UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE CIENCIAS EMPRESARIALES

"DETECCIÓN Y CONTEO AUTOMÁTICO DE ARANDANOS EN SECUENCIAS DE IMÁGENES DENSAS CON TÉCNICAS DE DEEP LEARNING

Tesis para optar al grado de Magíster de Ciencias de la Computación

Cesar Navarrete Ulloa

Profesor Guía

Cristhian Aguilera

Profesor Co-Guía

Pedro Campos

Concepción, Chile

Resumen

Tradicionalmente, el sector agropecuario realiza gran parte de las actividades de forma manual, dentro de estas tareas se incluye el conteo de frutos maduros para la estimación de la cosecha. Esta tarea, fundamental para la optimización logística y la gestión de recursos, requiere de una inversión significativa de tiempo y dinero, considerando la mano de obra a contratar y el tiempo requerido para completar el conteo. Recientemente, el campo del aprendizaje automático ha experimentado avances significativos, aplicándose en diversas áreas para automatizar tareas complejas. Destacándose dentro de estos avances el desarrollo de los modelos de detección de objetos en base a Redes Neuronales Convolucionales (CNNs) y Redes Transformer, los cuales son usados ampliamente en el campo de medicina para tareas como la detección de células cancerígenas, o en el sector de seguridad para la identificación de posibles amenazas, entre otros sectores.

Para el sector agropecuario, la automatización del conteo de frutos, mediante el uso de drones o robots, representa una solución innovadora, permitiendo estimaciones de cosecha con mínima o nula intervención humana. Esta tecnología promete beneficios como la planificación anticipada de la mano de obra y decisiones de mercado más informadas, tales como la fijación de precios, además de optimizar la gestión de la cadena de suministros.

En este trabajo, se propone el desarrollo de tres algoritmos que utilizan el aprendizaje profundo para el conteo de arándanos, diseñados para procesar secuencias de imágenes o videos. El algoritmo aborda desafíos específicos asociados a esta tarea, como la alta densidad de los arándanos, la oclusión que producen las hojas y los movimientos dinámicos de la cámara, que pueden provocar un doble conteo de los frutos. Se realizo además una revisión exhaustiva del estado del arte para determinar los modelos de detección, algoritmos de seguimiento y métodos de conteo apropiados para resolver la problemática.

Para poner a prueba la eficacia de estos algoritmos, se desarrolló un programa de simulación que permite evaluarlos bajo diversas condiciones, como la concentración de arándanos en la pantalla, la diversidad de estos y el nivel de oclusión, entre otros. Gracias a esta herramienta, fue posible llevar a cabo cientos de experimentos que, de otro modo, habrían requerido encontrar plantaciones con condiciones ideales para cada prueba, lo cual resulta inviable en el contexto de este proyecto y supondría una inversión de tiempo y recursos muy elevada. Se pretende compartir este programa con la idea de que futuros investigadores, interesados en el conteo de arándanos de forma automática, puedan utilizarlo.

Las diferencias entre los tres algoritmos propuestos se centran principalmente en el componente de detección. Para ello, se plantearon cinco modificaciones a la arquitectura de YOLOv8, de las cuales se seleccionaron las tres con mayor desempeño para la tarea de conteo de arándanos.

Una vez finalizados los 270 experimentos, se comprobó empíricamente que el algoritmo de conteo que emplea la arquitectura de YOLOv8 con cabezas de detección pequeña, mediana y grande (CD3-CD4-CD5) supera los resultados reportados en la literatura para el conteo de frutos, al obtener un MAPE de 3.86% en el conteo de arándanos. Los otros dos algoritmos propuestos lograron un MAPE de 4.5% y 4.77%, lo que los ubica en un nivel competitivo frente a las soluciones existentes para el conteo de frutas.

Esta tesis demuestra, de manera empírica, la necesidad de adaptar los algoritmos de detección al dominio específico del problema. En este caso, se modificó la capa de entrada de la arquitectura YOLOv8 para optimizar la captura de arándanos, cuyo tamaño diminuto dificulta su detección con la configuración predeterminada. Asimismo, se investigó cómo las cabezas de detección interactúan con el modelo y contribuyen a la detección de objetos pequeños, para lo cual se añadieron o eliminaron distintas combinaciones de dichas cabezas. Finalmente, se comprobó que, pese a tratarse de la detección de objetos pequeños, las cabezas de detección para objetos grandes y medianos aportan información valiosa que mejora la detección de arándanos.

Tabla de contenido

Capítulo 1: Introducción

La producción y exportación de frutas es uno de los principales componentes de la economía chilena. Chile es uno de los principales exportadores de arándanos a nivel mundial, ubicándose en el cuarto lugar[1], luego de China, Perú y Estados Unidos, produciendo en la temporada 2023-2024, 86.000 toneladas de este fruto [2].

Considerando esto, la estimación temprana de la producción de frutas juega un rol fundamental en las decisiones que toma la industria agrícola. En primer lugar, permite la planificación temprana de las fechas de cosecha, procesos de postcosecha, y una estimación del volumen total que debe ser procesado. Permite además planificar tempranamente la distribución los recursos necesarios para la cosecha, incluyendo el personal a contratar, la capacidad de almacenamiento necesaria, el transporte necesario, entre otros. Además, permite al productor estimar el precio a fijar, y las ganancias que generara la cosecha. Por otro lado, permite reducir el riesgo, si se espera, por ejemplo, una baja cosecha, ya que los productores tienen más tiempo para tomar decisiones que puedan mitigar las pérdidas económicas.

En los últimos años el Aprendizaje Profundo ha revolucionado múltiples campos, desde la industria automotriz con el desarrollo de autos automáticos, hasta el campo de la medicina donde algoritmos de Aprendizaje Profundo son utilizados para localizar e identificar enfermedades de manera automática, como la detección de enfermedades respiratorias en radiografías. Particularmente en el campo de la agricultura, el conteo automático de frutas permanece como un tópico de interés[3], [4], pues la correcta estimación temprana de la cosecha se traduce en los beneficios explicados anteriormente. A pesar de esto, el identificar las mejores técnicas, herramientas, y metodologías para cada paso particular, aun es un desafío.

El campo de la detección de imágenes ha experimentado un progreso remarcable en los últimos años. Inicialmente dependiente de técnicas de extracción de características como Scale-invariant feature transform (o SIFT), fue sujeto de un cambio substancial con la aparición de las Redes Neuronales Convolucionales (CNN). A partir de esta surgieron numerosas arquitecturas para la detección de objetos como “Single Shot Detector” (SDD), “Faster Regions with Convolutional Neural Networks” (Faster-RCNN), y más recientemente “You Only Look Once” (YOLO). Estas arquitecturas con el tiempo han mejorado considerablemente su rendimiento y velocidad, y hoy en día juegan un papel fundamental en la tarea de contar frutas de forma automática.

Considerando esto, para esta propuesta de Tesis, se plantea el desarrollo de un algoritmo de conteo de arándanos en imágenes densas el cual tiene como objetivo, contar automáticamente, de forma precisa los arándanos en un video grabado por un dron a pesar de su movimiento dinámico. Esto se realizará a través de la revisión del estado del arte para seleccionar, y probar los métodos y modelos de Aprendizaje Profundo más adecuados para resolver este problema en particular, que tiene como objetivo en un futuro automatizar este proceso en beneficio del sector agrícola.

Capítulo 2: Marco Conceptual

2.1. Campos Pertinentes

2.1.1. Visión por computador

El campo de "visión por computador" tiene como objetivo el permitir a los sistemas computacionales, entender e interpretar información visual del ambiente, con el fin de extraer información relevante y automatizar tareas. La visión por computador se aplica en múltiples campos, como la seguridad, donde permite la vigilancia automática; la medicina, facilitando diagnósticos más precisos mediante la interpretación de imágenes médicas[5] y la industria automotriz, con el desarrollo de vehículos autónomos que utilizan distintas tecnologías de visión por computador para navegar por su entorno[6].

A diferencia de los seres humanos los sistemas computacionales no pueden observar directamente una imagen, en cambio, las interpretan como una matriz de valores numéricos los cuales indican el color de cada píxel en particular[7]. A partir de esta matriz numérica, se aplican diversos algoritmos para procesar y analizar la imagen, lo que permite al sistema computacional "entender" y reaccionar respecto a lo que se encuentra en la imagen. Estos algoritmos pueden incluir desde métodos de detección de bordes y segmentación de objetos hasta técnicas más complejas de reconocimiento de patrones y aprendizaje profundo, los cuales facilitan la identificación y clasificación de elementos dentro de la imagen en fracciones de segundo.

2.1.2. Aprendizaje automático

El aprendizaje automático o "Machine Learning", es un campo del área de Inteligencia Artificial, el cual se enfoca en desarrollar técnicas que permiten a los sistemas computacionales aprender a partir de los datos. En lugar de requerir ser programadas explícitamente para realizar alguna tarea especifica, con el aprendizaje automático, estos sistemas en cambio pueden ser entrenados para realizar esta tarea de forma automática con mínima o nula intervención humana. El entrenamiento es similar al aprendizaje humano, ya que se deben presentar múltiples ejemplos a los sistemas para que aprendan los patrones de los datos.

Existen varios tipos de aprendizaje automático, de los cuales se destacan[8]:

- **Aprendizaje supervisado**: Los sistemas aprenden a partir de un conjunto de datos etiquetados, es decir, datos que ya tienen asignados los resultados correctos. El objetivo es que luego de ser entrenado, el sistema computacional pueda predecir el resultado correcto para nuevos datos.
- **Aprendizaje no supervisado:** En este caso, los sistemas trabajan con datos que no están etiquetados. El sistema intenta aprender la estructura de los datos para agruparlos en categorías o encontrar otras regularidades.

- **Aprendizaje por refuerzo:** Este tipo de aprendizaje se basa en un método de prueba y error, donde el sistema recibe recompensas por comportamientos correctos o penalizaciones por errores. El objetivo es maximizar las recompensas.

2.1.3. Aprendizaje profundo

El aprendizaje profundo es una subárea del Aprendizaje Automático. A diferencia de los modelos de Aprendizaje automático tradicionales, donde era necesario extraer características de forma manual a través de distintas técnicas, este tipo de modelos, en cambio pueden aprender características de forma automática y sin intervención humana. Esto se hace a través del uso de una jerarquía de múltiples capas, en las cuales el modelo transforma los datos en representaciones más abstractas. En las primeras capas estos modelos pueden aprender a capturar características simples como el color de los objetos en una imagen, mientras que, en capas más profundas, se aprenden representaciones complejas como la textura de la imagen.

El Aprendizaje profundo ha demostrado un desempeño considerable en una variedad de tareas, incluyendo el reconocimiento de imágenes, procesamiento de lenguaje natural, predicción de valores, entre otros. Su éxito se debe en gran medida a los avances en el poder computacional (como el uso de la unidad de procesamiento gráfico, abreviada como GPU), grandes volúmenes de datos disponibles para el entrenamiento, y mejoras en algoritmos de optimización.

2.2. Redes Neuronales Artificiales

Inspiradas en la estructura y funcionamiento del circuito neuronal del cerebro, las redes neuronales son un tipo de arquitectura computacional que aprenden a reconocer patrones y a resolver problemas específicos mediante el ajuste de conexiones internas en un proceso de entrenamiento en el cual grandes cantidades de datos relacionados con el problema son alimentadas a la red.

La primera implementación de una Red Neuronal fue hecha por Frank Rosenblatt en 1958 [9] con el denominado "Perceptron". Este modelo se basa en una sola capa de neuronas y fue diseñado inicialmente para realizar tareas de clasificación binaria, y reconocer patrones simples en los datos (ver Ilustración 1). El perceptrón ajusta automáticamente los pesos (weights) de las conexiones en función de los errores cometidos durante el proceso de aprendizaje, utilizando una regla de aprendizaje. El problema principal de este modelo es su limitación de solo ser capaz de resolver problemas linealmente separables. Esta limitación fue superada con el descubrimiento de las redes neuronales multicapa, que incluyen una o más capas ocultas además de la capa de entrada y salida. Estas capas adicionales consisten en neuronas que permiten la representación de funciones no lineales, lo que permite al modelo poder resolver problemas más complejos. Con la incorporación de funciones de activación no lineales, como la función sigmoide, ReLU (Rectified Linear Unit) o tanh, las redes neuronales multicapa (MLP por sus siglas en inglés) pueden aproximar prácticamente cualquier función continua.

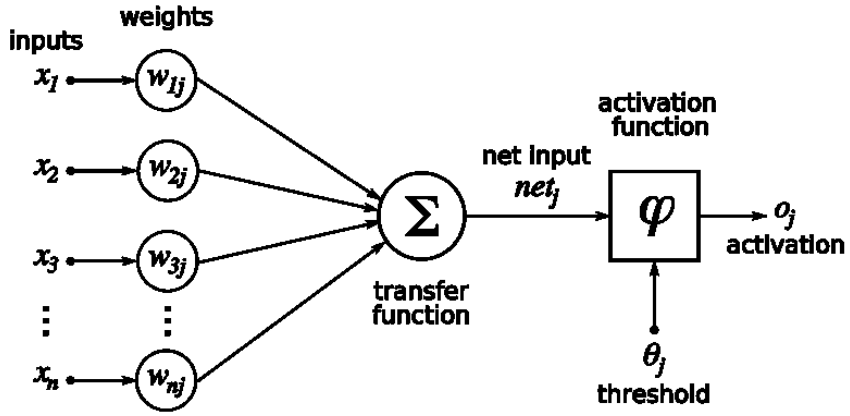

Ilustración 1: Estructura de una Neurona Artificial. Creada por Chrislb (CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=224555)

2.2.1. Estructura General

Una Red Neuronal artificial está compuesta típicamente por los siguientes elementos:

Capa de Entrada

La capa de entrada es la parte inicial de una red neuronal. En esta capa los datos de entrada son ingresados a la red para su posterior procesamiento. Está capa está compuesta de múltiples nodos o neuronas, y cada una de estas representa una característica correspondiente a los datos de entrada. Por ejemplo, para una imagen de 28x28 pixeles en escala de grises, cada neuronal de entrada pude representar el valor de un píxel. Tomando esto en cuenta los datos de entrada deben ser preprocesados de tal manera que sean compatibles con la capa de entrada. Siguiendo con el ejemplo de una imagen, esta no puede ser consumida directamente por la red, por lo que se debe descomponer en un vector de los 784 valores de pixeles, siendo cada valor procesado por una neurona de entrada.

Capas Ocultas

Las capas ocultas se encuentran entre la capa de entrada y la capa de salida de la Red Neuronal. Estas capas son las responsables de la habilidad de la red para aprender tareas complejas, ya que pueden extraer y aprender desde patrones generales a más específicos. Las capas ocultas están compuestas de neuronas las cuales reciben las salidas de neuronas en la capa anterior como entrada, luego calculan una suma ponderada de las entradas, y aplican una función de activación para introducir no linealidad en la red, lo que permite a la Red Neuronal aprender patrones complejos no necesariamente lineales. Lo cual permite expandir el número de tareas que es posible resolver a comparación del Perceptrón de una capa original el cual solo podía resolver problemas linealmente separables. El proceso en el cual las entradas se propagan desde la capa de entrada a la de salida se conoce como “Feed Forward”. A mayor número de capas, mayor profundidad de la red.

Pesos y Sesgos

Como se explicó anteriormente la entrada de una neurona es la suma ponderada de las salidas de todas las neuronas que la preceden en la capa anterior. Dicho esto, el peso (Weight) puede ser pensado como la conexión entre 2 neuronas consecutivas, el valor del peso representa cuanta influencia tiene la neurona anterior en el cálculo de la suma ponderada de la neurona actual. El sesgo (Bias) en cambio permite disparar una neurona independiente si la suma ponderada de los pesos es igual a cero.

Funciones de activación

Son ecuaciones matemáticas que modifican la salida de una neurona. Estas cumplen una función crucial en la habilidad de la red para aprender patrones complejos y resolver problemas difíciles, ya que la mayoría introducen la "no linealidad" en el sistema, sin ellas la red neuronal solo podría resolver problemas linealmente separables.

Algunas de las funciones más populares son:

Función Lineal: Es una de las funciones de activación más simples. Su salida es igual al valor total de sus entradas (ver Ec. 1). Es utilizada en la capa de salida en problemas de regresión ya que se espera que la Red Neuronal pueda predecir un amplio rango de valores reales.

$$f(x) = x \qquad \text{(Ec. 1)}$$

Función Sigmoide: Transforma la salida de una neurona en un valor entre el rango 0 y 1 (ver Ec. 2). Se ocupa generalmente en problemas donde es necesario predecir probabilidades dentro de este rango, como problemas de clasificación binaria en donde comúnmente una probabilidad mayor a 0.5 indica una clase positiva y menor a 0.5 una clase negativa.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \qquad \text{(Ec. 2)}$$

Función Tahn: Transforma las salidas de las neuronas en un valor en un rango de -1 a 1 (ver Ec. 3). La ventaja de esta función es que los valores muy negativos serán convertidos a valores cercanos a -1 mientras que los muy positivos a valores cercanos a +1. Se prefiere por sobre la función Sigmoide como función de activación en las capas ocultas ya que permite a la Red Neuronal converger más rápidamente.

$$nh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad \text{(Ec. 3)}$$

Función ReLU (Rectified Linear Unit): Esta función convierte cualquier salida negativa en 0, mientras que conserva el valor original si este es positivo (ver Ec. 4). Es una de las funciones de activación más populares en la actualidad ya que permite una rápida convergencia durante el entrenamiento de la Red Neuronal y estas además exhiben un mejor desempeño.

$$f(x) = \max(0, x) \qquad \text{(Ec. 4)}$$

Función Leaky Relu: Similar a la función de activación ReLU, con la diferencia que valores negativos, ya no son transformados a 0, en cambio son multiplicados por una pequeña constante α (ver Ec. 5). Esta función evita el problema fundamental de ReLU en donde una neurona con una salida constantemente negativa siempre será convertida en 0 y nunca contribuirá a la red por lo que es considerada como una neurona "muerta". Este tipo de neuronas son problemáticas ya que reducen la capacidad de aprendizaje del modelo.

$$f(x) = \max(\alpha x, x) \quad \text{(Ec. 5)}$$

Capa de Salida

Es la capa final de la Red Neuronal la cual recibe como entrada las salidas de la capa que la precede y transforma este valor dependiendo la tarea que realiza la Red Neuronal. Por ejemplo, para un problema de clasificación binario se aplicará una función Sigmoide sobre este valor, en cambio para un problema de regresión se utilizaría una función de activación lineal.

2.3. Redes Neuronales Convolucionales

Son un tipo de Red Neuronal Profunda especialmente diseñadas para procesar datos que tienen una estructura de cuadrícula, como imágenes. Estas redes son muy utilizadas en el campo de la visión por computadora para tareas como la clasificación de imágenes, la detección de objetos y la segmentación de imágenes. Los origines de estas redes datan a 1959 cuando David .H Hubel y Torsten Wiesel en uno de sus experimentos insertaron un microelectrodo en el cortex visual primario de un gato para luego proyectar patrones de luz[10]. En este experimento descubrieron que algunas neuronas se volvían activas cuando la luz era proyectada como un haz de luz en una dirección especifica, mientras que al rotar o cambiar de ángulo del haz de luz, otras neuronas disparaban en cambio. En el estudio que ambos autores publicaron en 1968 denominaron a las neuronas que solo responden a un estímulo de luz en una posición especifica y orientación especifica “células simples”. Mientras que neuronas que responden a un estímulo dependiendo su orientación, pero sin importar su posición “células complejas”[11]. Hubel y Wiesel propusieron un modelo de este tipo de células que puede ser utilizado para tarea de reconocimiento de patrones.

Uno de los elementos fundamentales de esta red es la capa convolucional la cual consiste en múltiples filtros. Cuando se procesa una imagen en esta capa los filtros se aplican sobre la imagen en un proceso de convolución. Estos filtros son capaces de detectar características como bordes o curvas[12]. En general una Red Neuronal Convolucional consiste en múltiples capas de convolución las cuales pueden extraer características de distinta complejidad dependiendo la profundidad en la que se encuentran. Las primeras capas pueden extraer características simples como el color, mientras que las capas más profundas son capaces de extraer características relevantes complejas como la textura de una imagen. El resultado de aplicar estos filtros sobre la imagen se conoce como un mapa de características. Generalmente se aplica una función de activación ReLU luego de realizar las convoluciones.

Un segundo elemento importante son las capas de agrupación las cuales se encuentran entre las capas convolucionales. Su función principal es la realizar un proceso de “down sampling” o en otras palabras reducir el tamaño de los mapas de características que son la salida generada por la capa convolucional anterior. Esto se hace con el objetivo de reducir el número de parámetros totales de la red disminuyendo el riesgo de “overfitting” o sobreajuste, es decir, que la red se ajuste en exceso a los datos de entrenamiento, perdiendo capacidad de generalizar sus respuestas ante nuevos datos. Esto también reduce la carga computacional del sistema, y además permite que la Red Neuronal sea invariante a pequeñas translaciones. Una forma común de agrupación es “Max Pooling” con un tamaño de filtro de 2x2 y un paso (stride) de 2 pixeles. Considérese un mapa de características de 16x16. Max Pooling tomará los primeros 4

pixeles en la posición (0, 0) a (2,2) y solo conservará el mayor valor. Luego el filtro será aplicado de la posición (2, 0) a (4, 2). Una vez terminado el proceso terminaremos con una matriz de 4x4, habiendo descartado el 75% de las activaciones.

Por último, es necesario señalar la capa de aplanado (flatten). En esta capa todos los mapas de características resultantes serán convertidos en un único vector de características el cual será alimentado a la Red Neuronal Totalmente Conectada que la prosigue.

2.3.1. AlexNet

Es una de las arquitecturas pioneras de Red Neuronal Convolucional en demostrar el poder de los modelos de aprendizaje profundo, creada en el año 2012 por Alex Krizhevsky en colaboración con Ilya Sutskever y Geoffrey Hinton [13]. Lo que destacó a AlexNet de otras arquitecturas de la época fue su profundidad (ver Ilustración 2), que le permite aprender patrones y características más complejos, lo que resultó en un mejor desempeño para la tarea de clasificación de imágenes. Igualmente, para evitar el problema del "Overfitting" dado sus 60 millones de parámetros, se utilizaron 2 técnicas de "Data Augmentation", técnica que consiste en incrementar la diversidad y el tamaño de un conjunto de datos sin la necesidad de recolectar datos reales adicionales, esto con el fin de obtener un modelo que pueda generalizar de forma efectiva. Igualmente aplicaron la técnica de "Dropout", utilizada para prevenir el "overfitting" durante el entrenamiento asignándole a cada neurona una probabilidad de no contribuir al entrenamiento, esto con el fin de no depender únicamente de neuronas específicas. Para este estudio se le asigno a cada neurona una probabilidad del 50% de apagarse. La arquitectura de esta red puede ser vista en la ilustración 2.

