss / num, running_classification_loss /
num
if __name__ == '__main__':
  timer = Timer()
  logging.info(args)
  if args.net == 'vgg16-ssd':
    create_net = create_vgg_ssd
    config = vgg_ssd_config
  elif args.net == 'mb1-ssd':
    create_net = create_mobilenetv1_ssd
    config = mobilenetv1_ssd_config
  elif args.net == 'mb1-ssd-lite':
    create_net = create_mobilenetv1_ssd_lite
    config = mobilenetv1_ssd_config
  elif args.net == 'sq-ssd-lite':
    create_net = create_squeezenet_ssd_lite
    config = squeezenet_ssd_config
  elif args.net == 'mb2-ssd-lite':
```

```
    create_net = lambda num: create_mobilenetv2_ssd_lite(num,
width_mult=args.mb2_width_mult)
    config = mobilenetv1_ssd_config
  elif args.net == 'mb3-large-ssd-lite':
    create_net = lambda num: create_mobilenetv3_large_ssd_lite(num)
    config = mobilenetv1_ssd_config
  elif args.net == 'mb3-small-ssd-lite':
    create_net = lambda num: create_mobilenetv3_small_ssd_lite(num)
    config = mobilenetv1_ssd_config
  else:
    logging.fatal("The net type is wrong.")
    parser.print_help(sys.stderr)
    sys.exit(1)
  train_transform = TrainAugmentation(config.image_size, config.image_mean,
config.image_std)
  target_transform = MatchPrior(config.priors, config.center_variance,
                config.size_variance, 0.5)
  test_transform = TestTransform(config.image_size, config.image_mean,
config.image_std)
  logging.info("Prepare training datasets.")
  datasets = []
  for dataset_path in args.datasets:
    if args.dataset_type == 'voc':
      dataset = VOCDataset(dataset_path, transform=train_transform,
                target_transform=target_transform)
      label_file = os.path.join(args.checkpoint_folder, "voc-model-labels.txt")
      store_labels(label_file, dataset.class_names)
      num_classes = len(dataset.class_names)
    elif args.dataset_type == 'open_images':
      dataset = OpenImagesDataset(dataset_path,
        transform=train_transform, target_transform=target_transform,
        dataset_type="train", balance_data=args.balance_data)
      label_file = os.path.join(args.checkpoint_folder, "open-images-model-labels.txt")
```

```
    store_labels(label_file, dataset.class_names)
    logging.info(dataset)
    num_classes = len(dataset.class_names)
  else:
    raise ValueError(f"Dataset type {args.dataset_type} is not supported.")
  datasets.append(dataset)
logging.info(f"Stored labels into file {label_file}.")
train_dataset = ConcatDataset(datasets)
logging.info("Train dataset size: {}".format(len(train_dataset)))
train_loader = DataLoader(train_dataset, args.batch_size,
          num_workers=args.num_workers,
          shuffle=True)
logging.info("Prepare Validation datasets.")
if args.dataset_type == "voc":
  val_dataset = VOCDataset(args.validation_dataset, transform=test_transform,
           target_transform=target_transform, is_test=True)
elif args.dataset_type == 'open_images':
  val_dataset = OpenImagesDataset(dataset_path,
             transform=test_transform, target_transform=target_transform,
             dataset_type="test")
  logging.info(val_dataset)
logging.info("validation dataset size: {}".format(len(val_dataset)))
val_loader = DataLoader(val_dataset, args.batch_size,
          num_workers=args.num_workers,
          shuffle=False)
logging.info("Build network.")
net = create_net(num_classes)
min_loss = -10000.0
last_epoch = -1
base_net_lr = args.base_net_lr if args.base_net_lr is not None else args.lr
extra_layers_lr = args.extra_layers_lr if args.extra_layers_lr is not None else args.lr
if args.freeze_base_net:
  logging.info("Freeze base net.")
```