Cabe señalar que esta arquitectura fue una de las primeras en demostrar el potencial de utilizar la Unidad de procesamiento gráfico (GPU) para acelerar el entrenamiento de la red. AlexNet fue revolucionaria en su momento obteniendo un error top-5[14] del 15.3% en la competencia "ImageNet Large Scale Visual Recognition Challenge"[15] del año 2012, lo que representa un error un 10.8% más bajo que el segundo lugar.

Como puede observarse en la ilustración 2, la arquitectura consiste en 5 capas convolucionales, las cuales extraen características visuales de una imagen inicial de dimensiones 224x224. Las capas iniciales extraen características como el contorno, o la textura, mientras que las capas mas profundas pueden extraer características más complejas como forma de los objetos. Las 3 capas siguientes actúan como un perceptrón multicapa con una función softmax de activación. Son estas capas las que tienen la función de finalmente clasificar la imagen de entrada.

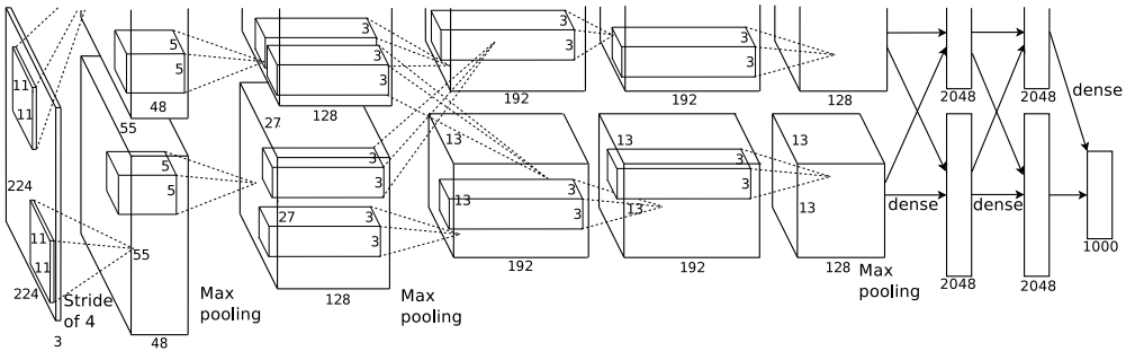

Ilustración 2: Diagrama de la arquitectura de un modelo VGG16. Figura tomada del estudio "ImageNet Classification with Deep Convolutional Neural Networks" [5].

2.3.2 Resnet

La Red Neuronal Residual, también conocida como ResNet, fue introducida en 2015 por Kaiming He y sus colaboradores [16]. ResNet significó un hito importante en el campo del aprendizaje profundo, especialmente en el área de las redes neuronales convolucionales (CNN). Su diseño estaba enfocado en abordar un problema clave conocido como el problema de la degradación.

A medida que las CNNs se hacen más profundas, es decir, cuando se aumenta el número de capas, se esperaba que las redes tuvieran una mayor capacidad para aprender patrones más complejos y detallados. Sin embargo, en la práctica, se observaba lo contrario, el rendimiento de estas redes comenzaba a degradarse después de alcanzar una cierta profundidad. El culpable de esto es un problema de propagación de gradientes conocido como desvanecimiento del gradiente, donde los gradientes se vuelven muy pequeños, haciendo que el aprendizaje sea muy lento o se estanque.

ResNet soluciona este problema introduciendo una arquitectura con "conexiones de salto" que permiten que las señales se transmitan directamente de una capa a otra más adelante en la red. En otras palabras, las conexiones de salto actúan como atajos o caminos alternativos que ayudan a evitar los cuellos de botella en el flujo de gradientes.

La introducción de ResNet fue significativa, permitiendo el desarrollo de modelos con cientos de capas, mientras se conserva o incluso se mejora el rendimiento en tareas de visión por computadora.

Como puede observarse en la ilustración 3, la arquitectura de ResNet, a partir de una imagen de 224×224, primero aplica una capa convolucional de 7×7 seguida de una operación "max pooling" de 3×3 para reducir la resolución. Luego, se organiza en cuatro "etapas residuales" compuestas de bloques con convoluciones y conexiones de salto, lo que permite un mejor flujo de gradientes.

En las primeras capas se extraen características simples (bordes, texturas), mientras que en las más profundas se capturan patrones complejos. Finalmente, una capa de "global average

pooling", la cual permite representar la última capa convolucional como un vector de una dimensión, y una capa totalmente conectada con softmax se encargan de la clasificación final de la imagen.

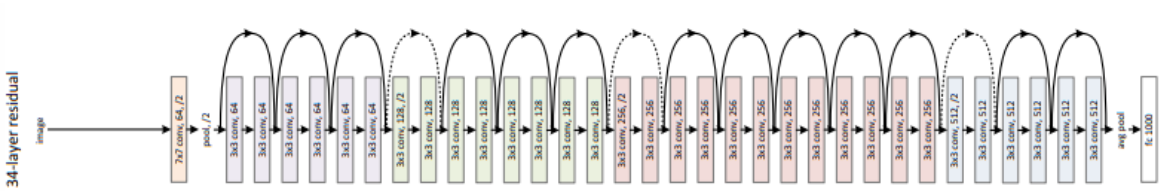

Ilustración 3:Diagrama de la arquitectura de un modelo ResNet34. Figura tomada del estudio "Deep Residual Learning for Image Recognition" [6].

2.4. Detección de objetos

La detección de objetos es una de las tareas claves en el campo de visión por computadora y aprendizaje profundo. Mientras que las redes discutidas en la sección anterior se enfocan en clasificar una imagen completa asignándole una etiqueta de alguna clase, por ejemplo "Gato", la tarea de detección de objetos en cambio tiene como objetivo localizar múltiples objetos en una imagen, y además clasificar cada uno de estos. Las propuestas de localización y clase de un objeto se conocen como "detecciones". Estas detecciones son generalmente entregadas en un formato conocido como "bounding box", el cual es un vector de 4 coordenadas las cuales engloban el objeto detectado en la imagen.

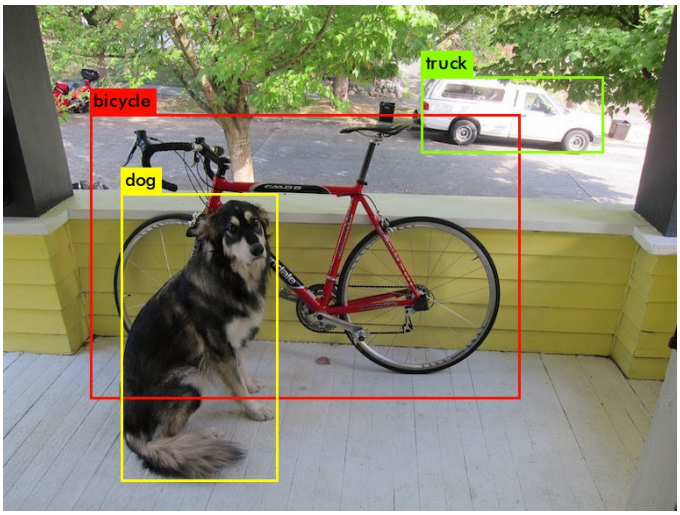

Ilustración 4:Ejemplo de detección de objetos con un modelo YOLO. Imagen tomada del estudio "You Only Look Once: Unified, Real-Time Object Detection" [14].

En el pasado, aplicaciones como el detector de rostros de Viola-Jones [17] dependían de una combinación de clasificadores simples y vectores de características creados manualmente, además de un detector de blobs, y una ventana deslizante para encontrar los objetos. Hoy en día, esto ya no es necesario debido a los avances en redes neuronales convolucionales (CNN) y arquitecturas de detección más sofisticadas como YOLO (You Only Look Once) y SSD (Single Shot MultiBox Detector), que permiten la detección de objetos en tiempo real con alta precisión y eficiencia. Estas técnicas modernas integran tanto la localización como la clasificación en un solo proceso, eliminando la necesidad de características manuales y procedimientos complicados, lo que simplifica y acelera considerablemente la tarea de detección de objetos.

Los detectores de objetos son utilizados ampliamente hoy en día en una multitud de aplicaciones como, aplicaciones de vigilancia en las cuales se detectan personas o aplicaciones de control de tráfico donde identificar individualmente cada vehículo es fundamental.

Los detectores de objetos modernos basados en Redes Neuronales Convoluciones se pueden dividir en 2 grupos según como se procesa una imagen: Detectores de 1 etapa y Detectores de 2 etapas.

Red de Pirámides de Características (Feature Pyramid Networks)

Las Feature Pyramid Networks (FPN), presentadas en 2016 por Tsung-Yi Lin et al. [8], surgieron para mejorar la detección de objetos al enriquecer la forma en que se extraen las características en una red neuronal convolucional. Estas redes aprovechan la naturaleza multiescalar de las CNN, donde una imagen se reduce gradualmente a representaciones cada vez más pequeñas conforme avanza por las capas de la red.

En las primeras capas (con mayor resolución), la red es capaz de captar detalles finos, útiles para detectar objetos pequeños. En las capas finales, la representación es más reducida, por lo que se enfoca en objetos de gran tamaño y en la comprensión del fondo, lo que facilita distinguir el objeto de interés. A cada una de estas salidas se les suele denotar como C2, C3, C4, C5 (la nomenclatura puede variar según la arquitectura).

Tradicionalmente, el camino de la imagen en la red se describe como “bottom-up”, porque pasa de capas de resolución alta a capas de resolución baja. La innovación de FPN consiste en introducir un camino “top-down”, en el cual se toma la salida de la capa más profunda (por ejemplo, C5), se redimensiona, normalmente mediante una operación de upsampling, la cual consiste en aumentar la resolución de un mapa de características a partir de un mapa de menor resolución, para ajustarla al tamaño de la salida previa (C4) y luego se combina con esta para formar un nuevo mapa de características, denominado P4. Este proceso se repite en cada nivel, generando P3, P2, etc.

La ventaja de esta fusión es que características globales (por ejemplo, las que ayudan a separar el fondo del objeto) pueden transferirse a niveles donde se detectan objetos más pequeños. Así, la red enriquece todos los mapas de características (P2, P3, P4, P5), logrando una mejor descripción multi-escala para la detección de objetos.

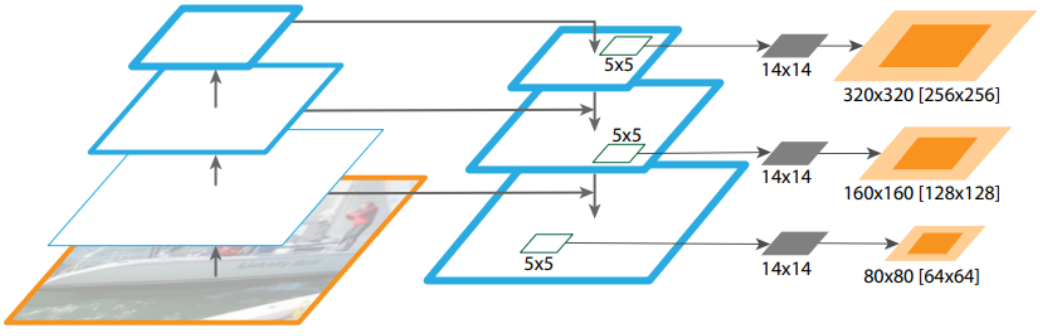

Ilustración 5: Estructura de una FPN. En la ilustración es posible apreciar la formación clásica del mapa de características en una red, seguido por la nueva ruta "top-down" que agrega FPN. Figura tomada del estudio “Feature Pyramid Networks for Object Detection” [18]

Red de Agregación de Rutas (Path Aggregation Network)

La red de agregación de rutas (PANet), presentada en 2018 por Shu Liu et al. [19], se basa en la estructura de FPN pero añade una ruta “bottom-up” que complementa la “top-down”. Después de obtener los mapas P2 a P5, se realiza un downsampling de P2 para igualar la resolución de P3 y fusionarlos, generando N3; luego se repite el proceso hasta llegar a N5, lo que facilita que características de alta resolución, como bordes de objetos, se transmitan a niveles superiores y mejoren la representación global.

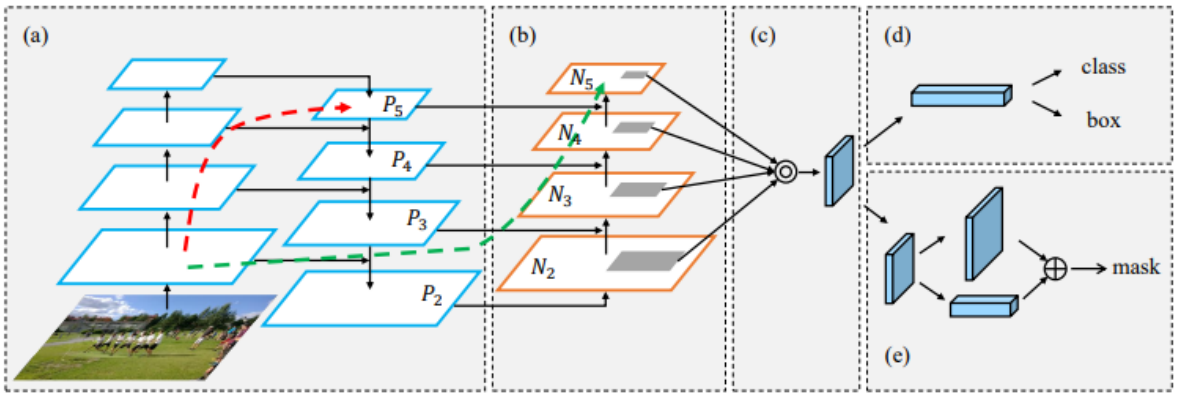

Ilustración 6: Estructura de PANet. La ilustración (a) es la estructura básica de una FPN. La ilustración (b) denota la nueva conexión "bottom-up" que es agregada a la red. Figura tomada del estudio “Path Aggregation Network for Instance Segmentation” [19]

2.4.1. Detectores de 2 etapas (Two-Shot Detectors)

Este tipo de detectores se caracterizan por tener 2 etapas, y en general logran una mayor precisión en la detección de objetos en comparación con los detectores de una sola etapa. A pesar de esta ventaja, su tiempo de ejecución es mucho mayor a los detectores de una etapa.

El objetivo de la primera etapa es generar la propuesta de regiones, en otras palabras, donde tentativamente se cree que se encuentran los objetos. Estas propuestas de regiones son áreas dentro de la imagen en donde existe una alta probabilidad de que contengan un objeto de interés. En esta etapa, se utilizan algoritmos como Region Proposal Network (RPN), o similares, que generan una serie de posibles "bounding boxes".

La segunda etapa se encarga de clasificar estas regiones propuestas y ajustar los "bounding boxes" para una mayor precisión. En esta etapa, cada región propuesta se extrae y se pasa a través de una red neuronal convolucional (CNN) que evalúa si realmente contiene un objeto y, si esto es así, de qué objeto se trata. Además, se ajustan las "bounding boxes" de la etapa anterior para que se encierren mejor al objeto detectado.

Redes R-CNN

Regions with Convolutional Neural Networks (R-CNN) es un detector de 2 pasos introducido por Ross Girshick y sus colaboradores en el año 2013 [20]. Dada una imagen de entrada el modelo aplica un mecanismo conocido como "Búsqueda Selectiva" (Selective Search) el cual es utilizado para extraer las Regiones de Interés (ROI) en donde el algoritmo cree que pueden existir objetos. Dependiendo el caso, la red puede generar hasta 2000 propuestas de regiones de interés. El siguiente paso es procesar cada una de estas propuestas por una CNN para extraer sus vectores de características. Finalmente, un grupo de modelos de clasificación de Super Vector Machine (SVM) son utilizados para determinar qué tipo de objeto existe en la región propuesta, y si es que realmente existe alguno.

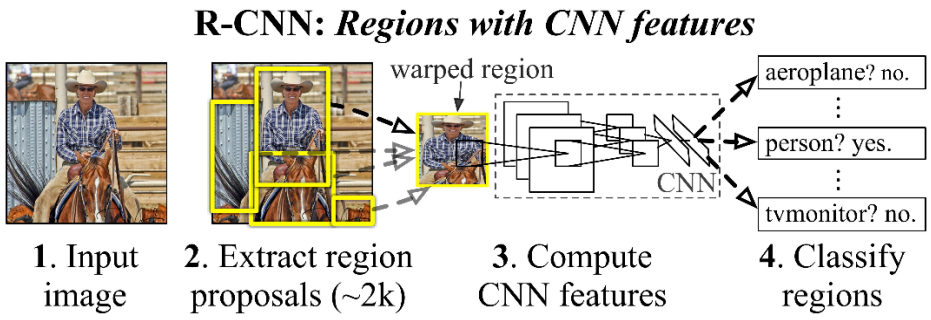

Ilustración 7: Esquema del procesamiento y detección de imágenes con R-CNN. Figura tomada del estudio "Rich feature hierarchies for accurate object detection and semantic segmentation" [20].

R-CNN cuenta con versiones posteriores las cuales presentan mejoras significativas en términos de precisión y eficiencia:

Fast R-CNN: Introducida por Ross Girshick en 2015 [21], Fast R-CNN optimiza el proceso integrando la detección de regiones y la clasificación en una única red neuronal. En lugar de aplicar una CNN a cada región de interés por separado, Fast R-CNN aplica la CNN a la imagen completa y luego utiliza una técnica llamada "ROI Pooling" para extraer características de las regiones propuestas. Esto reduce considerablemente el tiempo de entrenamiento y de inferencia.

Faster R-CNN: Propuesta por Shaoqing y sus colaboradores en el año 2016 [22], Faster R-CNN introduce una Red de Propuestas de Regiones (RPN), que reemplaza el método de Búsqueda Selectiva con una red neuronal que genera las regiones de interés de manera mucho más eficiente. La RPN se entrena junto con el resto de la red, lo que mejora tanto la velocidad como la precisión de la detección.

2.4.2. Detectores de 1 etapa (Single-Shot Detectors)

Los detectores de 1 etapa combinan la etapa de localización y clasificación en una sola etapa, y a diferencia de los detectores de 2 etapas, solo necesitan procesar la imagen una única vez, lo que permite que tengan un tiempo de inferencia mucho más rápido en comparación con estos. Esta característica ha permitido que se masifique su uso en aplicaciones que necesitan operar en tiempo real, como la vigilancia, la conducción autónoma y los dispositivos móviles. Sin embargo, a pesar de esta ventaja, tradicionalmente este tipo de detectores son generalmente menos precisos en comparación y tienen dificultades para detectar objetos pequeños.

SSD (Single-Shot Multibox Detector)

La arquitectura Red Neuronal Convolucional de 1 etapa, “SSD” fue introducida el 2015 por Wei Liu y sus colaboradores[23]. Este modelo representó un avance importante debido a su desempeño y precisión para procesar imágenes en tiempo real para detectar objetos de múltiples tamaños, convirtiéndolo rápidamente en una arquitectura popular.

Esta red combina las tareas de localización y clasificación en una sola etapa, eliminando la necesidad de ejecutar una etapa adicional para generar la propuesta de las regiones de interés como en el caso del detector de 2 etapas “R-CNN”. La arquitectura SSD mejora su velocidad de inferencia y su precisión para encontrar objetos, creando múltiples mapas de características para objetos de múltiples tamaños, y luego sobre estos mapas, aplica una serie de “bounding boxes” ya predefinidas con tamaños fijos conocidos como “anchor boxes”. Estas “anchor boxes” permiten al modelo adaptarse mejor a las diferentes escalas y aspectos de los objetos en las imágenes. Finalmente, este modelo introduce funciones de perdida para la clasificación y la localización de los objetos lo que permite al modelo refinar ambos durante el entrenamiento.

YOLO (You Only Look Once)

La arquitectura de YOLO fue presentada en 2015 por Joseph Redmon y sus colaboradores [24]. Desde su lanzamiento, YOLO y sus diversas versiones han ganado rápidamente popularidad, posicionándose como uno de los detectores de objetos más utilizados en la actualidad. Esto se debe a su combinación de tiempos de inferencia rápidos, que permiten su aplicación en tareas en tiempo real, y su notable desempeño en la detección de objetos.

Cuando una imagen es procesada por la red, está la divide en una grilla de celdas de tamaño SxS. Cada una de estas celdas es responsable de detectar los objetos cuyos centros se encuentra dentro de la misma. Las celdas realizan múltiples detecciones y generan un mínimo de 2 "bounding boxes" a las cuales se les asigna un porcentaje de confianza el cual representa que tan seguro está el modelo de que existe un objete en esa posición. Adicionalmente cada celda predice las probabilidades de clase de los objetos que pueden estar dentro de las celdas. Estas probabilidades de clase luego son multiplicadas por los porcentajes de confianza de cada "bounding box" en la celda para producir los porcentajes de confianza de clase de cada "bounding box". Finalmente, para refinar estas se aplica una técnica denominada "Non-Max Supression", la cual elimina aquellas "bounding boxes" con bajos porcentajes de confianza y también remueve "bounding boxes" que tengan un alto porcentaje de traslapado.

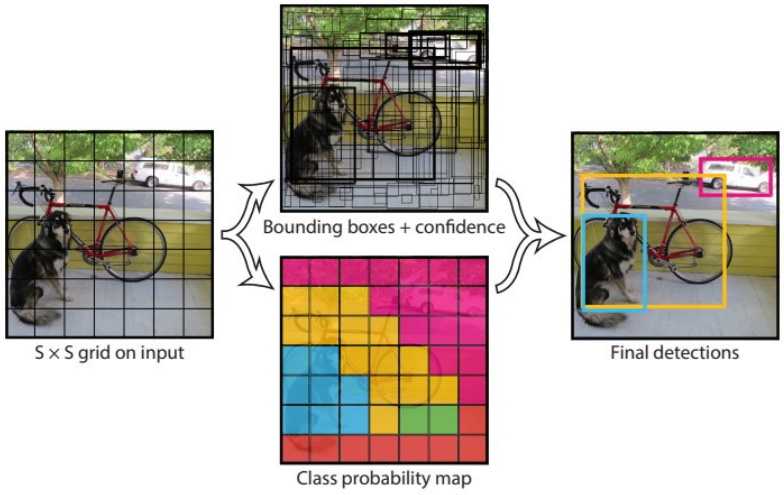

Ilustración 8: Generación de "bounding boxes" y el mapa de probabilidad de clase, que serán refinados para producir las predicciones finales. Figura tomada del estudio "You Only Look Once: Unified, Real-Time Object Detection" [24].

Visual Transformers

La arquitectura Visual Transformer apareció por primera vez en el año 2021 en el estudio "An Image is worth 16x16 words" [25] de Alexey Dosovitskiy y sus colaboradores. Esta arquitectura fue revolucionaria en cómo el campo del Aprendizaje Profundo entiende y procesa las imágenes. Este estudio inspirándose en cómo las Redes Transformers abarcan el problema del

procesamiento del lenguaje natural, adapta esta estructura para trabajar con imágenes en lugar de texto.

En lugar de utilizar convoluciones para extraer características de las imágenes, como se hace en las Redes Neuronales Convolucionales (CNN), los Visual Transformers dividen las imágenes en parches, que son luego proyectados y tratados como una secuencia de "palabras" al igual a cómo los Transformers manejan secuencias de texto. Cada parche se representa como un vector de características que se alimenta a través de una serie de capas de Transformer para capturar relaciones globales entre los parches.

Una de las innovaciones clave de los Visual Transformers son su capacidad para modelar dependencias a largo plazo en las imágenes, algo que las CNNs tradicionales pueden tener dificultades para lograr debido a su enfoque en las convoluciones. Esta capacidad permite a los Visual Transformers comprender y procesar patrones complejos y contextuales en las imágenes de manera más eficiente.

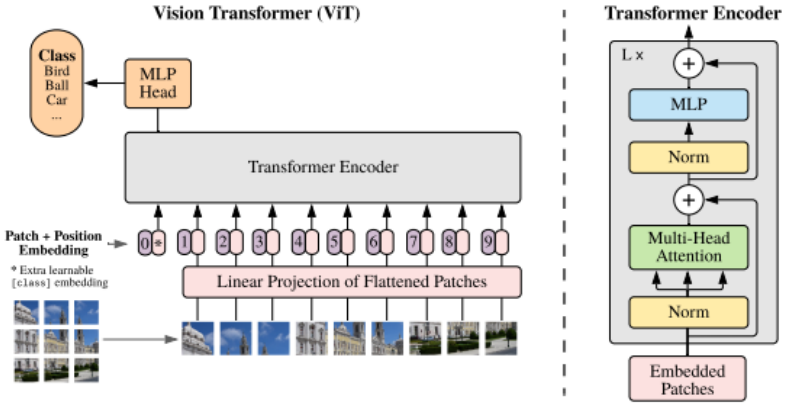

Ilustración 9: Descripción general del modelo ViT. Figura tomada del estudio "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" [25].

2.4.3. Métricas de Desempeño

Para evaluar y comparar el desempeño de los distintos algoritmos de detección de objetos, se utilizan una serie de métricas. Estas métricas permiten obtener una visión detallada de los distintos aspectos del rendimiento de cada algoritmo, permitiendo así identificar fortalezas y áreas de mejora. A continuación, se describen algunas de las métricas más comunes y relevantes en este campo:

Intersection Over Union (IOU): También conocido como el índice de Jaccard, se utiliza para medir el traslapado entre el "bounding box" generado por el detector, y el "bounding box" del objeto real en el ground truth (GT) o datos verdaderos. Este índice va desde 0 a 1. Con 0 siendo el caso donde ambas "bounding boxes" están completamente separadas, y 1 siendo el caso donde ambas están una sobre la otra.

$$IOU = \frac{|B \cap B^{gt}|}{|B \cup B^{gt}|}$$

Donde B es la bounding box dada por el detector y B^{gt} es la bounding box real del objeto en el GT.

True Positive (TP): Instancias donde el detector correctamente localiza y clasifica un objeto, y además la métrica IOU es superior a un umbral en específico, típicamente "0.5".

False Positive(FP): Instancias donde el detector incorrectamente detecta un objeto en una posición en la cual no existe ningún objeto de acuerdo con el ground truth. También califican como FP bounding boxes que tienen una métrica de IOU inferior al umbral especifico.

False Negative(FN): Instancias donde el detector no detecta objetos que están en el ground truth.

True Negative(TN): No es aplicable a la tarea de detección de objetos.

Precision: Cuantifica la capacidad del modelo de distinguir entre objetos reales y falsos positivos. Básicamente la habilidad del modelo de realizar detecciones que si sean verdaderas.

$$Precision = \frac{TP}{TP + FP}$$

Recall: Cuantifica la capacidad del modelo de detectar todos los objetos relevantes en la escena, en otras palabras, de correctamente detectar los objetos que se encuentran en el ground truth.

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

F1-Score: La media harmónica de la métrica de "Precision" y "Recall". Es una métrica que permite medir el desempeño del modelo de forma balanceada considerando Falsos Positivos y Falsos Negativos.

$$F1 - Score = \frac{2 * (Precision * Recall)}{(Precision + Recall)}$$

Mean Average Precision (mAP): Combina las métricas de "Precision" y "Recall" para distintos umbrales de IOU. Permite medir la habilidad del modelo de encontrar los objetos de forma precisa y además su habilidad de encontrar todos los objetos relevantes que están en el ground truth.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

Donde N representa el número de clases, y AP_i la métrica de Average Precison para la clase i.

2.5. Seguimiento de Objetos

El seguimiento de objetos es una tarea clave en el campo de la visión por computador y se integra en una variedad de sistemas modernos. Estos incluyen aplicaciones como el conteo de multitudes, el control de vehículos autónomos y el conteo de frutas en entornos agrícolas. La meta principal de esta tarea es identificar cada objeto de manera individual dentro de una secuencia de imágenes, asignarle una identidad única, y rastrear sus cambios en posición, forma y tamaño a lo largo de los sucesivos fotogramas.

En el campo de visión por computador suele utilizarse un algoritmo de detección junto a un algoritmo de seguimiento (Tracker) en un esquema conocido como "Tracking by Detection". Este esquema en primer lugar utiliza las detecciones que entrega el detector y asigna identidades únicas a cada objeto encontrado, luego en el siguiente fotograma el algoritmo de seguimiento intentara asociar las nuevas detecciones con los objetos ya identificados previamente. Las detecciones que no pueden ser asociadas se consideran nuevos objetos y se les asigna identidades únicas.

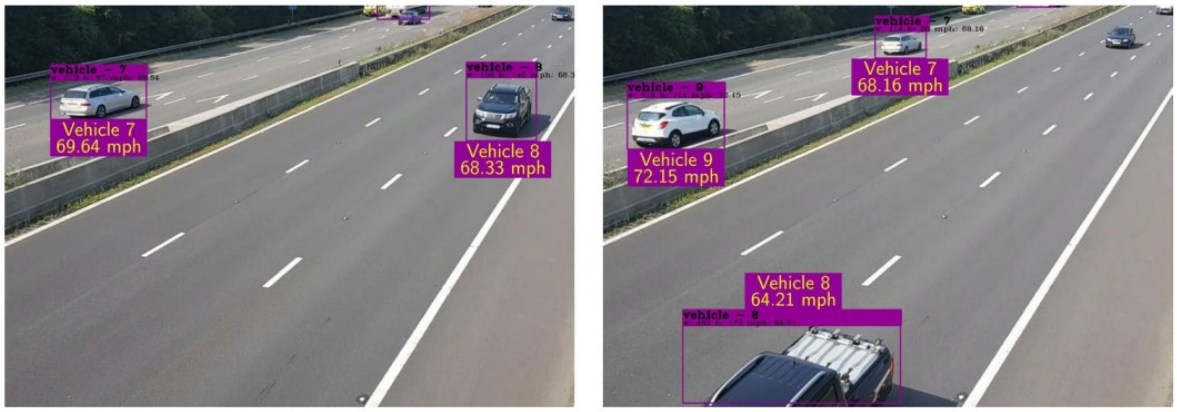

Ilustración 10:Ejemplo de seguimiento de objetos. Se asigna a cada vehículo un ID único y además se mide su velocidad. Imagen tomada del estudio "Camera-Based System for the Automatic Detection of Vehicle Axle Count and Speed Using Convolutional Neural Networks"[26].

2.5.1. Filtro de Kalman

El filtro de Kalman [27] es una algoritmo desarrollado por Rudolf E. Kalman en 1960 que se utiliza para estimar el estado de un sistema dinámico a partir de una serie de medidas incompletas o ruidosas. Se utiliza ampliamente en aplicaciones de ingeniería, como el control de vehículos y el procesamiento de señales, para predecir la posición, velocidad, y otras variables de estado de un objeto en movimiento.

Cuenta con dos pasos fundamentales: la predicción y la actualización.

En etapa de predicción, a partir de un estado inicial se predice el siguiente. Por estado entiéndase una serie de medidas que describen un objeto, como la velocidad, aceleración, posición entre otras. Tómese como ejemplo un vehículo que se mueve en una carretera. Utilizando el último estado conocido del vehículo, y considerando su posición y velocidad, el filtro de Kalman predice dónde y a qué velocidad se encontrará el vehículo en el siguiente momento. Además, en esta etapa se calcula la incertidumbre de la estimación, expresada como una matriz de covarianza, que indica cuán confiable y precisa es la estimación.

En la etapa de actualización, se determina el estado del sistema, utilizando una nueva observación real para calcular la ganancia de Kalman que determina cuánto peso se debe dar a la predicción anterior en comparación con la nueva observación al actualizar el estado actual. Si la nueva observación viene con una alta incertidumbre, el filtro de Kalman dará menos peso a esta observación en el cálculo del estado. Esto es común en situaciones donde los sensores son imprecisos. Estas 2 etapas son repetidas en un bucle.

2.5.2. Simple Online and Realtime Tracking (SORT)

Simple Online and Realtime Tracking conocido como SORT es un algoritmo de seguimiento por detección creado por A. Bewley et. al. [28] en el año 2016. Este algoritmo utiliza un filtro de Kalman para estimar las coordenadas futuras de las bounding boxes de cada uno de los objetos en el siguiente fotograma. Para asociar las detecciones en el fotograma actual con las detecciones en el último fotograma, se calcula la métrica de “Intersection Over Union” la cual permite calcular la superposición entre una detección y una predicción, donde 0 representa nula superposición y 1 total superposición entre ambas bounding boxes. Una vez calculada esta métrica entre cada predicción y detección, se crea una matriz de costo con los valores obtenidos. Luego se resuelve el problema de asignación usando el algoritmo húngaro, así asociando cada predicción a una sola detección con la cual se sobre lapa. Se asume que la detección en el fotograma actual y la detección del fotograma anterior, desde la cual se originó la predicción asociada, corresponden al mismo objeto.

Cuando los objetos entran en la escena por primera vez se les asigna una identidad, las detecciones en fotogramas subsiguientes que serán asociadas a este objeto compartirán esta identidad. Para evitar la creación de identidades nuevas a partir de detecciones erróneas SORT implementa un mecanismo de gestión de identidades que elimina aquellas que no reciben confirmaciones consistentes en los fotogramas subsiguientes. Específicamente, si una identidad

no se asocia con ninguna detección nueva en un número predefinido de fotogramas consecutivos, se presume que el objeto ha dejado la escena o que la detección inicial fue un error, y por tanto, esa identidad se descarta.

Debido a su simpleza y mínimo tiempo de ejecución se utiliza frecuentemente en aplicaciones que realizan seguimiento en tiempo real.

2.5.3. DeepSORT

Este algoritmo de seguimiento más conocido como DeepSORT fue creado en 2017 por Nicolai Wojke et. al. [29]. Representa una mejora para el algoritmo de SORT, con el cambio más destacable siendo la integración de un descriptor de apariencia el cual es obtenido aplicando una CNN pre-entrenada a las bounding boxes de las detecciones.

Para construir el problema de asociación se combinan dos métricas en una suma ponderada, la primera es la distancia de Mahalanobis entre las predicciones del filtro de Kalman y las detecciones en el nuevo fotograma. La segunda métrica corresponde a la similitud coseno entre el descriptor de apariencia de la detección actual y el descriptor de la detección en el fotograma previo. Se utiliza el algoritmo húngaro para resolver este problema de asignación.

Los descriptores de apariencia son cruciales cuando el componente de predicción de movimiento falla, especialmente después de que un objeto ha estado ocluido por varios fotogramas. En situaciones donde las trayectorias predichas pueden ser inexactas debido a la falta de visibilidad del objeto, los descriptores de apariencia permiten a DeepSORT "reconectar" con el objeto una vez que reaparece en la escena.

2.5.4. Strong Simple Online Real-Time Tracking

Este algoritmo de seguimiento conocido como StrongSORT fue creado en 2022 por YunHao Du et. al. [30] y representa una mejora substancial a DeepSORT. En primer lugar se reemplaza la CNN usada para extraer las características de apariencia por un método que extrae características mucho más discriminativas, “Bag of Tricks” (BoT) [31]. Se reemplaza el filtro de Kalman por el filtro NSA Kalman (Noise Scale Adjustment), el cual integra el porcentaje de confianza para calcular el estado de un objeto. Detecciones de alta confianza tendrán una mayor influencia en la estimación del estado, mejorando la precisión y estabilidad del seguimiento del objeto. Para compensar el movimiento de la cámara el cual suele impactar negativamente el proceso de seguimiento, StrongSORT emplea el algoritmo de “Enhanced Correlation Coefficient Maximization” (ECC), el cual estima movimiento de la cámara entre fotogramas, y aplica esta información para ajustar las predicciones del siguiente estado.

Para abordar los problemas de asociaciones y detecciones perdidas, StrongSORT permite la integración de dos algoritmos ligeros y plug-and-play que son independientes del modelo y no dependen de información de apariencia: AFLink y GSI. Cuando StrongSORT se combina con

estos algoritmos, se denomina StrongSORT++. Además, tanto AFLink como GSI pueden ser fácilmente incorporados en otros sistemas de seguimiento para mejorar su rendimiento sin requerir modificaciones significativas.

2.5.5. BoT-SORT

BoT-SORT es un algoritmo de seguimiento introducido en 2022 por Nir Aharon et al. [22]. Es una evolución de los algoritmos SORT y DeepSORT, pero también se basa en ByteTrack, integrando varias mejoras para un seguimiento más robusto y preciso.

Los algoritmos tradicionales basados en SORT utilizan un filtro de Kalman para predecir la posición de los objetos en los siguientes fotogramas. Sin embargo, BoT-SORT modifica este enfoque al mejorar el estado del filtro de Kalman, permitiendo no solo la predicción de la posición del objeto, sino también del ancho y alto del cuadro delimitador (bounding box) en fotogramas posteriores. Esto mejora la precisión en la asociación entre la predicción y la ubicación real del objeto.

Además, BoT-SORT introduce un mecanismo de compensación del movimiento de la cámara, lo que le permite ajustar la predicción de los objetos en escenas donde la cámara se mueve de manera brusca. Esto corrige el desplazamiento de los objetos debido al movimiento de la cámara, reduciendo los errores de seguimiento. BoT-SORT también adopta la estrategia de ByteTrack para mejorar la asociación de detecciones, procesando tanto detecciones de alta confianza como de baja confianza, lo que ayuda a mantener la identidad de los objetos a lo largo del tiempo.

2.5.6. Métricas de Desempeño

Para medir el desempeño de los algoritmos de seguimiento, se cuenta con múltiples métricas de evaluación, las cuales permiten medir aspectos fundamentales de la tarea de seguimiento como la eficacia de un algoritmo para seguir un objeto durante múltiples fotogramas, o que tan frecuentemente se asocia una nueva detección al objeto equivocado. A continuación, se definen las métricas más utilizadas según los resultados obtenidos en la Revisión Sistemática de la Literatura.

Falsos Positivos (FP): El algoritmo ha identificado un objeto en una posición de la imagen en la cual no existe un objeto de acuerdo con el ground truth (GT).

Falsos Negativos (FP): El algoritmo no logra identificar un objeto en una posición de la imagen en la cual sí existe un objeto de acuerdo con el ground truth (GT).

Intercambio de identidad o IDSW: Instancias donde el algoritmo de seguimiento cambia la identidad de un objeto de forma errónea. Por ejemplo, una persona identificada con el numero 3 incorrectamente le es asignada el identificador 2 en un fotograma.

Multi Object tracking Accuracy o MOTA: Mide la exactitud general del algoritmo de seguimiento, toma en cuenta que tan frecuentemente el algoritmo de seguimiento comete errores relacionados con Falsos Positivos, Falsos Negativos, e Intercambios de identidad. Una métrica cercana a 1 indica una alta exactitud.

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t}$$

Donde GT corresponde al Ground Truth y t corresponde a un instante de tiempo o fotograma.

Multi-Object Tracking Precision o MOTP: Mide la precisión de las predicciones de posición realizadas por el algoritmo de seguimiento. Se calcula la distancia entre las predicciones y las posiciones reales de los objetos por sobre el número de asociaciones correctas entre las predicciones y los objetos reales.

$$MOTP = \frac{\sum_{i,t} d_t^i}{\sum_t C_t}$$

Donde t representa el instante de tiempo específico, i corresponde al identificador unico de un objeto. d_t^i es la distancia entre la posición predicha y la posición real del objeto i en el instante de tiempo t.

Higher Order Tracking Accuracy o HOTA: Métrica creada recientemente en el año 2022, su objetivo es superar las limitaciones de métricas como MOTP y MOTA para medir el desempeño de un algoritmo de seguimiento de forma más precisa considerando la exactitud de detección y asociación del algoritmo de seguimiento.

$$HOTA = \sqrt{DetA * AssA}$$

Donde $DetA$ es la exactitud de la detección de objetos y $AssA$ es la exactitud de asociación la cual mide que tan bien el algoritmo de seguimiento mantiene las identidades sobre el tiempo.

IDentification F1 o IDF1: Mide la exactitud de las trayectorias predichas por el algoritmo de seguimiento considerando las métricas de "Precision" y "Recall" de los objetos que han sido identificados correctamente.

$$IDF1 = \frac{2|IDTP|}{2|IDTP| + |IDFP| + |IDFN|}$$

Donde $IDTP$ corresponde al número de detecciones correctamente asociadas a la trayectoria de un objeto definida por él GT. $IDFP$ es el número de detecciones que no pudieron ser asociadas a la trayectoria de un objeto definida por él GT, y finalmente $IDFN$ corresponde al número de veces que una trayectoria de un objeto en él GT no puede ser asociada a ninguna detección.

Capítulo 3: Estado del Arte

3.1. Preparación de la revisión

3.1.1 Preguntas de la investigación

P1: ¿Cuáles son las principales técnicas y/o algoritmos de Deep Learning utilizados para realizar la detección, seguimiento y conteo de objetos en una secuencia de imágenes (video)?

P2: ¿Qué herramientas, bibliotecas, repositorios y lenguajes de programación fueron ocupados para implementar los distintos algoritmos?

P3: ¿Cuáles son las métricas de evaluación utilizadas para evaluar los algoritmos de conteo de objetos?

P4: ¿Cuál es la distribución del dataset que se ocupa para el entrenamiento de los algoritmos, y que tratamiento se les aplica a los datos?

3.1.2 String de búsqueda

"object detection" AND "tracking" AND "count" AND "video" AND "occlusion" AND ("camera" OR "stereo camera" OR “drone”) AND ("dense" OR “crowd”)

Para la búsqueda de estudios se utilizan los siguientes motores de búsqueda, los cuales fueron escogidos debido a su reputación y su amplia cobertura de literatura académica. Estas plataformas ofrecen acceso a una diversidad de recursos, incluyendo artículos revisados por pares, tesis, y actas de conferencias, lo que garantiza una revisión exhaustiva y actualizada del estado del arte en área de conteo de objetos mediante secuencias de imágenes (videos).

- ScienceDirect
- IEEXplore
- Springer
- ACM Digital Library
- Wiley Online Library
- Scopus
- WOS

3.1.3. Criterios de inclusión y exclusión

Criterios de inclusión	Criterios de Exclusión
• Estudios que estén escritos en el idioma de búsqueda. • Estudios que han sido publicados dentro de los últimos 5 años. • Los estudios deben estar relacionados a la identificación, seguimiento y conteo de objetos en secuencias de imágenes. • Los estudios deben utilizar una CNN (Convolutional Neural Network) o red Transformer para detectar y/o segmentar los objetos.	• Estudios que están escritos en un idioma distinto al idioma de búsqueda. • Estudios con más de 5 años de antigüedad. • Estudios que no están relacionados al conteo automático de objetos. • Estudios que no utilizan una CNN o una red Transformer para detectar y/o segmentar los objetos.

Tabla 1: Criterios de Inclusión y Exclusión.

3.2. Resultados de la ejecución de la búsqueda

Se realizo la búsqueda en todos los motores previamente mencionados y se obtuvieron los siguientes resultados preliminares:

Fuente	***Resultados***
Scopus	84
Science Direct	267
IEEEXplore	355
Springer	229
Willey	59
ACM	14
WOS	0

Tabla 2: Resultados de la búsqueda.

Se examinaron los resultados minuciosamente para eliminar duplicados y estudios que no cumplieran con los criterios de inclusión y exclusión. Finalmente se obtuvieron los siguientes resultados por motor de búsqueda.

Fuente	***Resultados***
Scopus	4
Science Direct	6
IEEEXplore	7
Springer	6
Willey	2
ACM	0
WOS	0
Total	**25**

Tabla 3: Resultados finales.

3.3. Resultados de la revisión sistemática de la literatura

Luego de realizar la revisión sistemática de la literatura, es posible responder a las preguntas de investigación que se plantearon al comienzo de la investigación.

P1: ¿Cuáles son las principales técnicas y/o algoritmos de Deep Learning utilizados para realizar el la detección y conteo de objetos en una secuencia de imágenes (video)?

Según lo recopilado en la literatura académica, las arquitecturas que más predominan en la tarea de detección de objetos son YOLOv3, y YOLOv5, ambos con una representación de un 24% en los estudios analizados. La arquitectura de YOLOv4 les sigue con un 16%.

Cabe destacar que el 76% de los estudios utilizan un detector de la familia de detectores de YOLO.