```
    freeze_net_layers(net.base_net)
    params = itertools.chain(net.source_layer_add_ons.parameters(),
net.extras.parameters(),
                net.regression_headers.parameters(), net.classification_headers.parameters())
    params = [
      {'params': itertools.chain(
        net.source_layer_add_ons.parameters(),
        net.extras.parameters()
      ), 'lr': extra_layers_lr},
      {'params': itertools.chain(
        net.regression_headers.parameters(),
        net.classification_headers.parameters()
      )}
    ]
  elif args.freeze_net:
    freeze_net_layers(net.base_net)
    freeze_net_layers(net.source_layer_add_ons)
    freeze_net_layers(net.extras)
    params = itertools.chain(net.regression_headers.parameters(),
net.classification_headers.parameters())
    logging.info("Freeze all the layers except prediction heads.")
  else:
    params = [
      {'params': net.base_net.parameters(), 'lr': base_net_lr},
      {'params': itertools.chain(
        net.source_layer_add_ons.parameters(),
        net.extras.parameters()
      ), 'lr': extra_layers_lr},
      {'params': itertools.chain(
        net.regression_headers.parameters(),
        net.classification_headers.parameters()
      )}
    ]
```

```
timer.start("Load Model")
if args.resume:
  logging.info(f"Resume from the model {args.resume}")
  net.load(args.resume)
elif args.base_net:
  logging.info(f"Init from base net {args.base_net}")
  net.init_from_base_net(args.base_net)
elif args.pretrained_ssd:
  logging.info(f"Init from pretrained ssd {args.pretrained_ssd}")
  net.init_from_pretrained_ssd(args.pretrained_ssd)
logging.info(f'Took {timer.end("Load Model"):.2f} seconds to load the model.')
net.to(DEVICE)
criterion = MultiboxLoss(config.priors, iou_threshold=0.5, neg_pos_ratio=3,
            center_variance=0.1, size_variance=0.2, device=DEVICE)
optimizer = torch.optim.SGD(params, lr=args.lr, momentum=args.momentum,
             weight_decay=args.weight_decay)
logging.info(f"Learning rate: {args.lr}, Base net learning rate: {base_net_lr}, "
      + f"Extra Layers learning rate: {extra_layers_lr}.")
if args.scheduler == 'multi-step':
  logging.info("Uses MultiStepLR scheduler.")
  milestones = [int(v.strip()) for v in args.milestones.split(",")]
  scheduler = MultiStepLR(optimizer, milestones=milestones,
                          gamma=0.1, last_epoch=last_epoch)
elif args.scheduler == 'cosine':
  logging.info("Uses CosineAnnealingLR scheduler.")
  scheduler = CosineAnnealingLR(optimizer, args.t_max, last_epoch=last_epoch)
else:
  logging.fatal(f"Unsupported Scheduler: {args.scheduler}.")
  parser.print_help(sys.stderr)
  sys.exit(1)
logging.info(f"Start training from epoch {last_epoch + 1}.")
for epoch in range(last_epoch + 1, args.num_epochs):
```

```
    scheduler.step()
    train(train_loader, net, criterion, optimizer,
          device=DEVICE, debug_steps=args.debug_steps, epoch=epoch)

    if epoch % args.validation_epochs == 0 or epoch == args.num_epochs - 1:
        val_loss, val_regression_loss, val_classification_loss = test(val_loader, net, criterion,
DEVICE)
        logging.info(
            f"Epoch: {epoch}, " +
            f"Validation Loss: {val_loss:.4f}, " +
            f"Validation Regression Loss {val_regression_loss:.4f}, " +
            f"Validation Classification Loss: {val_classification_loss:.4f}"
        )
        model_path = os.path.join(args.checkpoint_folder, f"{args.net}-Epoch-{epoch}-Loss-
{val_loss}.pth")
        net.save(model_path)
        logging.info(f"Saved model {model_path}")
```

Anexo 4: Script de ejecución del algoritmo

```
from vision.ssd.vgg_ssd import create_vgg_ssd, create_vgg_ssd_predictor
from vision.ssd.mobilenetv1_ssd import create_mobilenetv1_ssd, create_mobilenetv1_ssd_predictor
from vision.ssd.mobilenetv1_ssd_lite import create_mobilenetv1_ssd_lite, create_mobilenetv1_ssd_lite_predictor
from vision.ssd.squeezenet_ssd_lite import create_squeezenet_ssd_lite, create_squeezenet_ssd_lite_predictor
from vision.ssd.mobilenet_v2_ssd_lite import create_mobilenetv2_ssd_lite, create_mobilenetv2_ssd_lite_predictor
from vision.ssd.mobilenetv3_ssd_lite import create_mobilenetv3_large_ssd_lite, create_mobilenetv3_small_ssd_lite
from vision.utils.misc import Timer
import cv2
import sys
#import serial