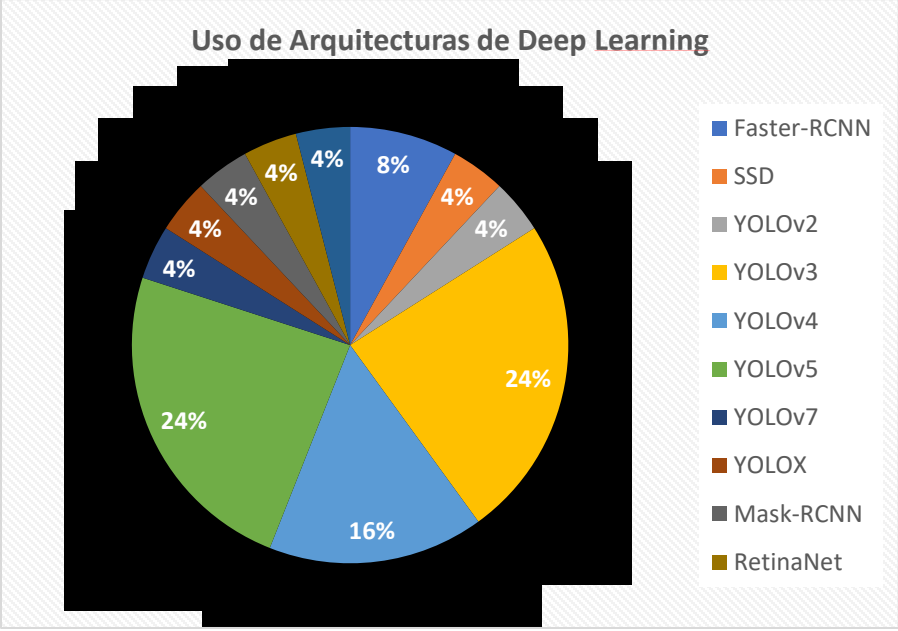

Ilustración 11: Grafico Circular de uso de modelos de detección.

En lo que respecta a los algoritmos de seguimiento de objetos, DeepSORT es el más utilizado según los estudios analizados, estando presente en el 32% de la literatura. Este algoritmo realiza la asociación de objetos basándose en su proximidad y similitud visual, utilizando una red neuronal convolucional (CNN) para generar vectores de características de los objetos detectados. Los algoritmos del filtro Kalman y StrongSORT le siguen en frecuencia de uso, representando el 16% y el 12% de los estudios, respectivamente.

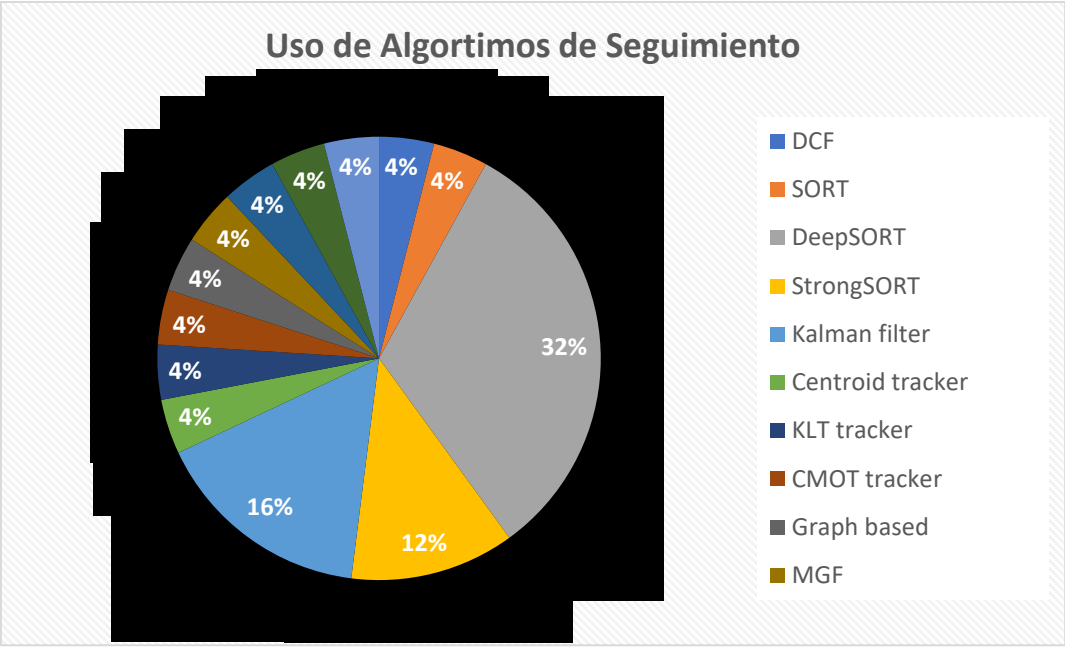

Ilustración 12: Grafico Circular de uso de algoritmos de seguimiento.

En cuanto a los métodos de conteo empleados, el 48% de los estudios analizados realizan el conteo de objetos basándose en las trayectorias generadas por los algoritmos de seguimiento. En segundo lugar, con una representación del 28%, se encuentra el método de la línea de conteo. Este método consiste en contar un objeto cada vez que su trayectoria cruza una línea imaginaria previamente definida en la pantalla. Se utiliza frecuentemente para mitigar el doble conteo de objetos, problema que puede surgir debido a la oclusión de objetos en la escena u otras complicaciones que dificultan la detección precisa y, por lo tanto, pueden generar múltiples trayectorias. Generalmente, la línea de conteo se sitúa en la región que presenta las mejores condiciones para la detección de objetos y la generación de trayectorias por parte de los algoritmos de seguimiento.

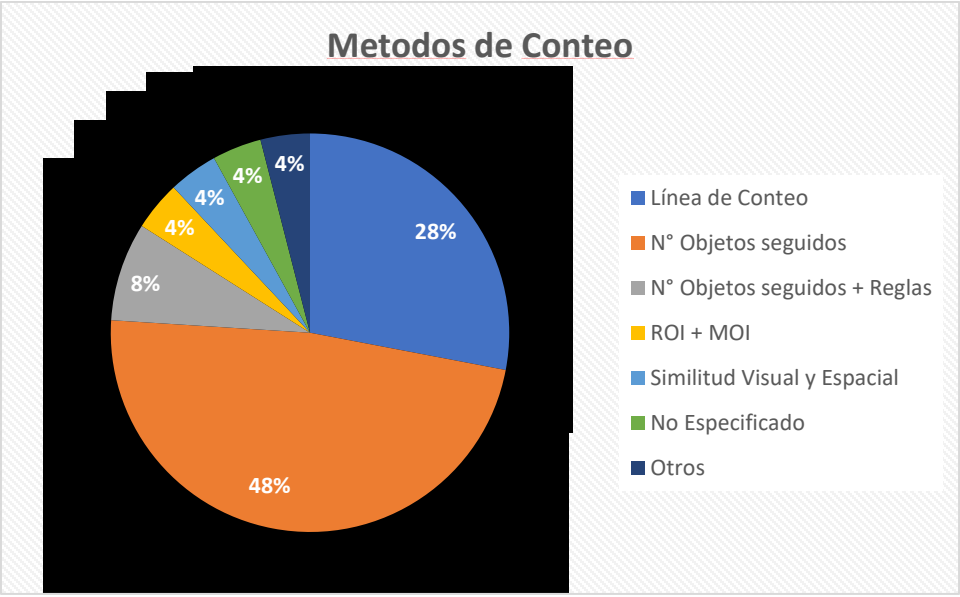

Ilustración 13: Grafico Circular de uso de métodos de conteo.

Es importante destacar que, dentro de la literatura, un estudio [32] aplica la técnica de "Structure from Motion" (SfM) para modelar la secuencia completa de imágenes en una sola escena en 3D. Este método permite proyectar las coordenadas 2D de los objetos a un espacio 3D y facilita la identificación de trayectorias repetidas, dado que estas exhibirán coordenadas similares en el modelo tridimensional.

P2: ¿Qué herramientas, bibliotecas, repositorios y lenguajes de programación fueron ocupados para implementar los distintos algoritmos?

El lenguaje de programación Python, es usado por todos los estudios que especifican el lenguaje que utilizaron para desarrollar los algoritmos. Respecto a las bibliotecas, "Pytorch" es usado para el desarrollo e implementación de los modelos de aprendizaje profundo en todos los estudios, excepto uno, el cual usa "Tensorflow". En 3 de los estudios, el framework de nvidia, "Tensorrt" se ocupa para optimizar los modelos de aprendizaje profundo disminuyendo los tiempos de inferencia y el costo computacional para ejecutarlos. Para el procesamiento de imágenes, OpenCV y Pillow son las librerías más populares.

P3: ¿Cuáles son las métricas de evaluación utilizadas para evaluar los algoritmos de conteo de objetos?

Según lo revisado en la literatura, para la evaluación de algoritmos de detección, la métrica más utilizada es Mean Average Precision (mAP) con una frecuencia de un 30% en todos los estudios. Le sigue la métrica de "Recall" con un 25%, y "Precision" con un 20%. Un patrón común observado es el uso de las métricas de mAP, Recall y Precision en conjunto.

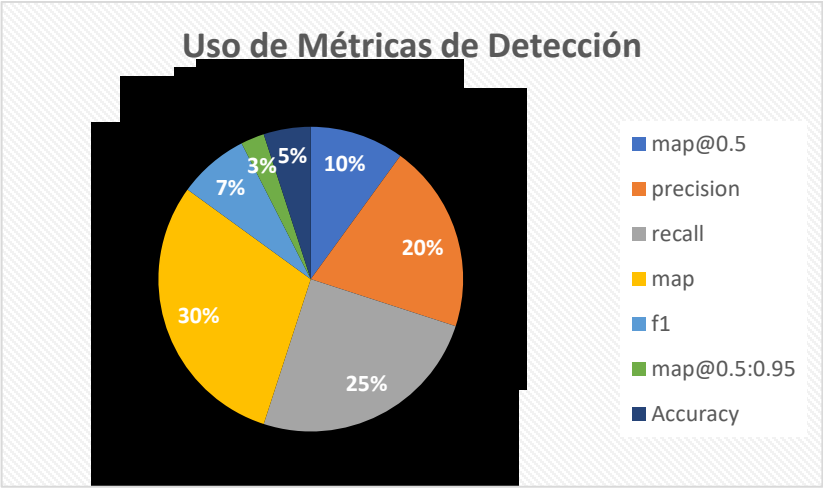

Ilustración 14: Grafico Circular de uso de métricas de detección.

Según lo revisado, la métrica que aparece con más frecuencia en los estudios es "Identity Switch" y Multi-Object Tracking Accuracy (MOTA) ambas con una frecuencia del 29% en todos los estudios. La primera mide el número de veces que el algoritmo de seguimiento incorrectamente cambia el ID de un objeto seguido, confundiéndolo por otro existente o uno completamente nuevo. La segunda métrica (MOTA) en cambio mide la exactitud del algoritmo de seguimiento, permite condensar en una métrica; falsos positivos, falsos negativo e Identity Switches.

En tercer lugar, los sigue MOTP con un 14% de uso.

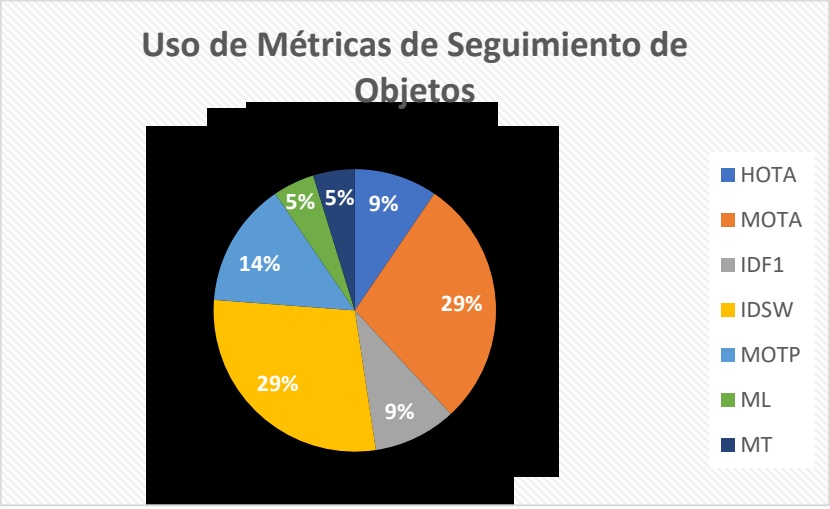

Ilustración 15: Grafico Circular de uso de métricas de seguimiento de objetos.

Por último, para la evaluación de los algoritmos de conteo propuestos, la métrica más utilizada, presente en el 27% de los estudios, es "Accuracy" (Exactitud), la cual expresa como porcentaje, el número de objeto en el ground truth dividido por el número de objetos contado por el

algoritmo propuesto. En segundo lugar, se encuentra la métrica “Mean Absolute Error” presente en un 20% de los estudios, por último, en tercer lugar, la métrica más común es “Coeficiente de determinación”, más conocida como R2 con un 17%.

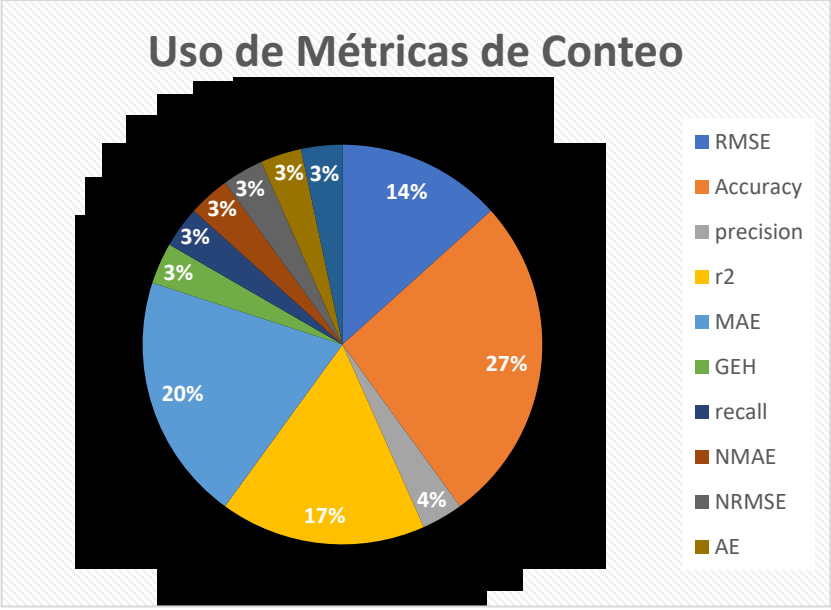

Ilustración 16: Grafico Circular de uso de métricas de conteo de objetos.

P4: ¿Cuál es la distribución del dataset que se ocupa para el entrenamiento de los algoritmos, y que tratamiento se les aplica a los datos?

Dado los 25 estudios analizados, existe una tendencia la tendencia de ocupar entre un 70% y 80% del Dataset para e entrenamiento de los modelos. Respecto al set de validación, su existencia varia, ya que al menos un 56% de los estudios no lo utilizan, y en cambio solo ocupan el set de prueba. En los estudios que lo utilizan generalmente está formado por entre un 10% y 20% de las muestras. Por último, para el set de prueba, en los casos donde no se utiliza el set de validación su composición es de entre un 20% y 30% de los datos, en el caso contrario, se compone de entre un 10% y 20% del total.

Solo un 28% de los estudios indican que utilizan alguna técnica del aumento de datos (data augmentation). Algunas de las técnicas ocupadas son “Flip Horizontal”, “Flip Vertical”, “Escalado de imagen”, modificación al azar del brillo, contraste y desenfoque gaussiano entre otros.

Capítulo 4: Planteamiento del proyecto de Tesis

4.1. Problema

La estimación de la cosecha es una tarea fundamental en el sector agrícola, ya que permite calcular anticipadamente diversas variables, como la cantidad de personal necesario, el volumen de almacenamiento requerido, los costos de transporte y la estimación del precio de venta. Tradicionalmente, este proceso se realiza de forma manual mediante la toma de muestras, lo cual implica un costo significativo en mano de obra y, además, introduce la posibilidad de errores humanos.

En los últimos años, el uso de tecnologías basadas en aprendizaje profundo ha surgido como una solución para abordar esta problemática. Los modelos de detección basados en Redes Neuronales Convolucionales (CNN) y redes transformer han ganado popularidad debido a su alta eficacia en este tipo de tareas. Estas tecnologías representan una oportunidad para reducir los costos asociados a la logística del conteo manual, optimizando así los procesos de estimación.

En el caso específico de la propuesta planteada, la recolección de datos para los experimentos se realiza mediante un dron que sobrevuela las plantaciones de arándanos, capturando secuencias de imágenes digitales de los frutos, así como otros datos relevantes. Este enfoque permite automatizar la toma de muestras para el desarrollo de modelos de detección eficientes.

El conteo automático de arándanos utilizando modelos de Aprendizaje profundo cuenta con las siguientes dificultades:

- Una de estas dificultades surge debido a la manera en que crecen, en racimos densos, lo que significa que los frutos se agrupan muy cerca unos de otros. Esta proximidad puede ocasionar que algunos arándanos queden ocultos o parcialmente cubiertos por otros frutos. En consecuencia, solo una porción de estos frutos es visible desde una perspectiva exterior, y no se puede apreciar el fruto en su totalidad. La ocultación parcial de los arándanos complica la tarea de determinar cuáles están maduros y listos para ser recolectados, lo que puede resultar en una recolección menos eficiente y potencialmente en una pérdida de producto. Además, para los sistemas de visión tradicionales, este agrupamiento denso puede interferir en la capacidad de los modelos para detectar correctamente cada fruto individual.
- El uso de drones equipados con cámaras para monitorear y analizar cultivos presenta varios desafíos técnicos significativos. Uno de los problemas más notables es la gestión de la trayectoria de vuelo del dron, que puede incluir movimientos dinámicos como rotaciones, ascensos, y descensos, así como el sobrevuelo de áreas ya inspeccionadas previamente. En el caso de los arándanos, si un dron sobrevuela una sección del cultivo que ya ha sido capturada por la cámara, el algoritmo de conteo puede no ser capaz de reconocer que esos arándanos ya fueron contabilizados. Esto resulta en un conteo duplicado, lo cual puede llevar a estimaciones incorrectas del volumen de la cosecha.

A pesar de estos retos, actualmente existen una amplia variedad de modelos de detección de objetos, incluyendo Redes Neuronales Convolucionales y Redes Transformers, complementados con técnicas avanzadas de procesamiento de imágenes, seguimiento de objetos y métodos de conteo. Estas herramientas tecnológicas ofrecen la posibilidad de desarrollar algoritmos de conteo mediante aprendizaje profundo que sean capaces de abordar y superar eficazmente estas dificultades específicas.

4.2 Hipótesis

La hipótesis del proyecto es definida como:

"Utilizando algoritmos avanzados de aprendizaje profundo, específicamente redes neuronales convolucionales y transformers, junto con técnicas de visión por computadora y algoritmos de seguimiento, se puede desarrollar un algoritmo capaz de realizar un conteo preciso de arándanos bajo distintos niveles de oclusión y movimiento dinámico de camara.

4.3. Objetivos

4.3.1. Objetivo General

Desarrollar un algoritmo de visión por computador capaz de detectar y contar arándanos de manera precisa en secuencias de imágenes densas, como videos, bajo el escenario de movimiento dinámico de la cámara.

4.3.2. Objetivos Específicos

OE1: Revisar la literatura académica para seleccionar técnicas y algoritmos adecuados para el conteo de objetos, enfocándose en identificar técnicas para la detección, el seguimiento y métodos de conteo de objetos.

OE2: Adaptar e implementar tres algoritmos con técnicas seleccionadas para lograr un conteo preciso de objetos en escenarios denso de objetos y en donde la cámara presenta un movimiento dinámico.

OE3: Evaluar los algoritmos de conteo implementados en situaciones donde el movimiento de la cámara sea dinámico, y los objetos en la escena, y el nivel de oclusión, varíen.

4.4. Alcance de la investigación

La tesis propuesta se enmarca en el proyecto FONDEF ID21I10256, en el cual se obtuvieron los materiales como el dron, y se capturaron las secuencias de video con arándanos que serán utilizadas para la etapa de desarrollo y experimentación de la tesis.

Cabe destacar que, según lo investigado durante la revisión sistemática de la literatura, el problema en específico a abordar en el desarrollo de esta tesis, el cual es el "Conteo Automático de Arándanos en Secuencias de Imágenes Densas", aún no está resuelto. Considerando esto, durante la ejecución de esta tesis se abordó el problema con el objetivo principal de contar los arándanos de forma automática en videos donde el desplazamiento de la cámara del dron es en una sola dirección y además la distancia entre la cámara y los arándanos es constante durante el video.

4.5) Metodología de Trabajo

La metodología de trabajo que se utilizó para el desarrollo de los objetivos de este trabajo está conformada por 5 etapas:

4.5.1 Revisión sistemática de la literatura

Con el fin de cumplir con el primer objetivo específico, se realizó una exhaustiva revisión del estado del arte, a través de una Revisión sistemática de la literatura, la cual se ejecutó en base a la metodología para Ingeniería de Software, propuesta por Barbara Kitchenham la cual consiste en 3 etapas:

- **Etapa 1:** Planificación de la revisión.
    - Identificación de la necesidad de la revisión.
    - Desarrollo de un protocolo de investigación.
- **Etapa 2:** Desarrollo de la revisión.
    - Identificación de la investigación.
    - Selección de los estudios primarios
    - Evaluación de la calidad del estudio.
    - Extracción y monitoreo de los datos.
    - Síntesis de datos.
- **Etapa 3:** Publicación de los resultados

4.5.2. Recolección de datos

Para el entrenamiento, validación y prueba de los modelos de aprendizaje profundo, y además para la prueba del algoritmo de conteo elaborado en este trabajo, es necesario contar con un conjunto de datos lo suficientemente amplio y robusto para que los modelos puedan aprender los patrones necesarios para la correcta detección de los arándanos, y además poder evaluar el algoritmo final en distintas condiciones de imagen.

Las muestras y secuencias de videos adicionales fueron capturadas por un dron con cámara el cual capturara las imágenes durante el vuelo. Las imágenes extraídas fueron etiquetadas de forma manual utilizando la herramienta gratuita Label Studio[33].

4.5.3. Implementación de los algoritmos

Con el fin de cumplir con el objetivo específico 2 (OE2), es necesario implementar los distintos algoritmos de detección, seguimiento y métodos de conteo que se descubran en la revisión del estado del arte para comparar sus desempeños y si es que se ajustan a las necesidades del problema propuesto.

Para la implementación de estos algoritmos y el desarrollo de los algoritmos de conteo final se contempló el uso del lenguaje de programación, “Python”, el cual cuenta con una gran cantidad de librerías para el desarrollo de modelos de aprendizaje profundo, como “Pytorch” y “Tensorflow” librerías que son ampliamente utilizadas en el área de la ciencia de datos. Además, cuenta con un gran soporte en una de las tareas fundamentales del proyecto como lo es la manipulación de imágenes, como lo son “OpenCV” y “Pillow”.

Por último, para realizar la selección de los algoritmos a utilizar, a partir de los resultados de la revisión sistemática, se emplearon las métricas correspondientes para medir el desempeño de los algoritmos de detección, seguimiento y los métodos de conteo para así poder realizar una comparación de estos, y seleccionar aquellos que mejor se adapten al problema propuesto.

4.5.4. Experimentos

La fase de experimentación comprende el proceso en el cual el algoritmo de conteo desarrollado será sometido a múltiples pruebas bajo diversas condiciones. El objetivo principal es verificar que el algoritmo sea eficaz en el conteo de arándanos en secuencias de imágenes densas y con movimiento dinámico de cámara. El desempeño de los algoritmos se evaluará utilizando las métricas correspondientes.

Para facilitar la evaluación, se desarrolló un software de simulación que permitió recrear el vuelo de un dron equipado con cámara, simulando la captura de imágenes mientras vuela de izquierda a derecha o de derecha a izquierda. Este software ofrece la posibilidad de ajustar varios parámetros, como:

- El número de arándanos a detectar.
- El nivel de oclusión.
- El nivel de iluminación, tamaño, ángulo, entre otros, de los arándanos.
- El modelo de detección a utilizar.
- Estrategias de conteo, como la definición de una línea de conteo.

Este enfoque permitió probar el algoritmo en condiciones controladas y personalizables, asegurando una evaluación exhaustiva de su desempeño. Además, facilitó la incorporación o modificación de parámetros clave para optimizar la precisión en el conteo de arándanos. Por ejemplo, fue posible cambiar fácilmente el modelo de detección por una arquitectura modificada para mejorar su rendimiento, si se estima conveniente.

Además, este enfoque permite realizar cientos de pruebas bajo diversas condiciones, lo que sería inviable en términos de tiempo y costo si se intentara replicar con secuencias de imágenes reales en las mismas circunstancias. Por último, adicionalmente el enfoque del simulador permitirá reproducir los resultados.

4.5.5. Reporte de Resultados

Una vez finalizada la fase de experimentación, se recopilaron los resultados del desempeño de los modelos para cada una de las pruebas realizadas. Analizando en detalle estos resultados se obtuvieron conclusiones respecto a que algoritmos se adaptan mejor a las necesidades de la tesis propuesta, las cuales son la detección, el seguimiento y el conteo preciso de los arándanos bajo distintas condiciones de densidad y oclusión. Igualmente, a partir de los hallazgos se detallan las dificultades y puntos de mejora que puedan existir.

Capítulo 5: Implementación y resultados

5.1 Preparación de Materiales

5.1.1 Conjunto de Datos

Para entrenar los modelos de detección de arándanos, que forman parte de los algoritmos de conteo, primero fue necesario recopilar un conjunto de datos. En este estudio, dicho conjunto se obtuvo en el contexto del proyecto FONDEF "ID21I10256", donde se capturaron múltiples videos de alta resolución utilizando un dron. Estos videos tienen una resolución de 3840 x 2160 píxeles y una frecuencia de 60 fotogramas por segundo (fps).

En general, los videos fueron grabados a una distancia de 1 metro de los arbustos de arándanos, y en la mayoría de los casos, el dron seguía un movimiento lineal de izquierda a derecha o de derecha a izquierda.

Aunque se capturaron múltiples videos, para el entrenamiento, validación y prueba de los modelos de detección se decidió utilizar solo tres de ellos. Esta selección se debió a que el resto presentaba una calidad de imagen inferior, lo que podía afectar negativamente la capacidad del modelo para generalizar. Además, de los videos seleccionados, no se utilizó todo el contenido grabado, ya que se recortaron aquellas secciones que no eran relevantes para el estudio o que no aportarían información útil al modelo.

En primer lugar, se descartaron las secciones donde se presentaba un efecto de difuminado, que dificultaba la identificación de los arándanos incluso para un observador humano. En segundo lugar, se eliminaron las secciones donde el dron repetía el mismo recorrido, ya que esto implicaría entrenar el modelo con fotogramas redundantes. Además, se descartaron situaciones en donde no se capturan arbustos de arándanos.

Siguiendo con el tratamiento del conjunto de datos, no se utilizaron todos los fotogramas disponibles debido a la alta tasa de captura por segundo. Este ritmo generaba problemas de redundancia, ya que la información entre fotogramas consecutivos no variaba significativamente. En su lugar, se decidió conservar un fotograma por cada cinco, creando una distancia razonable entre ellos.

Se entiende como razonable una distancia en la que el siguiente fotograma contiene nuevos arándanos visibles, mientras que los arándanos del fotograma anterior pueden ser observados desde ángulos diferentes o quedar parcialmente ocluidos debido a la nueva posición de la cámara. Este proceso fue automatizado, pero considerando que la velocidad del dron podía variar en distintas escenas, se verificó manualmente que se cumpliera este criterio. Los fotogramas redundantes identificados durante esta revisión manual fueron eliminados.

Para el etiquetado del conjunto de datos, se utilizó la herramienta gratuita “Label Studio” (Disponible en https://github.com/heartexlabs/label-studio).

Dado que el objetivo del estudio es determinar la cantidad de arándanos listos para cosechar, solo se etiquetaron aquellos en estado de maduración, identificados por su característico color azul oscuro. Los arándanos inmaduros o parcialmente maduros fueron deliberadamente ignorados.

Una vez completado el etiquetado, el conjunto de datos fue dividido en tres subconjuntos con una proporción aproximada de 70:15:15, correspondientes al entrenamiento, validación y prueba, de acuerdo con lo investigado durante la revisión sistemática de la literatura.

Es importante señalar que cada subconjunto proviene de un video diferente, y los fotogramas no fueron mezclados aleatoriamente. Esto se debe a que mezclar fotogramas de diferentes videos podría causar contaminación entre los subconjuntos, ya que las imágenes en una secuencia de video suelen ser muy similares. Si imágenes similares terminan en los subconjuntos de entrenamiento, validación y prueba, el experimento se vería comprometido. Por esta razón, cada subconjunto se conformó exclusivamente con secuencias de imágenes provenientes de un video distinto, evitando así la contaminación y asegurando la integridad del conjunto de datos.

Teniendo en cuenta lo anterior se obtienen los siguientes subconjuntos para el entrenamiento de los modelos de detección.

Subconjunto	**Imágenes etiquetadas**	**Arándanos etiquetados**
Entrenamiento	151	6791
Validación	30	1377
Prueba	26	1168
Total	207	9336

Tabla 4: Distribución del conjunto de datos.

5.1.2) Entorno de Desarrollo

En este estudio, es fundamental entrenar modelos de detección utilizando imágenes de alta resolución, además de realizar pruebas con los algoritmos de conteo desarrollados y evaluar los programas de utilidad los cuales permiten comprobar el desempeño del conteo automático.

Por estas razones, fue indispensable disponer de un equipo con especificaciones técnicas avanzadas. En el ámbito del aprendizaje profundo, resulta esencial contar con una tarjeta gráfica de alto rendimiento que permita entrenar los modelos de manera eficiente. Para la ejecución de las rutinas necesarias, se utilizó un equipo proporcionado por el Laboratorio de Imágenes de la Universidad del Bío-Bío. Este equipo cuenta con las siguientes características:

- Procesador: Intel I9 4Ghz
- Ram: 64 GB
- GPU: Nvidia RTX 4080

Aunque el equipo originalmente cuenta con el sistema operativo Windows 11, se instaló Ubuntu 22 a través del "Windows Subsystem for Linux" (WSL) para facilitar el entrenamiento y la ejecución de los modelos. Esto se debe a que, en general, las herramientas y frameworks de aprendizaje automático tienen un soporte más robusto en sistemas Linux, mientras que este soporte puede ser limitado o inexistente en Windows. WSL cuenta además con la ventaja de permitir la interacción de archivo alojados Windows con programas en Ubuntu. Adicionalmente, con el objetivo de tener un mayor control sobre el entorno, los paquetes instalados y garantizar la reproducibilidad futura de los experimentos, se decidió realizar todas las pruebas y desarrollos dentro de un contenedor Docker.

En términos generales, el contenedor cuenta con las siguientes características:

- Ubuntu 22
- CUDA 11.8
- Torch 2.2.2

Para el desarrollo de los algoritmos, se utilizó exclusivamente el editor de código fuente "Visual Studio Code", por su flexibilidad, amplio soporte para herramientas de desarrollo y múltiples funcionalidades que permiten agilizar la escritura de código.

5.2) Experimentos

Para la detección de objetos se empleó la arquitectura YOLOv8 [34], desarrollada por Ultralytics. En el momento de iniciar esta investigación, dicha versión era la más reciente de la familia YOLO. Además, el repositorio de Ultralytics ofrece gran facilidad para integrar, probar y modificar los modelos, lo que resulta especialmente útil al momento de implementar mejoras o personalizaciones.

Por otra parte, el seguimiento de los objetos se realizó utilizando el algoritmo BoT-SORT, el cual cuenta con métricas de seguimiento superiores en comparación con los algoritmos más utilizados, como SORT y Deep-SORT. Este rendimiento destacado de BoT-SORT lo convierte en una opción adecuada para rastrear de manera eficaz los objetos detectados en cada fotograma.

La arquitectura de YOLOv8, por defecto, comprende 3 cabezas de detección las cuales son las encargadas de localizar y asignar las clases correspondiente a cada objeto, en este caso arándanos. Cada una de estas cabezas de detección esta encargada de detectar objetos de cierto tamaño. Para efectos de esta tesis las cabezas de detección de denomina "CD" para simplificar la nomenclatura.

Tomando en cuenta que por defecto el modelo redimensiona todas las imágenes a 640x640 antes de procesarlas, lo siguiente es correcto.

- **Cabeza de detección "small":** Esta encargada de detectar objetos pequeños. Recibe un mapa de características de 80 x 80 desde P3. Se denomina CD3.
- **Cabeza de detección "medium":** Esta encargada de detectar objetos de mediano tamaño. Recibe un mapa de características de 40 x 40 desde P4. Se denomina CD4.
- **Cabeza de detección "large":** Esta encargada de detectar objetos pequeños. Recibe un mapa de características de 20 x 20 desde P5. Se denomina CD5.

Como puede observarse los mapas de características de mayor tamaño están encargados de detectar objetos pequeños esto, ya que una mayor superficie significa que los objetos pequeños aún son distinguibles luego de las operaciones de downsampling. En cambio, los mapas de menor superficie solo conservaran los objetos de mayor tamaño eliminando los pequeños.

Cuando las imágenes originales capturadas por el dron, con una resolución de 3840 x 2160, son procesadas por YOLOv8, estas son redimensionadas a 640 x 640 píxeles. Además, para preservar la relación de aspecto (proporción ancho/alto), se aplica un proceso de letterboxing y padding, que da como resultado una imagen efectiva de 640 x 360 píxeles, con bandas negras de 140 píxeles añadidas en la parte superior e inferior.

Analizando el set de entrenamiento encontramos que en general un arándano tiene las dimensiones de 35x35 pixeles, lo que significa que luego del escalamiento, el arándano solo medirá 5.83 pixeles dentro de la imagen de 640x640. Una vez procesada la imagen y siendo el turno de las cabezas detectoras, las dimensiones finales del arándano en cada uno de los 3 mapas de características finales serán las siguientes:

- p3 ≈ 0.73 píxeles
- p4 ≈ 0.36 píxeles

- p5 ≈ 0.18 píxeles

Como consecuencia, el arándano no llega siquiera a abarcar una celda completa en estos mapas de características, lo que dificulta su correcta detección. Además, durante el proceso de downsampling, buena parte de la información del fondo se fusiona con la del objeto, complicando aún más la distinción del arándano en las capas finales.

Para solucionar este problema, inicialmente se aumentó la resolución de la imagen a 1280×1280, con el fin de generar mapas de características de mayor tamaño y, por ende, aumentar la representación del arándano en los mapas de características finales.

Buscando aumentar la representación del arándano, y tomando en cuenta la existencia de arándanos de menor tamaño al promedio, se decide optimizar la búsqueda de objetos pequeños agregando una cuarta cabeza de detección, de acuerdo con lo planteado por Zhenhui Zheng et al. [35]. Este enfoque incorpora una cabeza de detección específica para objetos muy pequeños (CD2), la cual opera sobre el mapa de características P2 de 160×160 cuando la resolución de entrada es 640×640, o 320×320 en el caso de 1280×1280. De este modo, P2 proporciona una mayor representación final del objeto, en este caso un arándano (alrededor de 2.92 celdas), y al ser procesado por la cabeza de detección CD2 se incrementa la probabilidad de detectar frutos de menor tamaño[36].

Teniendo estos cambios en cuenta, las dimensiones finales del arándano en cada uno de los mapas de características son las siguientes:

Mapa de Características	Dimensiones	Tamaño final del arándano (celdas)
P2	320 x 320	2.92
P3	160 x 160	1.46
P4	80 x 80	0.73
P5	40 x 40	0.36

Tabla 5: Dimensiones del mapa de características para cada objeto de varios tamaños.

Dado que se decidió experimentar con un modelo basado en la inclusión de CD2 al modelo de arquitectura YOLOv8 con cabezas CD3-CD4-C5, resultando en CD2-CD3-CD4-CD5, también se consideró evaluar diversas configuraciones, como el uso exclusivo de CD2 o CD2-CD3, con el objetivo de observar cómo afecta la eliminación o incorporación de cabezas de detección al desempeño del modelo en la detección de objetos de tamaño pequeño.

La principal justificación para remover cabezas como CD4 y CD5 radica en permitir que el modelo se enfoque exclusivamente en objetos pequeños. Estas cabezas están especializadas en la detección de objetos más grandes, lo que podría desviar el enfoque del modelo de los arándanos, que son significativamente pequeños en las imágenes procesadas. Reducir el número de cabezas de detección podría optimizar la precisión del modelo para este tipo de objetos, al mismo tiempo que simplifica la arquitectura y el procesamiento.

Dicho esto, los experimentos modificando la arquitectura de YOLOv8 serían los siguientes:

N° Arquitectura	Configuración
1	CD2
2	CD2-CD3
3	CD2-CD3-CD4
4	CD2-CD3-CD4-CD5
5	CD3-CD4-CD5 (Base)

Tabla 6: Arquitecturas experimentales para YOLOv8.

Como se puede observar en la ilustración 17, los principales cambios radican en el incremento de la capa de entrada en el Backbone de 640x640 a 1280x1280. La adición de un nuevo bloque de Upsampling y Downsampling en el cuello (Neck) de la red, para enriquecer a la red con las características extraídas por el mapa de características P2. Finalmente se agrega una cabeza extra de detección CD2, la cual está especializada en la detección de objetos muy pequeños y depende en gran parte como entrada del mapa de características P2.

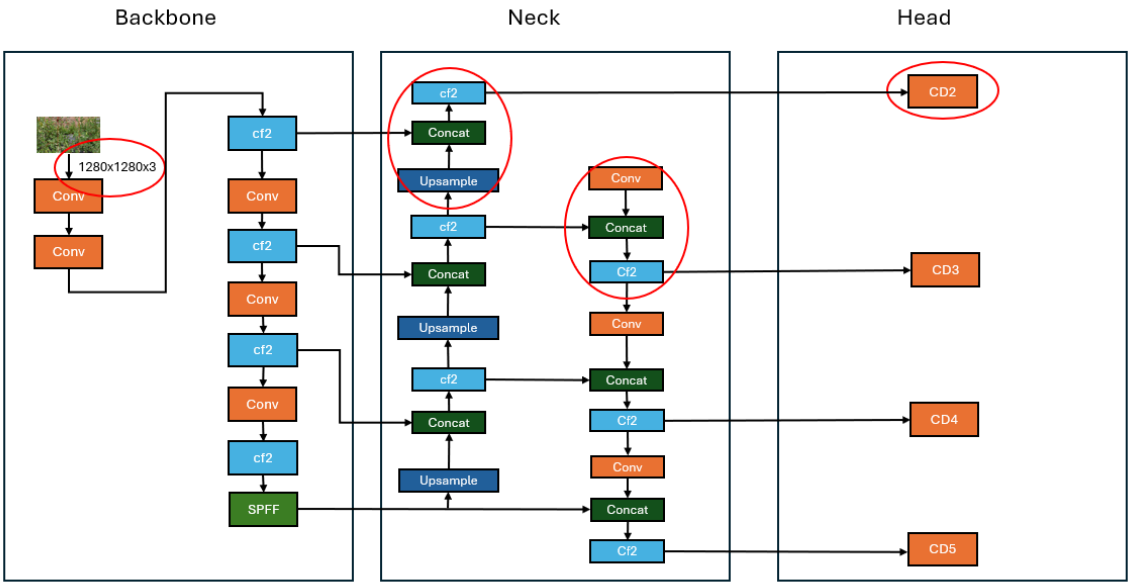

Ilustración 17: Diagrama resumido de modificaciones realizadas a YOLOv8.

5.3) Desarrollo del Simulador

Como se mencionó anteriormente, con el propósito de realizar múltiples experimentos bajo diversas condiciones, donde variables como el número de arándanos y el nivel de oclusión varían, se estimó necesario desarrollar un software de simulación. Esto se debe a que, considerando la cantidad de experimentos a realizar, 270 como se detalla en la sección siguiente, el tiempo, los recursos necesarios y las especificidades de las variables involucradas hacen inviable obtener los datos manualmente dentro del alcance de este proyecto.

El objetivo principal del simulador es simular el vuelo de un dron mientras graba una secuencia de imágenes en las que se observan arándanos y hojas que ocluyen parcialmente los frutos.

Este software de simulación además permite el cumplimento de los objetivos específicos OE2 y OE3, ya que permite implementar los algoritmos de detección con arquitectura modificada y además permite su evaluación en un ambiente donde el dron simulado realiza un movimiento dinámico y existe una gran densidad de objetos.

5.3.1) Descripción General

El software de simulación es una herramienta desarrollada en Python que permite simular el vuelo de un dron deslizándose de izquierda a derecha o de derecha a izquierda sobre una plantación de arándanos. Este simulador fue diseñado con el objetivo de generar distintos escenarios de variada complejidad, donde puedan evaluarse las diferentes modificaciones en la arquitectura de los modelos de detección y analizar cómo estas afectan los resultados del conteo.

Para crear estos escenarios, se implementaron múltiples parámetros configurables, entre los cuales se incluyen:

- Cantidad de "pantallas" en el experimento, que de forma básica simulan la cantidad de arbustos.
- Número de arándanos por pantalla.
- Porcentaje de oclusión en cada pantalla.
- Semilla de aleatoriedad del experimento, para garantizar la reproducibilidad

Además, el usuario puede seleccionar el modelo de detección a utilizar y ajustar parámetros clave, como:

- El porcentaje mínimo de confianza requerido para validar una detección.
- El porcentaje de "intersección sobre unión" (IoU)” necesario para considerar una detección válida.

Este enfoque proporciona un entorno flexible y controlado para realizar pruebas exhaustivas, facilitando el análisis del impacto de las modificaciones en los modelos de detección.

5.3.2) Funciones

Como se especificó en la sección anterior, la función principal del software es la simulación de la captura de secuencias de imágenes que toma un dron de una plantación de arándanos bajo diversas condiciones, y el conteo de arándanos utilizando múltiples arquitecturas de modelos de detección en conjunto los algoritmos de conteo. Con el objetivo de realizar una variedad de pruebas bajo diferentes condiciones, el software de simulación ofrece múltiples funcionalidades que lo hacen posible. Estas se pueden agrupar en tres categorías o submenús:

- Generación de Arándanos Sintéticos
- Generación de Oclusiones Sintéticas
- Opciones de Detección y Conteo

Generación de Arándanos sintéticos

Este submenú ofrece múltiples opciones para generar los arándanos a contar bajo diversas condiciones de iluminación, tamaño, posición, etc.

Se divide en 2 submenús

El primero nombrado como “Generar Samples” permite indicar las características básicas de los arándanos simulados durante el experimento.

- N° Samples: Define la cantidad de Arándanos por pantalla a generar. Como se ve en la imagen X, se generarían 100 muestras por cada pantalla. La generación es totalmente aleatoria por lo que las muestras pueden aparecer en cualquier posición de la pantalla, incluso permitiendo a cada arándano ocluir a otro. (Explicar el concepto de pantalla)
- Tamaño Base: El tamaño en pixeles de los arándanos a simular. El tamaño es dado por [Tamaño Base x Tamaño Base], donde la altura es igual al ancho ya que los arándanos tienden a tener una forma esférica.
- Velocidad: Es el número de pixeles que la cámara se moverá en una dirección en cada fotograma, simulando la velocidad con la que un dron recorre una plantación.

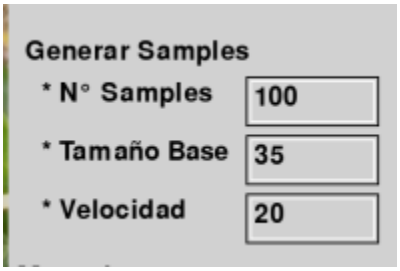

Ilustración 18: Submenú de generación de instancias de arándanos sintéticos.

El Segundo submenú [im2], denominado “mutaciones” contiene múltiples opciones para modificar la apariencia de los arándanos que serán generados.

Es necesario mencionar que los arándanos simulados, no provienen de la nada, sino que en cambio el simulador utiliza como base 20 imágenes de arándanos reales sin fondo. Es con el fin de diversificar la apariencia de los arándanos base, y de replicar las distintas condiciones de iluminación, posición, rotación y enfoque y tamaño de cada arándano en particular en una plantación real, que se aplican las siguientes transformaciones o “mutaciones” como se denominan dentro del simulador.

Cabe destacar que cada una de las mutaciones tienen una probabilidad de 0 a 100 porciento de ocurrir dado por el parámetro “Prob”.