if len(sys.argv) < 5:
  print('Usage: python run_ssd_example.py <net type> <model path> <label path> [video file]')
  sys.exit(0)
net_type = sys.argv[1]
model_path = sys.argv[2]
label_path = sys.argv[3]
threshold= sys.argv[4]

if len(sys.argv) >= 6:
  cap = cv2.VideoCapture(sys.argv[4]) # capture from file
else:
  #cap = cv2.VideoCapture(0)
  cap = cv2.VideoCapture('/home/user/Documents/Dany/pytorch-ssd/Data/video/video.mp4')  # capture from camera
  cap.set(3, 1920)
```

```
  cap.set(4, 1080)

class_names = [name.strip() for name in open(label_path).readlines()]
num_classes = len(class_names)



if net_type == 'vgg16-ssd':
  net = create_vgg_ssd(len(class_names), is_test=True)
elif net_type == 'mb1-ssd':
  net = create_mobilenetv1_ssd(len(class_names), is_test=True)
elif net_type == 'mb1-ssd-lite':
  net = create_mobilenetv1_ssd_lite(len(class_names), is_test=True)
elif net_type == 'mb2-ssd-lite':
  net = create_mobilenetv2_ssd_lite(len(class_names), is_test=True)
elif net_type == 'mb3-large-ssd-lite':
  net = create_mobilenetv3_large_ssd_lite(len(class_names), is_test=True)
elif net_type == 'mb3-small-ssd-lite':
  net = create_mobilenetv3_small_ssd_lite(len(class_names), is_test=True)
elif net_type == 'sq-ssd-lite':
  net = create_squeezenet_ssd_lite(len(class_names), is_test=True)
else:
  print("The net type is wrong. It should be one of vgg16-ssd, mb1-ssd and mb1-ssd-lite.")
  sys.exit(1)
net.load(model_path)

if net_type == 'vgg16-ssd':
  predictor = create_vgg_ssd_predictor(net, candidate_size=200)
elif net_type == 'mb1-ssd':
  predictor = create_mobilenetv1_ssd_predictor(net, candidate_size=200)
elif net_type == 'mb1-ssd-lite':
  predictor = create_mobilenetv1_ssd_lite_predictor(net, candidate_size=200)
elif net_type == 'mb2-ssd-lite' or net_type == "mb3-large-ssd-lite" or net_type == "mb3-
small-ssd-lite":
```

```
  predictor = create_mobilenetv2_ssd_lite_predictor(net, candidate_size=200)
elif net_type == 'sq-ssd-lite':
  predictor = create_squeezenet_ssd_lite_predictor(net, candidate_size=200)
else:
  print("The net type is wrong. It should be one of vgg16-ssd, mb1-ssd and mb1-ssd-lite.")
  sys.exit(1)
timer = Timer()
while True:
  ret, orig_image = cap.read()
  if orig_image is None:
    continue
  image = cv2.cvtColor(orig_image, cv2.COLOR_BGR2RGB)
  timer.start()
  boxes, labels, probs = predictor.predict(image, 10, 0.4)
  interval = timer.end()
  print('Time: {:.2f}s, Detect Objects: {:d}.'.format(interval, labels.size(0)))
  for i in range(boxes.size(0)):
    box = boxes[i, :]
    label = f"{class_names[labels[i]]}: {probs[i]:.2f}"
    split_val=label.split(sep=":",maxsplit=2)
    prob = float(split_val[1])
    if(class_names[labels[i]]=='Strawberry' and prob>=float(threshold)):
      cv2.rectangle(orig_image, (box[0], box[1]), (box[2], box[3]), (255, 255, 0), 4)
      y_c= float((abs(box[3]-box[1])/2)+box[1])
      x_c= float((abs(box[0]-box[2])/2)+box[0])
      cv2.circle(orig_image,(int(x_c), int(y_c)),5,(0,200,0),2)
      cv2.putText(orig_image, label, (box[0]+20, box[1]+40),
cv2.FONT_HERSHEY_SIMPLEX,1, # font scale
          (255, 0, 255),
          2) # line type
      #port.write(b"D/n")
  cv2.imshow('annotated', orig_image)
  if cv2.waitKey(1) & 0xFF == ord('q'):
```