- Rotación: El porcentaje de rotación en sentido del reloj o contrarreloj que puede llegar a aplicarse. Lo que permite simular parcialmente las distintas posiciones en las que puede aparecer un arándano.
- Tamaño: El porcentaje que puede llegar a incrementarse el tamaño de un arándano. Simulando la variedad en tamaño que naturalmente presentan los arándanos.
- Flip Horizontal: Invertir horizontalmente los pixeles de un arándano.
- Gaussian Blur: Aplicar un filtro de desenfoque gaussiano sobre el arándano a simular para simular el desenfoque natural de los arándanos que no están siendo enfocados por el lente. Se permite ajustar la intensidad con el parámetro “kernel”.
- Contraste y Brillo: Con el fin de simular las condiciones de diversa iluminación que puede presentar un arándano ya siendo ocluido o estando en sectores de la planta donde la iluminación es menor, se agrega este parámetro conjunto.

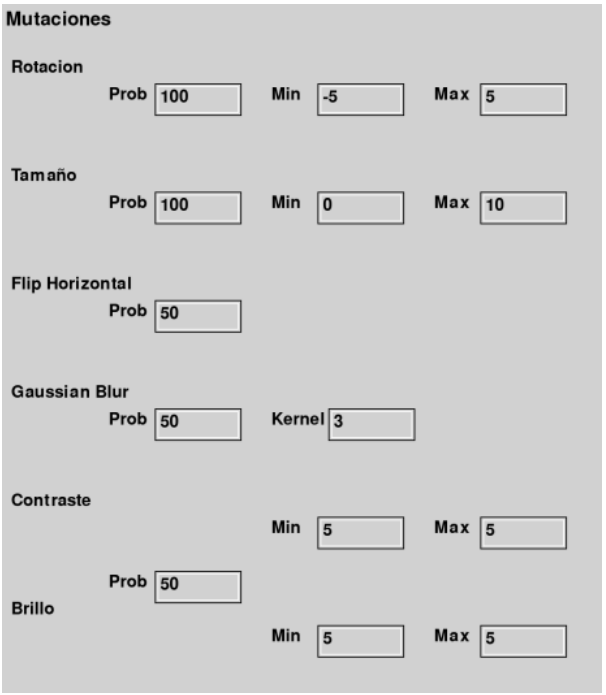

Ilustración 19: Submenú de mutaciones.

Ilustración 20: Muestras de arándanos reales usados para la generación de arándanos sintéticos.

Generación de Oclusiones sintéticas

Este submenú contiene opciones para la generación de oclusiones en la forma de parches de hojas reales, los cuales pueden generados sobre los arándanos simulando la forma en la que ocurren las oclusiones reales.

Permite escoger entre 2 parches de hojas, un parche de 64 x 64 pixeles y el otro d 80 x 80 px. Adicionalmente permite definir el porcentaje de la pantalla que será ocluido, con un 0% siendo el mínimo donde no existe oclusión, y 100% en el caso donde la pantalla se encuentra totalmente cubierta de parches de hojas. En la imagen [im3] se puede visualizar distintas configuraciones de oclusión.

Imagen ejemplo 1, 2, 3

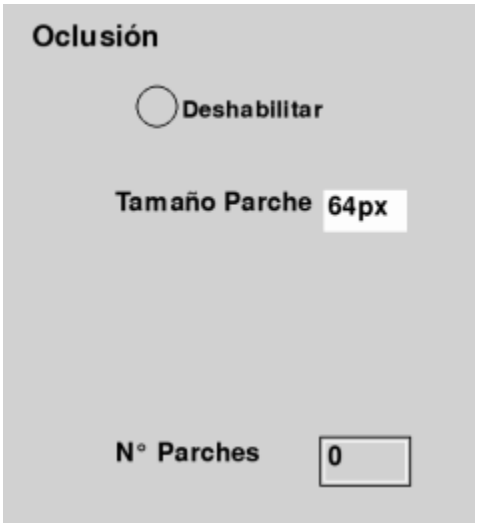

Ilustración 21: Submenú de oclusión.

Opciones de Detección y Conteo

Por último, este submenú contiene funciones relevantes para la tarea de detección y el conteo de arándanos. Así como una opción para asegurar la reproductibilidad de los experimentos y simulaciones a realizar.

El submenú “Detección” contiene 3 parámetros para rápida y sencillamente cambiar el modelo de detección a utilizar o modificar los parámetros principales para la tarea de detección. Estos son:

- Modelo: Indica la ruta al modelo de detección basado en arquitectura YOLOv8 a utilizar para la tarea de conteo de arándanos.
- Min Conf: Porcentaje de confianza mínima que debe tener un arándano detectado para ser considerado como un objeto detectado. En el caso de la ilustración 22, solo los objetos con un porcentaje de confianza mayor a 0.25 serán aceptados como una detección de arándano, en cambio los objetos bajo el umbral serán descartados.
- IOU: El umbral de Intersección sobre Unión (IoU) máximo que pueden tener los objetos entre sí para la Supresión No Máxima (NMS). Este parámetro permite reducir las detecciones duplicadas.

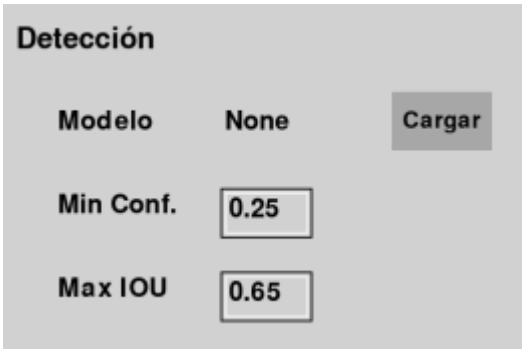

Ilustración 22: Submenú de configuración del modelo de detección.

El submenú de Conteo permite modificar parámetros relevantes para el conteo de los arándanos.

Los parámetros Línea 1 y Línea 2 permiten establecer líneas de conteo en áreas de la pantalla, las cuales al ser cruzas por uno de los arándanos al cual se le realiza un seguimiento, se procederá a incrementar el conteo global de arándanos en 1. Definir “Línea 1” es obligatorio para el funcionamiento del programa, mientras que “Línea 2” tiene el propósito de atrapar los arándanos que no fueron contados por “Línea 1” por cualquier motivo.

El parámetro “ROI” y sus sub-parámetros permiten definir la zona de la pantalla sobre la cual se quiere realizar el conteo. Se definen las coordenadas de un punto 1 (x, y) y punto 2 (x, y) los cuales definen un rectángulo o región de interés (ROI).

Por último, el sub-parametro “Seed” permite definir la semilla de aleatoriedad con la será posible reproducir los resultados que fueron obtenidos durante el desarrollo de esta tesis en el caso que a futuro se quiera evaluar o modificar los algoritmos propuestos en este estudio.

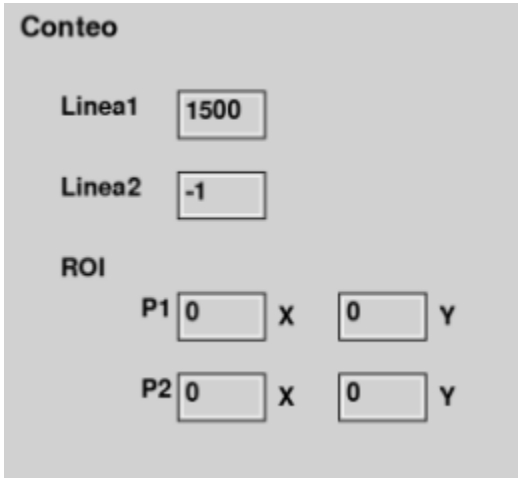

Ilustración 23: Submenú para opciones de conteo.

Ejecución de la simulación

En el simulador, una simulación está compuesta por múltiples pantallas. El término "pantalla" dentro del simulador hace referencia a un área de 3840 x 2160 píxeles, que representa la vista capturada por la cámara del dron de acuerdo con su resolución. Cada pantalla contiene α arándanos simulados y un β% de oclusión, donde la oclusión es generada por parches de hojas que simulan la obstrucción natural de los frutos debido a la vegetación circundante.

Como se muestra en la Imagen 4, en este ejemplo específico, cada pantalla contiene:

- 100 arándanos simulados.
- 20% de la pantalla ocluida por parches de hojas, replicando la oclusión natural que ocurre en los arbustos de arándanos.

Los experimentos descritos en la Sección 5 están conformados por 10 pantallas, las cuales son recorridas en secuencia de derecha a izquierda, simulando el movimiento del dron en vuelo sobre la plantación.

Para mejorar la precisión en el conteo de arándanos, el simulador implementa un sistema de doble línea de conteo, siguiendo el enfoque descrito en [37]. Este método se emplea para capturar arándanos que no fueron detectados en la primera línea, lo que puede ocurrir si el modelo de detección identifica un arándano después de haber pasado la primera referencia de conteo.

En escenarios reales, debido al ángulo de captura de la cámara, un arándano puede estar ocluido en un fotograma, lo que provoca que sea ignorado por la primera línea de conteo. Sin embargo, al avanzar el dron y cambiar la perspectiva, es posible que ese mismo arándano se vuelva visible

en un fotograma posterior, permitiendo su detección en la segunda línea de conteo. Por lo tanto, es un método que beneficia al algoritmo de conteo tanto para casos simulados como casos reales.

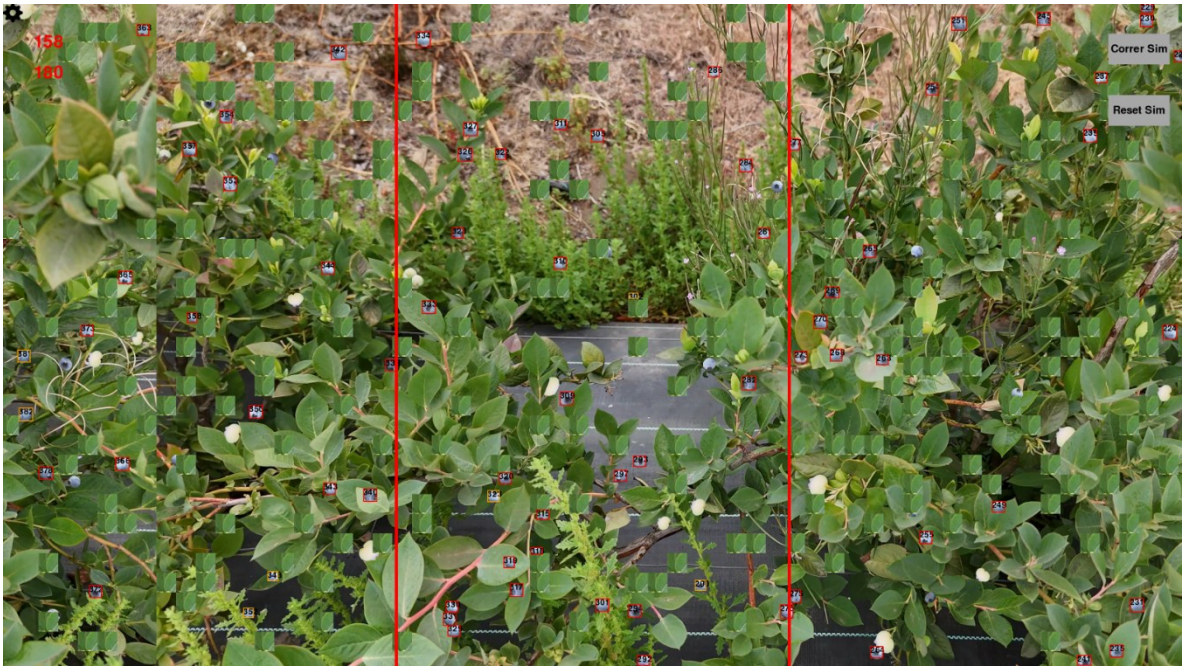

Ilustración 24: Simulador en funcionamiento.

5.3.3) Lógica de Conteo

El algoritmo de conteo está diseñado para garantizar un registro preciso y confiable del número de arándanos detectados en cada experimento. Para ello, ejecuta una serie de operaciones organizadas en tres fases principales: Fase de Creación: Identificación y asignación de un identificador único a cada nuevo arándano detectado, estableciendo un mecanismo de validación para evitar falsos positivos. Fase de Seguimiento: Actualización de las posiciones de los arándanos rastreados en cada fotograma, manejando casos de desaparición temporal por oclusión y evitando el sobre conteo. Fase de Eliminación: Depuración de detecciones incorrectas y eliminación de arándanos que han salido del campo de visión del dron o que no cumplen los criterios de seguimiento.

Cada una de estas fases trabaja en conjunto para mejorar la precisión del conteo, minimizando errores y garantizando la fiabilidad de los resultados obtenidos en la simulación. A continuación, se detalla el funcionamiento de cada fase.

- **Fase de Creación:** Cuando un arándano aparece por primera vez en la pantalla y no ha sido previamente detectado y rastreado por el algoritmo, se le asigna un identificador único. Este identificador ayudara al algoritmo de seguimiento rastrear su posición en los fotogramas siguientes.
  Para evitar el seguimiento y conteo de falsos positivos, cada arándano recién detectado se le asigna inicialmente la etiqueta de "tentativo". Solo después de haber sido rastreado

de manera consistente en cinco fotogramas consecutivos, su estado cambia a "confirmado", permitiendo que sea contabilizado.
Este método ayuda a prevenir situaciones donde una detección errónea (falso positivo) cruza la línea de conteo y afecta el conteo global. Si un arándano en estado "tentativo" no logra ser seguido en cinco fotogramas seguidos, se elimina del seguimiento y no se suma al conteo global, evitando un incremento erróneo en los resultados.

- **Fase de Seguimiento:** Durante esta fase, el algoritmo actualiza las posiciones de los arándanos que están siendo rastreados en el fotograma actual. Además, se implementa un mecanismo para manejar la desaparición y reaparición de arándanos en la escena. Si un arándano previamente rastreado no aparece en el fotograma actual, su edad se incrementa en 1. Si un arándano que había desaparecido reaparece, su edad se reinicia a 0.
  En esta fase, si un arándano rastreado cruza una línea de conteo, se le asigna la etiqueta "contado" y se incrementa el contador global en 1. Este mecanismo impide que un mismo arándano sea contado nuevamente si cruza la segunda línea de conteo, evitando así un sobre conteo.
  Finalmente, un arándano en estado "tentativo" puede convertirse en "confirmado" si logra ser rastreado en cinco fotogramas consecutivos.

- **Fase de Eliminación:** Durante esta fase son eliminados los arándanos cuya edad supera los 60, lo que significa que durante los últimos 60 fotogramas no se pudo volver a localizar un arándano previamente rastreado, asumiendo que han salido del campo de visión de la cámara.
  En esta fase además se eliminan los falsos positivos de arándanos que pudieron ser rastreados debido a alguna falencia del detector. Esto se logra cuando un “arándano” que está siendo rastreado, el cual tiene una etiqueta “tentativo” no logra ser rastreado en 5 fotogramas consecutivos. De ser este el caso, se elimina este “arándano” del diccionario de arándanos rastreados. Si dicho arándano había cruzado la línea de conteo previamente, su eliminación reduce el contador global en 1 para corregir el conteo.

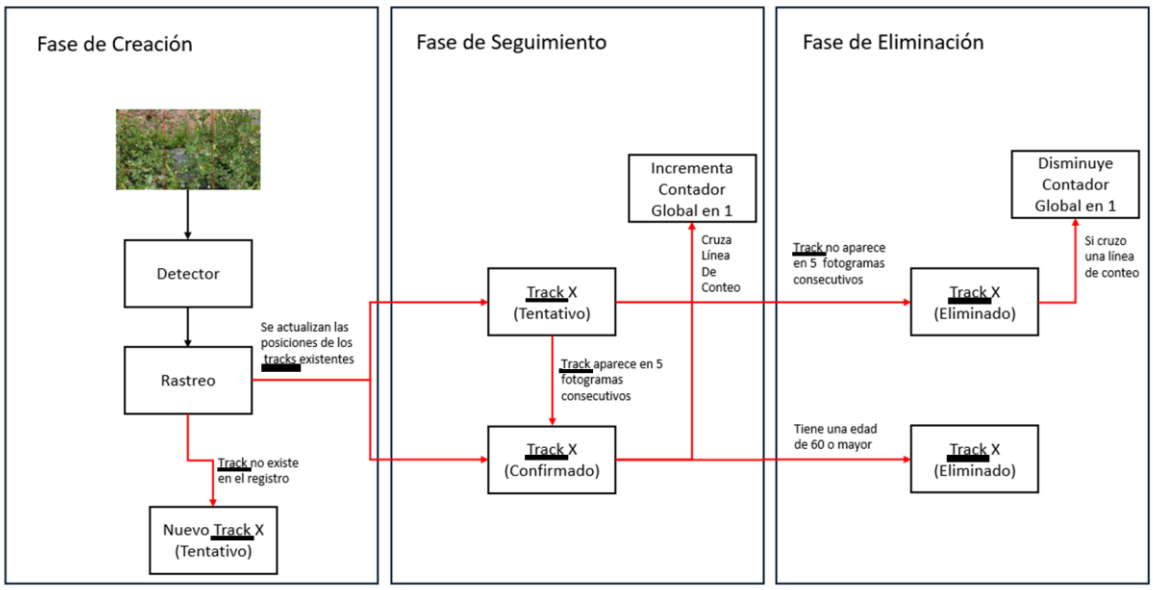

Ilustración 25: Visualización de la estrategia de conteo, y cada una de sus etapas.

5.4) Resultados de Experimentos

5.4.1) Entrenamiento de modelos de detección

Como se especifica en la sección de diseño de experimentos fue necesario evaluar 5 arquitecturas de modelo de detección de YOLOv8. Cada una agregando o removiendo cabezas de detección con el objetivo de comprobar que arquitecturas son las más efectivas para detectar objetos pequeños los cuales en este caso son los frutos de arándanos.

Las arquitecturas que se evaluaron son:

- CD2
- CD2-CD3
- CD2-CD3-CD4
- CD2-CD3-CD4-CD5
- CD3-CD4-CD5

Con el fin de evitar el caso donde un entrenamiento de un modelo obtiene resultados negativos simplemente por la aleatoriedad del entrenamiento, se decidió entrenar 3 modelos por arquitectura, lo que implico entrenar 15 modelos. De los 3 modelos generados por arquitectura solamente conservamos el cual tenga una mayor métrica de mAP@0.5, y en caso de que 2 modelos tengan la misma métrica se utilizó la métrica de f1-score para resolver esta disputa.

Los hiper-parametros de entrenamiento para cada modelo fueron:

Hiper Parámetro	**Valor**
Epochs	300
Batch Size	4
Img Size	1280
Pesos Pre-entrenados	Yolov8x entrenado con el Dataset de COCO
Patience	100
Learning rate	0.01

Tabla 7: Hiper Parámetros de entrenamiento.

Como puede observarse, los modelos fueron entrenados utilizando el máximo batch size soportado por la tarjeta gráfica al entrenar el modelo más complejo. En este caso, el modelo "CD2-CD3-CD4-CD5" permitió un batch size de 4 imágenes.

Para garantizar condiciones de entrenamiento equitativas entre todas las arquitecturas evaluadas, se utilizaron los mismos hiperparámetros durante el proceso de entrenamiento. Esto implica que todas las arquitecturas fueron entrenadas con un batch size de 4, a pesar de que, en teoría, los modelos más ligeros podrían haber soportado un batch size mayor.

Esta decisión asegura una comparación justa del desempeño de cada modelo, eliminando las posibles ventajas que podrían significar configuraciones de entrenamiento diferenciadas.

El entrenamiento de cada modelo tomo aproximadamente 7 horas.

Una vez entrenados los modelos, los resultados por arquitectura fueron:

Arquitectura	N° Modelo	Imágenes	Instancias	Precision	Recall	F1-score	mAP@0.5
CD2	1	26	1168	0,899	0,87	**0,884**	0,919
	2	26	1168	0,894	0,865	0,879	0,918
	3	26	1168	0,899	0,853	0,875	0,917
CD2-CD3	1	26	1168	0,872	**0,883**	0,877	0,92
	2	26	1168	0,892	0,867	0,879	0,919
	3	26	1168	0,898	0,851	0,874	0,92
CD2-CD3-CD4	1	26	1168	0,873	0,867	0,87	0,912
	2	26	1168	0,887	0,864	0,875	**0,921**
	3	26	1168	0,884	0,865	0,874	0,917
CD2-CD3-CD4-CD5	1	26	1168	0,887	0,856	0,871	0,915
	2	26	1168	**0,905**	0,85	0,877	0,917
	3	26	1168	0,884	0,851	0,867	0,916
CD3-CD4-CD5	1	26	1168	0,878	0,878	0,878	0,918
	2	26	1168	0,878	0,878	0,878	0,918
	3	26	1168	0,878	0,878	0,878	0,918

Tabla 8: métricas de detección de objetos para todos los modelos entrenados.

Conservando solamente el mejor modelo por cada arquitectura los resultados fueron los siguientes:

Arquitectura	Imágenes	Instancias	Precision	Recall	F1-score	mAP@0.5
CD2	26	1168	0,899	0,87	**0,884**	0,919
CD2-CD3	26	1168	0,872	**0,883**	0,877	0,92
CD2-CD3-CD4	26	1168	0,887	0,864	0,875	**0,921**
CD2-CD3-CD4-CD5	26	1168	**0,905**	0,85	0,877	0,917
CD3-CD4-CD5	26	1168	0,878	0,878	0,878	0,918

Tabla 9: Métricas de detección de objetos para los modelos con las mejores métricas para cada arquitectura.

Como se puede observar, todos los modelos presentan métricas de mAP@0.5 superiores al 90%, lo que indica un desempeño general sólido en la detección de arándanos. Además, los modelos muestran valores muy similares entre sí, con una variación menor al 1% tanto en el F1-score como en el mAP@0.5, lo que sugiere una consistencia en el rendimiento de las diferentes arquitecturas.

Las diferencias más notables entre modelos se observan en las métricas de precisión (precision) y sensibilidad (recall). En general, los modelos de detección de arándanos muestran una mayor capacidad para minimizar falsos positivos, lo que se refleja en valores elevados de precisión.

Entre todas las arquitecturas evaluadas, el modelo con la configuración CD3-CD4-CD5 se destaca al obtener las métricas más balanceadas de precisión y recall, con ambos valores igualados en 0.878. Esto indica que este modelo mantiene un equilibrio óptimo entre la detección de verdaderos positivos y la minimización de falsos negativos.