```
    break
cap.release()
cv2.destroyAllWindows()
```

2Anexo 5: Tablas de cálculo para determinar los valores de la matriz de confusión

- **Tabla de Abreviaturas**

Abreviaturas	Descripción
TP	Predichos positivos y realmente positivos.
FP	Predichos positivos y en realidad son negativos.
NT	negativos previstos y realmente negativos.
FN	Predicho negativo y en realidad es positivo.

- **Tabla de las 100 imágenes con objeto a detectar**

#	Nombres	Formato	Contiene fresa	TP	FP
1	1	.jpg	si	1	0
2	2	.jpg	si	1	0
3	3	.jpg	si	1	0
4	4	.jpg	si	1	0
5	5	.jpg	si	1	0
6	6	.jpg	si	1	0
7	7	.jpg	si	1	0
8	8	.jpg	si	0	1
9	9	.jpg	si	0	1
10	10	.jpg	si	0	1
11	11	.jpg	si	1	0
12	12	.jpg	si	1	0
13	13	.jpg	si	1	0
14	14	.jpg	si	1	0
15	15	.jpg	si	1	0
16	16	.jpg	si	1	0
17	17	.jpg	si	1	0
18	18	.jpg	si	1	0
19	19	.jpg	si	0	1
20	20	.jpg	si	0	1
21	21	.jpg	si	1	0
22	22	.jpg	si	1	0
23	23	.jpg	si	1	0
24	24	.jpg	si	1	0
25	25	.jpg	si	1	0
26	26	.jpg	si	1	0
27	27	.jpg	si	1	0
28	28	.jpg	si	1	0
29	29	.jpg	si	1	0
30	30	.jpg	si	1	0

#	Nombres	Formato	Contiene fresa	TP	FP
31	31	.jpg	si	1	0
32	32	.jpg	si	1	0
33	33	.jpg	si	1	0
34	34	.jpg	si	1	0
35	35	.jpg	si	0	1
36	36	.jpg	si	0	1
37	37	.jpg	si	0	1
38	38	.jpg	si	0	1
39	39	.jpg	si	0	0
40	40	.jpg	si	0	1
41	41	.jpg	si	1	0
42	42	.jpg	si	1	0
43	43	.jpg	si	1	0
44	44	.jpg	si	1	0
45	45	.jpg	si	1	0
46	46	.jpg	si	1	0
47	47	.jpg	si	1	0
48	48	.jpg	si	0	1
49	49	.jpg	si	1	0
50	50	.jpg	si	1	0
51	51	.jpg	si	1	0
52	52	.jpg	si	1	0
53	53	.jpg	si	1	0
54	54	.jpg	si	1	0
55	55	.jpg	si	1	0
56	56	.jpg	si	1	0
57	57	.jpg	si	1	0
58	58	.jpg	si	1	0
59	59	.jpg	si	1	0
60	60	.jpg	si	1	0
61	61	.jpg	si	1	0
62	62	.jpg	si	1	0
63	63	.jpg	si	1	0
64	64	.jpg	si	1	0
65	65	.jpg	si	1	0
66	66	.jpg	si	1	0
67	67	.jpg	si	1	0
68	68	.jpg	si	1	0
69	69	.jpg	si	1	0
70	70	.jpg	si	1	0
71	71	.jpg	si	1	0
72	72	.jpg	si	1	0

#	Nombres	Formato	Contiene fresa	TP	FP
73	73	.jpg	si	0	1
74	74	.jpg	si	0	1
75	75	.jpg	si	1	0
76	76	.jpg	si	1	0
77	77	.jpg	si	1	0
78	78	.jpg	si	1	0
79	79	.jpg	si	1	0
80	80	.jpg	si	1	0
81	81	.jpg	si	1	0
82	82	.jpg	si	1	0
83	83	.jpg	si	1	0
84	84	.jpg	si	1	0
85	85	.jpg	si	1	0
86	86	.jpg	si	1	0
87	87	.jpg	si	1	0
88	88	.jpg	si	1	0
89	89	.jpg	si	1	0
90	90	.jpg	si	1	0
91	91	.jpg	si	1	0
92	92	.jpg	si	1	0
93	93	.jpg	si	1	0
94	94	.jpg	si	1	0
95	95	.jpg	si	1	0
96	96	.jpg	si	1	0
97	97	.jpg	si	1	0