5.4.2) Algoritmos de conteo

Como se mencionó previamente en la sección de diseño de experimentos, el algoritmo de conteo será evaluado bajo distintas condiciones de densidad de arándanos y niveles de oclusión. Además, se investigará cómo las diferentes arquitecturas del modelo de detección influyen en el desempeño del algoritmo de conteo.

Considerando estas variables, el número inicial de experimentos a realizar es de 90. Sin embargo, para garantizar que los resultados obtenidos no sean consecuencia de una configuración aleatoria favorable en la distribución de los arándanos y los parches de oclusión, cada conjunto de experimentos se repetirá tres veces, utilizando una semilla única en cada iteración para asegurar la reproducibilidad. Las semillas utilizadas para la generación de valores pseudoaleatorios fueron obtenidas de random.org, un servicio que emplea ruido atmosférico para generar valores verdaderamente aleatorios (True Random). Se optó por este método con el objetivo de minimizar cualquier posible sesgo en la distribución de los arándanos y los parches de oclusión.

Por lo tanto, el número total de experimentos a realizar será de 270, permitiendo un análisis más robusto y estadísticamente confiable del rendimiento del algoritmo.

Variables	**Rango de valores**
Arquitectura	• CD2 • CD2-CD3 • CD2-CD3-CD4 • CD2-CD3-CD4-CD5 • CD3-CD4-CD5
N° de Instancias	• 50 • 100 • 200
Porcentaje de Oclusión	• 0 • 2 • 5 • 10 • 20 • 50
Semillas de aleatoriedad	• 24266 • 53621 • 92521

Tabla 10: Las variables para los experimentos y los valores que pueden tomar.

5.4.3) Resultados de los experimentos

Una vez ejecutados los 270 experimentos, se obtienen los resultados presentados en la Tabla 11. Cada arquitectura evaluada fue sometida a 54 experimentos, los cuales se descomponen en 18 experimentos por cada una de las 3 semillas aleatorias utilizadas.

Los resultados reportados en la tabla están expresados en términos de las métricas MAPE (Error Porcentual Absoluto Medio), MAE (Error Absoluto Medio) y RMSE (Raíz del Error Cuadrático Medio), las cuales cuantifican los errores cometidos en los experimentos. Cada métrica resume el desempeño del algoritmo al condensar los errores obtenidos en las tres semillas dentro de 18 valores representativos por arquitectura.

CD2

N° de Instancias x pantalla	% oclusión	MAPE	MAE	RMSE
50	0	0,666667	3,333333	3,829708
	2	1,004689	5	6,137318
	5	2,4278	12	12,56981
	10	4,885086	24	24,17988
	20	9,580406	45,33333	45,64355
	50	22,27447	82	82,47424
100	0	0,3	3	3,316625
	2	0,735313	7,333333	8,831761
	5	2,154882	21,33333	22,01515
	10	4,206164	41	41,34812
	20	8,913503	83,66667	84,71718
	50	20,59741	156,3333	157,4092
200	0	0,616667	12,33333	13,22876
	2	1,270647	25,33333	26,17887
	5	2,660245	52,66667	53,23533
	10	5,212588	102	103,0825
	20	9,664736	182,3333	182,8998
	50	21,53103	320	320,1156

CD2-CD3

N° Instancias x pantalla	% oclusión	MAPE	MAE	RMSE
50	0	0,2	1	1,732051
	2	1,205762	6	7,071068
	5	3,102158	15,33333	16,10383
	10	5,902588	29	29,32007
	20	10,143	48	48,33908
	50	23,35804	86	86,58329
100	0	0,566667	5,666667	5,744563
	2	1,43813	14,33333	15,02221
	5	3,198653	31,66667	31,79623
	10	5,197167	50,66667	50,7346
	20	9,764814	91,66667	92,42474
	50	21,12688	160,3333	161,688
200	0	1,433333	28,66667	29,33712
	2	2,173372	43,33333	43,9621
	5	3,687348	73	73,38256
	10	6,132188	120	121,1143
	20	10,60412	200	200,3613
	50	23,27982	346	346,2003

CD2-CD3-CD4

N° Instancias x pantalla	% oclusión	MAPE	MAE	RMSE
50	0	0,133333	0,666667	0,816497
	2	1,005093	5	5,259911
	5	2,764365	13,66667	14,08309
	10	5,291728	26	26,20433

CD2-CD3-CD4-CD5

N° Instancias x pantalla	% oclusión	MAPE	MAE	RMSE
50	0	0,266667	1,333333	1,632993
	2	1,072028	5,333333	5,715476
	5	2,628594	13	13,0767
	10	4,749033	23,33333	23,42363

	20	9,158314	43,33333	43,76452		20	8,734139	41,33333	41,44876
	50	21,09264	77,66667	78,08329		50	19,37544	71,33333	71,50758
100	0	0,333333	3,333333	3,366502	100	0	0,433333	4,333333	4,358899
	2	1,070088	10,66667	11,19524		2	1,136486	11,33333	11,83216
	5	2,525253	25	25,40997		5	2,592593	25,66667	26,05763
	10	4,37707	42,66667	42,95734		10	4,035398	39,33333	39,68207
	20	8,273363	77,66667	78,18568		20	7,91958	74,33333	75,15983
	50	19,02075	144,3333	145,1287		50	18,58487	141	142,4886
200	0	0,933333	18,66667	18,85029	200	0	1,05	21	21,61018
	2	1,621566	32,33333	32,56276		2	1,554799	31	31,6807
	5	2,794807	55,33333	55,45569		5	2,660271	52,66667	53,07228
	10	5,093303	99,66667	100,507		10	4,718216	92,33333	92,7811
	20	8,924383	168,3333	168,5734		20	8,465259	159,6667	159,7884
	50	20,23095	300,6667	300,9164		50	18,70567	278	278,2026

CD3-CD4-CD5

N° Instancias x pantalla	**% oclusión**	**MAPE**	**MAE**	**RMSE**
50	0	0,133333	0,666667	1,154701
	2	0,803752	4	4,966555
	5	2,157619	10,66667	11,16542
	10	4,002533	19,66667	20,10804
	20	7,327215	34,66667	35,14731
	50	18,09197	66,66667	68,09797
100	0	0,266667	2,666667	2,828427
	2	0,836119	8,333333	9,469248
	5	2,020202	20	20,70427
	10	3,557358	34,66667	35,65576
	20	7,031369	66	67,35478
	50	16,86949	128	129,6328
200	0	0,75	15	15,45962
	2	1,203679	24	24,2212
	5	2,27285	45	45,22905
	10	4,020562	78,66667	79,53616
	20	7,352108	138,6667	139,2288
	50	18,033	268	268,2623

Tabla 11: Resultados para los experimentos por arquitectura.

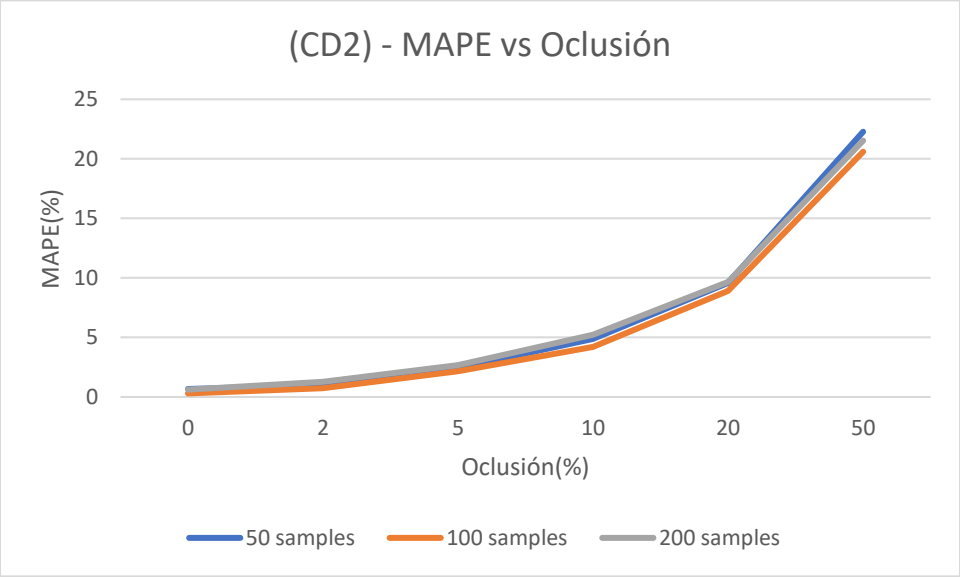

Ilustración 26: Grafico de MAPE vs Oclusión para la arquitectura "CD2".

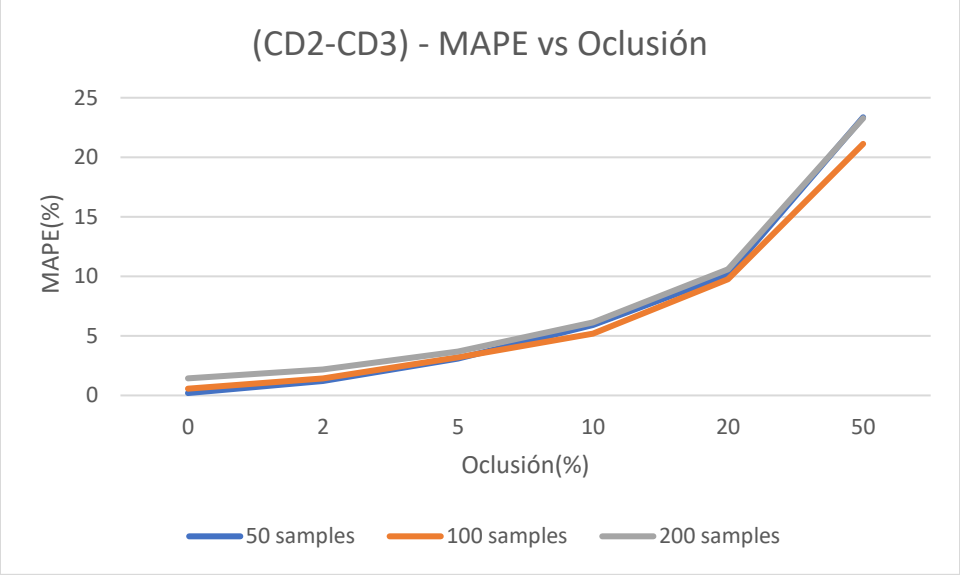

Ilustración 27: Grafico de MAPE vs Oclusión para la arquitectura "CD2-CD3".

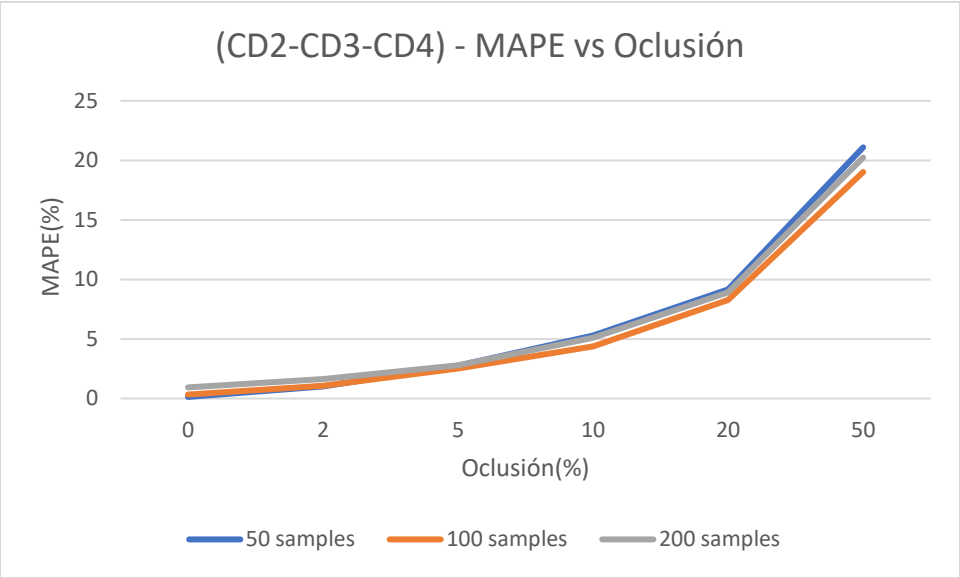

Ilustración 28:Grafico de MAPE vs Oclusión para la arquitectura "CD2-CD3-CD4".

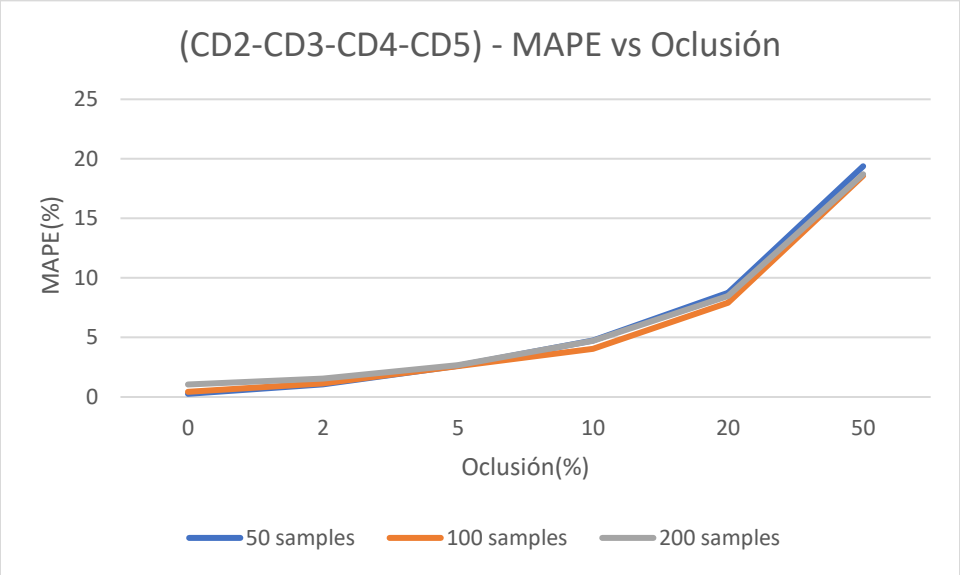

Ilustración 29:Grafico de MAPE vs Oclusión para la arquitectura "CD2-CD3-CD4-CD5".

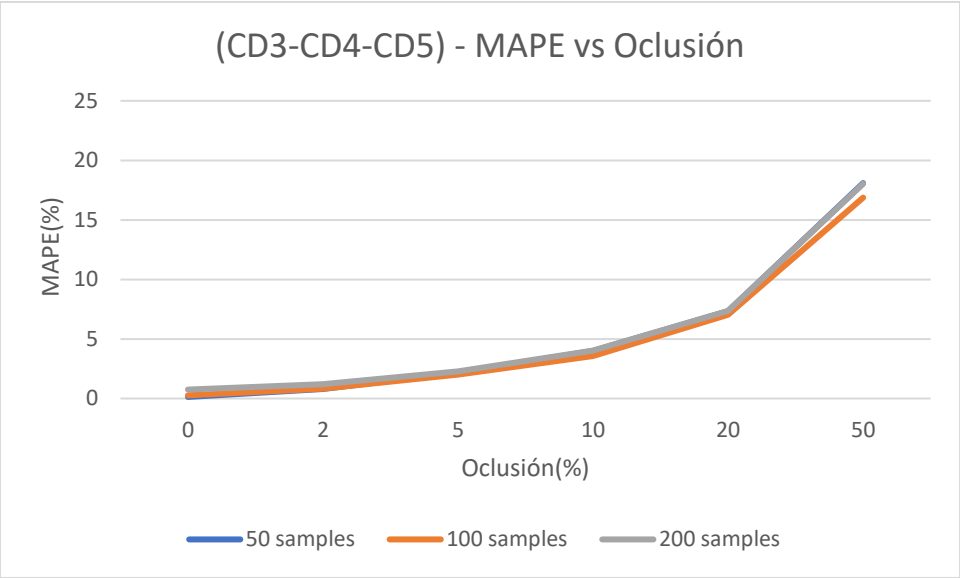

Ilustración 30:Grafico de MAPE vs Oclusión para la arquitectura "CD3-CD4-CD5".

Dado los resultados obtenidos los cuales pueden apreciarse en la tabla 11 y las ilustraciones 25 a 29, es posible notar que el modelo que utiliza las cabezas de detección CD3-CD4-CD5 siempre obtiene el menor MAPE en comparación al resto de los modelos para cada situación dada de nivel de Oclusión.

En cambio, el modelo que utiliza las cabezas de detección CD2-CD3 obtiene los peores resultados respecto al MAPE, siendo consistentemente superado por los otros modelos.

Por último, en la mayoría de los casos los resultados con un nivel de oclusión igual o inferior al 10% obtienen una métrica de MAPE inferior al 5%.

Los resultados presentes en la tabla 11, son analizados en detalle en la sección 6.2.

Capítulo 6: Discusión de Resultados

6.1) Rendimiento de los modelos de detección con distintas arquitecturas

Como puede observarse en la tabla 9, los resultados de los 5 modelos sobre el conjunto de prueba son positivos, ya que todos obtienen métricas sobre el 91% para mAP@0.5 lo que indica un buen desempeño en la detección de los objetos de interés (arándanos). Además, las métricas de precisión y recall muestran valores altos y equilibrados sobre un 85%, lo que sugiere que los modelos no solo identifican correctamente los objetos, sino que también minimizan los falsos negativos y falsos positivos. Esto indica que los modelos son capaces de generalizar adecuadamente sobre el conjunto de prueba, manteniendo una alta fiabilidad en sus predicciones.

Ahondando en los resultados podemos observar que para la métrica de mAP@0.5 obtenidas entre los 5 modelos son muy similares entre si con una variación menor al 1%. Esto sugiere que los modelos presentan un rendimiento consistente y estable en la tarea de detección de objetos. La baja variabilidad en los valores de mAP@0.5 indica que, a pesar de diferencias en arquitectura, todos los modelos han logrado un nivel de desempeño similar. Esto podría deberse a la calidad del dataset de entrenamiento, permitiendo a todos los modelos generalizar de forma robusta.

Los resultados obtenidos para las métricas de precisión y recall son similares entre los modelos evaluados, con variaciones de aproximadamente 2% en precisión y hasta 3% en recall. El f1-score se encuentra entre 87.5% y 88.4%.

Un modelo particularmente destacable es "CD3-CD4-CD5", ya que presenta el mismo valor de precisión y recall (87.8%), lo que indica un equilibrio óptimo entre ambas métricas. Esto significa que este modelo logra una buena capacidad de detección de objetos (minimizando los falsos negativos) sin comprometer la reducción de detecciones incorrectas (minimizando los falsos positivos).

Insertar la misma imagen evaluada por los 5 modelos escogidos y hablar de posibles diferencias.

Si bien en la mayoría de los casos los resultados de detección son precisos, como se muestra en la Figura [5], las métricas no son perfectas por lo que existen errores. En algunas imágenes o arbustos de arándanos pueden presentarse errores de detección, tales como, la doble detección de un mismo arándano (falso positivo) o la fusión de dos arándanos en una sola detección (falso negativo).

Estos errores varían según el modelo utilizado, ya que un modelo puede fallar en la detección de un arándano específico, mientras que otro modelo puede identificarlo correctamente, como se ilustra en la Figura [6], donde “CD2”, “CD2-CD3-CD4-CD5”, y “CD3-CD4-CD5” presentan un resultado similar al ground truth del set de prueba, mientras que “CD2-CD3” y “CD2-CD3-CD4” exhiben errores de detección doble.

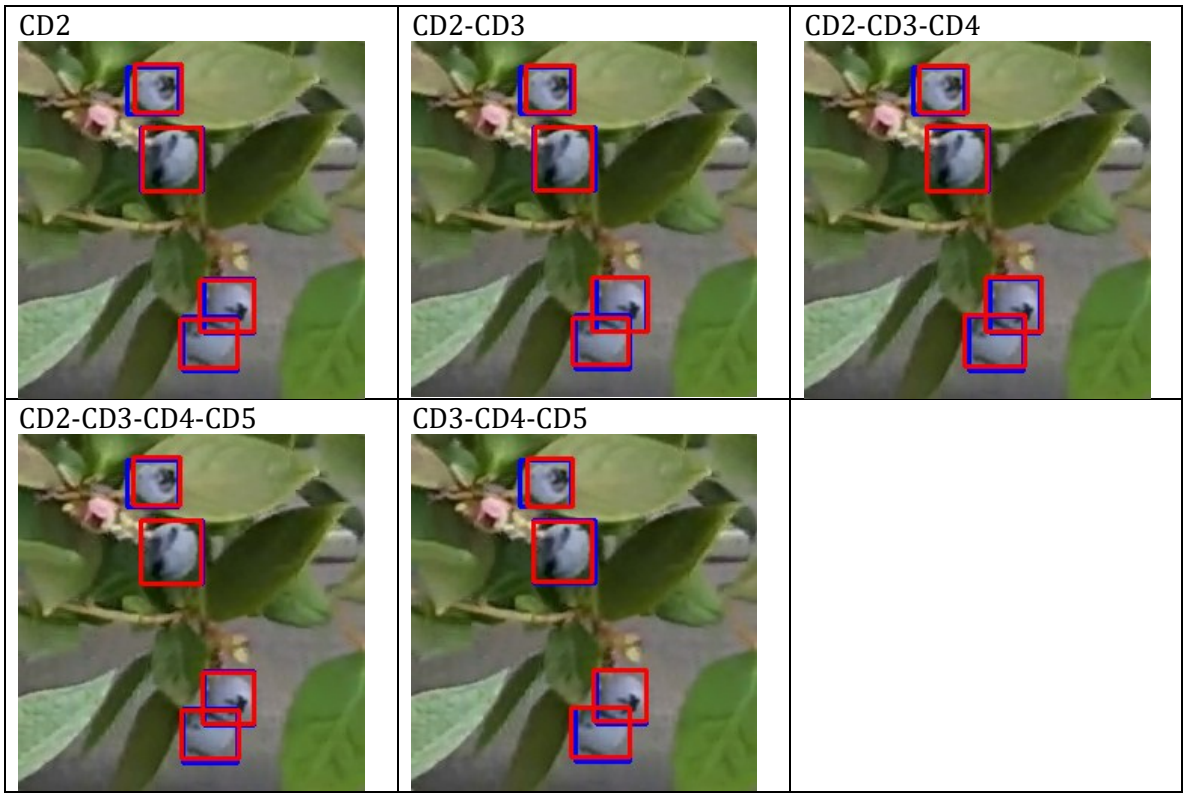

Ilustración 31: Resultados de detección en donde los 5 modelos presentan resultados cercanos al ground truth.