98	98	.jpg	si	1	0
99	99	.jpg	si	1	0
100	100	.jpg	si	1	0
			Total	86	13

- **Tabla de las 100 imágenes sin objeto a detectar**

#	Nombres	Formato	Contiene fresa	NT	FN
101	101	.jpg	no	1	0
102	102	.jpg	no	1	0
103	103	.jpg	no	1	0
104	104	.jpg	no	1	0
105	105	.jpg	no	1	0
106	106	.jpg	no	1	0
107	107	.jpg	no	1	0
108	108	.jpg	no	1	0

#	Nombres	Formato	Contiene fresa	NT	FN
109	109	.jpg	no	0	1
110	110	.jpg	no	0	1
111	111	.jpg	no	1	0
112	112	.jpg	no	1	0
113	113	.jpg	no	1	0
114	114	.jpg	no	0	1
115	115	.jpg	no	0	1
116	116	.jpg	no	1	0
117	117	.jpg	no	0	1
118	118	.jpg	no	1	0
119	119	.jpg	no	1	0
120	120	.jpg	no	0	1
121	121	.jpg	no	1	0
122	122	.jpg	no	1	0
123	123	.jpg	no	1	0
124	124	.jpg	no	1	0
125	125	.jpg	no	1	0
126	126	.jpg	no	0	1
127	127	.jpg	no	0	1
128	128	.jpg	no	0	1
129	129	.jpg	no	0	1
130	130	.jpg	no	1	0
131	131	.jpg	no	1	0
132	132	.jpg	no	1	0
133	133	.jpg	no	1	0
134	134	.jpg	no	1	0
135	135	.jpg	no	1	0
136	136	.jpg	no	1	0
137	137	.jpg	no	1	0
138	138	.jpg	no	1	0
139	139	.jpg	no	1	0
140	140	.jpg	no	1	0
141	141	.jpg	no	1	0
142	142	.jpg	no	1	0
143	143	.jpg	no	1	0
144	144	.jpg	no	1	0
145	145	.jpg	no	0	1
146	146	.jpg	no	1	0
147	147	.jpg	no	1	0
148	148	.jpg	no	0	1
149	149	.jpg	no	1	0
150	150	.jpg	no	1	0

#	Nombres	Formato	Contiene fresa	NT	FN
151	151	.jpg	no	1	0
152	152	.jpg	no	1	0
153	153	.jpg	no	1	0
154	154	.jpg	no	1	0
155	155	.jpg	no	1	0
156	156	.jpg	no	1	0
157	157	.jpg	no	1	0
158	158	.jpg	no	1	0
159	159	.jpg	no	1	0
160	160	.jpg	no	1	0
161	161	.jpg	no	1	0
162	162	.jpg	no	1	0
163	163	.jpg	no	1	0
164	164	.jpg	no	1	0
165	165	.jpg	no	0	1
166	166	.jpg	no	0	1
167	167	.jpg	no	1	0
168	168	.jpg	no	1	0
169	169	.jpg	no	1	0
170	170	.jpg	no	1	0
171	171	.jpg	no	1	0
172	172	.jpg	no	1	0
173	173	.jpg	no	1	0
174	174	.jpg	no	1	0
175	175	.jpg	no	1	0
176	176	.jpg	no	0	1
177	177	.jpg	no	0	1
178	178	.jpg	no	0	1
179	179	.jpg	no	0	1
180	180	.jpg	no	1	0
181	181	.jpg	no	1	0
182	182	.jpg	no	1	0
183	183	.jpg	no	1	0
184	184	.jpg	no	1	0
185	185	.jpg	no	1	0
186	186	.jpg	no	1	0
187	187	.jpg	no	1	0
188	188	.jpg	no	1	0
189	189	.jpg	no	1	0
190	190	.jpg	no	1	0
191	191	.jpg	no	1	0
192	192	.jpg	no	1	0

#	Nombres	Formato	Contiene fresa	NT	FN
193	193	.jpg	no	0	1
194	194	.jpg	no	0	1
195	195	.jpg	no	0	1
196	196	.jpg	no	1	0
197	197	.jpg	no	1	0
198	198	.jpg	no	1	0
199	199	.jpg	no	1	0
200	200	.jpg	no	1	0
			Total	79	21