Ilustración 32: Errores de detección. "P2-P3" y "P2-P3-P4" presentan falsos positivos que pueden llevar a contar 2 veces un mismo arándano.

6.2) Rendimiento y resultados del algoritmo de conteo con arquitecturas de detección modificadas

Para permitir cuantificar y comparar el error entre las distintas pruebas donde el número de arándanos, parches de oclusión y la arquitectura utilizada varia, se decidió utilizar la métrica de MAPE (Mean absolute percentage error) ya que permite cuantificar el error absoluto medio (MAE) en un porcentaje sencillo de entender. Cabe destacar que según los resultados de la revisión sistemática de la literatura un 20% de los estudios utilizan el MAE para evaluar el conteo.

Como puede observarse en la tabla 11, si bien se calculan otras métricas como RSME, por las razones descritas anteriormente se analizan la mayoría de los resultados desde el punto de vista del MAPE. Dado esto se presentan los resultados en función del MAPE en la tabla 12.

Observando los resultados puede verse un patrón esperado, donde entre mayor oclusión exista mayor es el error de conteo, esto ya que dependiendo del nivel de oclusión que experimenta el arándano puede ser muy complicado para el detector localizarlo. Lo cual es cierto incluso para contadores humanos.

Otro resultado a notar es como aun que no exista oclusión alguna, igualmente podemos presenciar errores pequeños para todos los modelos de detección que fueron usados en conjunto al algoritmo de conteo. Esto puede explicarse ya que en todas las pruebas puede ocurrir la situación en donde 2 arándanos pueden ser generados muy cerca el uno del otro de forma que la oclusión parcial puede hacer que parezca un solo arándano en vez de 2. Lo cual puede ser apreciado en la Ilustración 32.

Ilustración 33: Error de detección, 2 arándanos individuales son detectaos como una unidad.

6.2.1) Punto de Corte en la Evaluación del Algoritmo

Analizando los resultados, se observa que, para la mayoría de las arquitecturas utilizadas en el conteo, el error MAPE es inferior al 5% cuando la oclusión es del 10% o menor. Asimismo, el error se mantiene por debajo del 10% cuando la oclusión alcanza el 20%, lo que representa resultados positivos en el contexto de la automatización del conteo de arándanos.

Por otro lado, aunque los resultados en escenarios con 50% de oclusión no son óptimos, es importante recordar que este experimento se llevó a cabo con fines exploratorios y de curiosidad científica, ya que este nivel de oclusión no es representativo de una plantación real.

De hecho, según lo observado durante la fase de elaboración del conjunto de datos, tal grado de oclusión no ocurre en la práctica.

En condiciones reales, el nivel de oclusión en una planta de arándanos es relativamente bajo. Esto se debe a que, al igual que en la mayoría de los arbustos que producen bayas, las flores (y posteriormente los frutos) se desarrollan en la parte exterior de la planta para facilitar la polinización. En general, la oclusión ocurre principalmente entre los racimos, cuando los frutos se superponen parcialmente entre sí o cuando las hojas de los racimos cubren parte de los arándanos.

Dado este comportamiento y los resultados positivos obtenidos en experimentos con niveles de oclusión del 10% o menores, se ha decidido utilizar estos valores de MAPE como referencia para evaluar el desempeño efectivo del algoritmo de conteo. En otras palabras, la evaluación del algoritmo se centrará en escenarios con oclusiones del 10% o menores, ya que estos representan condiciones más realistas y relevantes para el conteo automatizado de arándanos en aplicaciones prácticas.

6.2.2) Selección de arquitecturas con mejor desempeño

Selección de la primera arquitectura

Como puede observarse, una vez realizados todos los experimentos planeados, la arquitectura “CD3-CD4-CD5”, claramente consigue el error más pequeño para todas las condiciones de prueba excepto 1, la cual sería al evaluar 200 arándanos por pantalla sin oclusión alguna. Aun así, la diferencia del error entre “CD2” y “CD3-CD4-CD5” en esta instancia es muy pequeña, siendo 0.62% vs 0.75% o en términos de RMSE, 13.23 vs 21.61 arándanos, para una población total aproximada de 2000 arándanos en este experimento en particular.

Selección de la segunda arquitectura

Si bien la arquitectura con mejor desempeño se identifica con facilidad, determinar cuál ocupa el segundo lugar requiere un análisis más detallado de los resultados. A simple vista, las arquitecturas "CD2", "CD2-CD3-CD4" y "CD2-CD3-CD4-CD5" son las principales candidatas, ya que presentan resultados muy similares en múltiples escenarios.

En el caso donde hay 50 arándanos por pantalla, los resultados obtenidos por estas arquitecturas muestran diferencias menores al 0.5%, lo que indica un rendimiento prácticamente equivalente. Debido a esta similitud extrema, se hace necesario recurrir a la métrica RMSE para evaluar con mayor precisión las diferencias en el comportamiento de cada modelo.

Al profundizar en los datos, se observa que las variaciones entre estos modelos son mínimas en condiciones de oclusión entre el 2% y el 10%, con diferencias menores a un arándano. Sin embargo, en el escenario sin oclusión, la diferencia aumenta a dos arándanos. En general, "CD2-CD3-CD4-CD5" obtiene mejores resultados que "CD2", y al analizar los datos más allá del punto

de corte (10% de oclusión), se evidencia que la brecha de error entre ambas arquitecturas se amplía progresivamente, favoreciendo aún más a "CD2-CD3-CD4-CD5".

Por ejemplo, con un 10% de oclusión, "CD2-CD3-CD4-CD5" comete aproximadamente un error menos que "CD2". A medida que la oclusión aumenta, la diferencia se hace más notable: con 20% de oclusión, "CD2-CD3-CD4-CD5" comete cuatro errores menos, y con 50% de oclusión, la diferencia asciende a 11 errores menos respecto a "CD2".

En la comparación entre "CD2-CD3-CD4-CD5" y "CD2-CD3-CD4", se observa que "CD2-CD3-CD4-CD5" comienza a superar a "CD2-CD3-CD4" a partir del 5% de oclusión.

En los escenarios con 100 arándanos por pantalla, se mantiene la tendencia observada en los casos de menor densidad. En el rango de oclusión entre 0% y 5%, la arquitectura "CD2" muestra un mejor desempeño. Sin embargo, al alcanzar el 10% de oclusión, "CD2-CD3-CD4-CD5" empieza a superar progresivamente a "CD2", con una ventaja que se amplía a medida que aumenta la oclusión. La comparación entre "CD2-CD3-CD4-CD5" y "CD2-CD3-CD4" muestra el mismo patrón: "CD2-CD3-CD4" comienza a cometer más errores a partir del 10% de oclusión.

Finalmente, en el escenario con 200 arándanos por pantalla, se mantiene el mismo comportamiento. "CD2" sigue siendo superior en niveles de oclusión bajos, pero a partir del 10% de oclusión, "CD2-CD3-CD4-CD5" muestra un desempeño progresivamente mejor. Particularmente, al comparar "CD2-CD3-CD4-CD5" contra "CD2-CD3-CD4", se observa que la ventaja de "CD2-CD3-CD4-CD5" aparece incluso desde un 2% de oclusión, consolidando su superioridad en entornos más densos.

En términos generales, se puede concluir que la efectividad de "CD2-CD3-CD4-CD5" aumenta a medida que incrementa la oclusión. Incluso en escenarios con mínima oclusión, la diferencia en MAPE entre "CD2", "CD2-CD3-CD4-CD5" y "CD2-CD3-CD4" es extremadamente baja, lo que hace necesario recurrir a la métrica RMSE para distinguir las diferencias de desempeño con mayor precisión.

Desde una perspectiva práctica, en el contexto de esta tesis, donde el conteo de arándanos en escenarios de alta oclusión es un aspecto clave, y tomando en cuenta el punto de corte del 10% de nivel de oclusión, se considera que "CD2-CD3-CD4-CD5" es superior tanto a "CD2" como a "CD2-CD3-CD4", consolidándose como la segunda mejor opción para condiciones complejas.

Selección de la tercera arquitectura

Analizando las arquitecturas restantes, las principales candidatas a ocupar el tercer lugar en eficiencia para el conteo de arándanos son "CD2" y "CD2-CD3-CD4".

En el escenario con 50 arándanos por pantalla, se observa que cuando la oclusión está entre 0% y 2%, "CD2-CD3-CD4" comete aproximadamente 3 y 1 error menos que "CD2", respectivamente. Sin embargo, cuando la oclusión aumenta a 5% y 10%, la tendencia se invierte y "CD2" comete aproximadamente 1 y 2 errores menos que "CD2-CD3-CD4". Al analizar los datos más allá del punto de corte (10% de oclusión), "CD2-CD3-CD4" vuelve a mostrar un mejor desempeño,

reduciendo el número de errores en aproximadamente 1 y 4 errores menos que "CD2" para niveles de 20% y 50% de oclusión, respectivamente.

En la situación con 100 arándanos por pantalla, "CD2" obtiene un menor error en todo el rango de oclusión hasta el 10%, lo que significa que es la mejor opción en condiciones oclusión dentro del rango del punto de corte. Aun así, la diferencia entre ambos modelos es pequeña, sin superar los 3 arándanos en este rango. Sin embargo, cuando la oclusión alcanza 20% o 50%, "CD2-CD3-CD4" se convierte en la mejor opción, con una brecha de 5 y 12 errores menos, respectivamente.

Finalmente, en el escenario con 200 arándanos por pantalla, "CD2" mantiene un mejor desempeño en rangos de oclusión entre 0% y 5%, pero "CD2-CD3-CD4" reduce los errores de conteo a partir del 10% en adelante.

En general, se observa que "CD2" es más eficiente en condiciones de baja oclusión, logrando minimizar los errores cuando la oclusión es menor al 10%. Sin embargo, cuando el nivel de oclusión aumenta, "CD2-CD3-CD4" comienza a mejorar su desempeño, superando a "CD2" a partir del 20% de oclusión y convirtiéndose en la mejor opción en escenarios con alta densidad de obstáculos.

Dado que el punto de corte para evaluar el desempeño del algoritmo se estableció en 10% de oclusión, "CD2" es seleccionada como la tercera mejor arquitectura para el conteo de arándanos, ya que obtiene mejores resultados en el rango de oclusión considerado como relevante en el contexto de esta tesis.

Instancias por Pantalla	Nivel de Oclusión (%)	CD2	CD2-CD3	CD2-CD3-CD4	CD2-CD3-CD4-CD5	CD3-CD4-CD5
50	0	0,666667	0,2	0,133333	0,266667	**0,133333**
	2	1,004689	1,205762	1,005093	1,072028	**0,803752**
	5	2,4278	3,102158	2,764365	2,628594	**2,157619**
	10	4,885086	5,902588	5,291728	4,749033	**4,002533**
	20	9,580406	10,143	9,158314	8,734139	**7,327215**
	50	22,27447	23,35804	21,09264	19,37544	**18,09197**
100	0	0,3	0,566667	0,333333	0,433333	**0,266667**
	2	0,735313	1,43813	1,070088	1,136486	**0,836119**
	5	2,154882	3,198653	2,525253	2,592593	**2,020202**
	10	4,206164	5,197167	4,37707	4,035398	**3,557358**
	20	8,913503	9,764814	8,273363	7,91958	**7,031369**
	50	20,59741	21,12688	19,02075	18,58487	**16,86949**
200	0	**0,616667**	1,433333	0,933333	1,05	0,75
	2	1,270647	2,173372	1,621566	1,554799	**1,203679**
	5	2,660245	3,687348	2,794807	2,660271	**2,27285**
	10	5,212588	6,132188	5,093303	4,718216	**4,020562**
	20	9,664736	10,60412	8,924383	8,465259	**7,352108**
	50	21,53103	23,27982	20,23095	18,70567	**18,033**

Tabla 12: Métricas de MAPE para cada arquitectura.

6.2.3) Discusión de los resultados

Como se observó en la sección anterior, se determinó que el algoritmo con mejores resultados para el conteo de arándanos es, de manera indiscutible, aquel que utiliza el modelo de detección con la arquitectura "CD3-CD4-CD5". Observando los gráficos de la figura [X], podemos fácilmente apreciar cómo "CD3-CD4-CD5" muestra consistentemente valores significativamente más bajos respecto al porcentaje de error (MAPE) que el resto de los modelos. Su curva se mantiene más cercana al eje inferior para todos los experimentos realizados con 50, 100 y 200 muestras por pantalla.

En el análisis de los experimentos donde se agregaron de forma progresiva cabezas de detección para objetos de diferentes tamaños a la arquitectura de YOLOv8, se pudo observar que la inclusión de la cabeza de detección para objetos de gran tamaño "P5" parece ser muy beneficiosa para el conteo de arándanos. De hecho, los dos modelos que incorporan dicha cabeza de detección "CD3-CD4-CD5" y "CD2-CD3-CD4-CD5"– resultaron ser los dos con menor porcentaje de error de conteo (MAPE) en casi totalidad de los experimentos realizados.

Por el contrario, si nos fijamos en las otras arquitecturas que incorporan la cabeza de detección para objetos muy pequeños (aquellas que incluyen "CD2"), se observa que la gran mayoría de los resultados obtenidos son estrictamente inferiores a "CD3-CD4-CD5", sucediendo esto en el 94.44% de los experimentos. Esto a pesar de que en teoría contar con una capa de detección dedicada a escalas muy pequeñas (P2) podría mejorar la detección de arándanos (dado su reducido tamaño), en la práctica esto no ocurrió y terminó perjudicando los resultados de conteo.

Por tanto, considerando que "CD3-CD4-CD5" y "CD2-CD3-CD4- CD5" son los dos modelos con los mejores resultados, se puede concluir que la inclusión del mapa de características "P2" y su correspondiente cabeza de detección para objetos muy pequeños no produjo la mejora esperada en la detección de arándanos. En contraste, la incorporación de la cabeza para objetos de gran tamaño "CD5" sí conllevó una mejora en las métricas obtenidas, a pesar de tratarse de una clase de objeto físicamente pequeña.

Esto puede explicarse si consideramos el mecanismo de retroalimentación presente en arquitecturas de tipo PANet [19] (Path Aggregation Network) o las variaciones empleadas en YOLOv8. Estas redes no solo utilizan una ruta "top-down" (de capas profundas a capas superficiales), sino también conexiones "bottom-up" que reinyectan características refinadas a los niveles inferiores. De esta forma, la cabeza de detección de objetos grandes (CD5), a través de las métricas de perdida para clasificación y cuadro delimitador, permite no solo refinar la detección de objetos grandes, sino que además refinar los filtros de extracción de características para P5 (considerando la naturaleza end-to-end del entrenamiento de YOLOv8), lo que a través de PANet retroalimenta a niveles como P4 y P3 agregando información contextual de la imagen que se está observando. En la práctica, esto significa que una capa P5 más "refinada" contribuye a que P3 (que trabaja con escalas más pequeñas) herede mejores representaciones de las características, lo que se traduce en un rendimiento superior al contar arándanos, incluso cuando estos son de tamaño reducido.

Aunque a primera vista puede resultar confuso que una capa enfocada en objetos de gran tamaño contribuya a mejorar la detección de objetos pequeños, conviene recordar que P5 reduce la resolución de la imagen a tan solo 40×40 píxeles, con el objetivo de suprimir objetos muy pequeños y concentrarse en elementos de mayor tamaño, así como en el contexto general de la imagen.

Dado que en la detección de arándanos no existen realmente objetos de gran tamaño, la capa P5 puede concentrarse en extraer características del fondo, aportando información contextual a P4 y P3. De esta forma, por ejemplo, la capa P3 no solo se encarga de resaltar objetos de menor tamaño, sino que también mejora su capacidad para distinguir entre un objeto pequeño y el fondo, incrementando así la precisión en la detección de arándanos de pequeñas dimensiones.

En cuanto al bajo desempeño al agregar CD2, es importante recordar que P2 reduce la resolución de la imagen a 320×320, lo que produce una representación muy similar a la imagen original. Aunque esto facilita la detección de objetos muy pequeños, también permite que el modelo procese gran parte del ruido y detalles que en capas inferiores (como P3) serían suprimidos por la compresión de la imagen y los distintos filtros convolucionales que actúan sobre esta. Este exceso de información puede dificultar la detección efectiva, al confundir la diferenciación entre los arándanos y el fondo.

Por lo tanto, los resultados sugieren que añadir la cabeza de detección de objetos grandes (CD5) significa una mejora en la detección y el conteo de arándanos, mientras que la incorporación de la capa P2 no ofrece las ventajas esperadas para objetos muy pequeños. Esto evidencia la importancia de la estrategia de fusión de características en múltiples escalas (mediante algoritmos como FPN o PANet) y demuestra que un nivel más “alto” de la pirámide de características, aun enfocado a objetos de mayor tamaño, puede incrementar la calidad de las representaciones para escalas inferiores, contribuyendo de forma significativa en la tarea de contar arándanos.

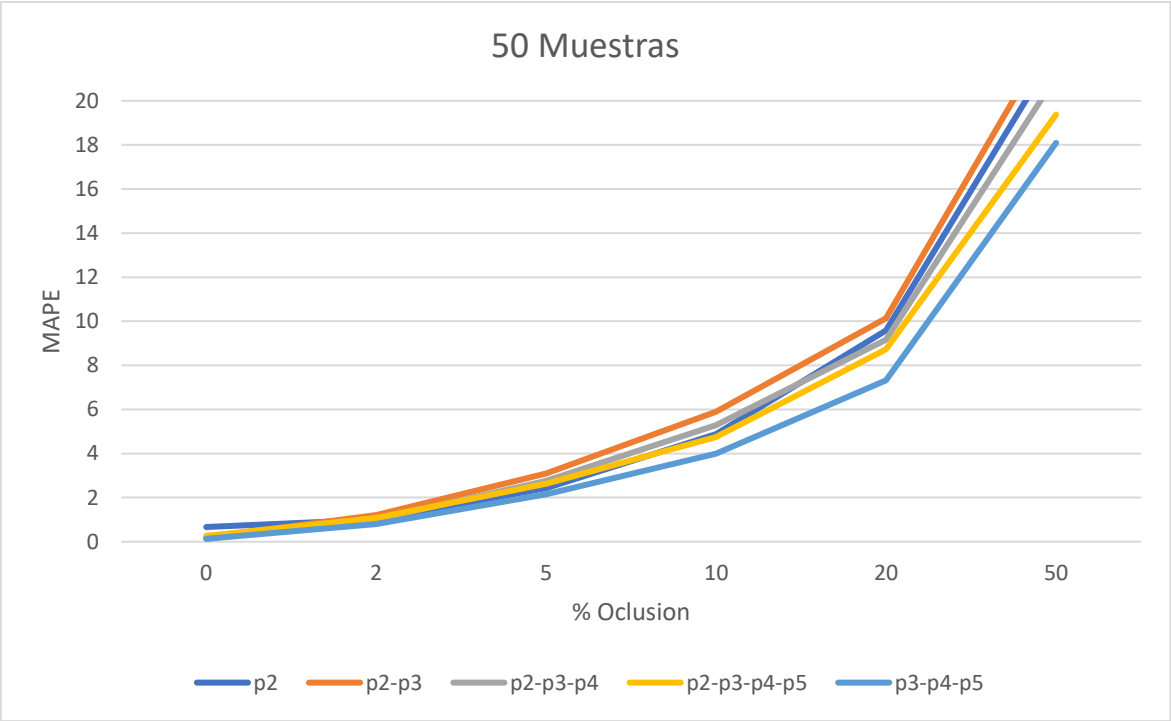

Tabla 13: Grafico de MAPE vs Oclusión para 50 muestras.

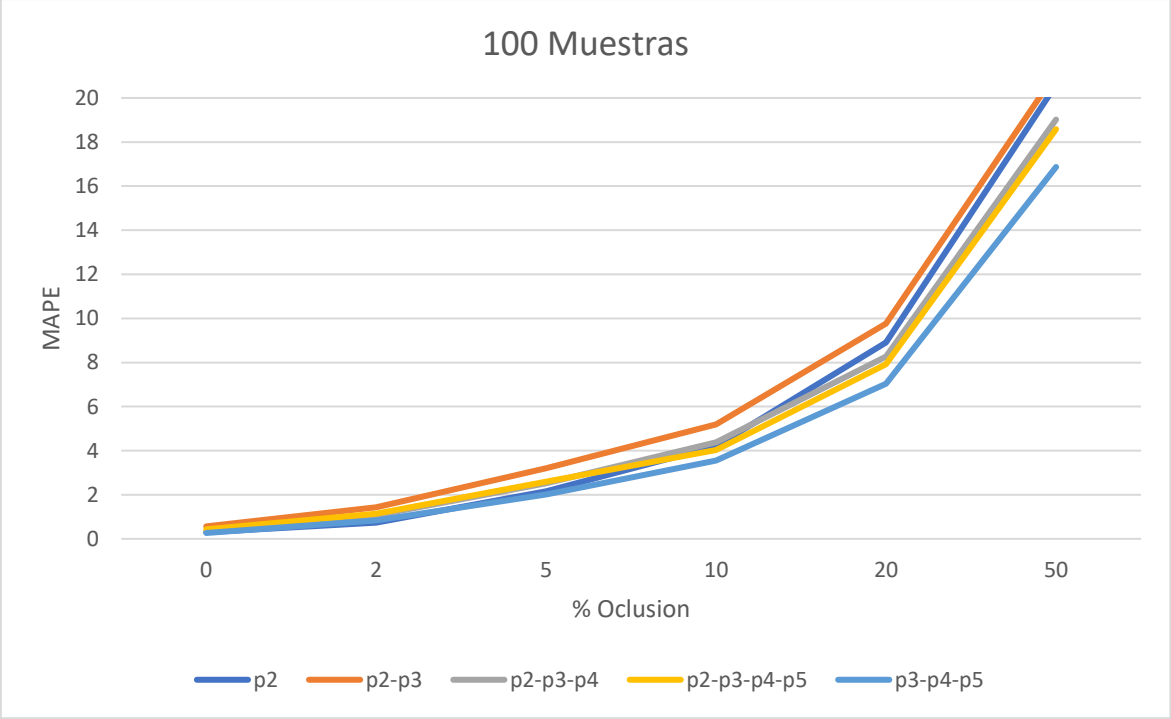

Tabla 14: Grafico de MAPE vs Oclusión para 100 muestras.

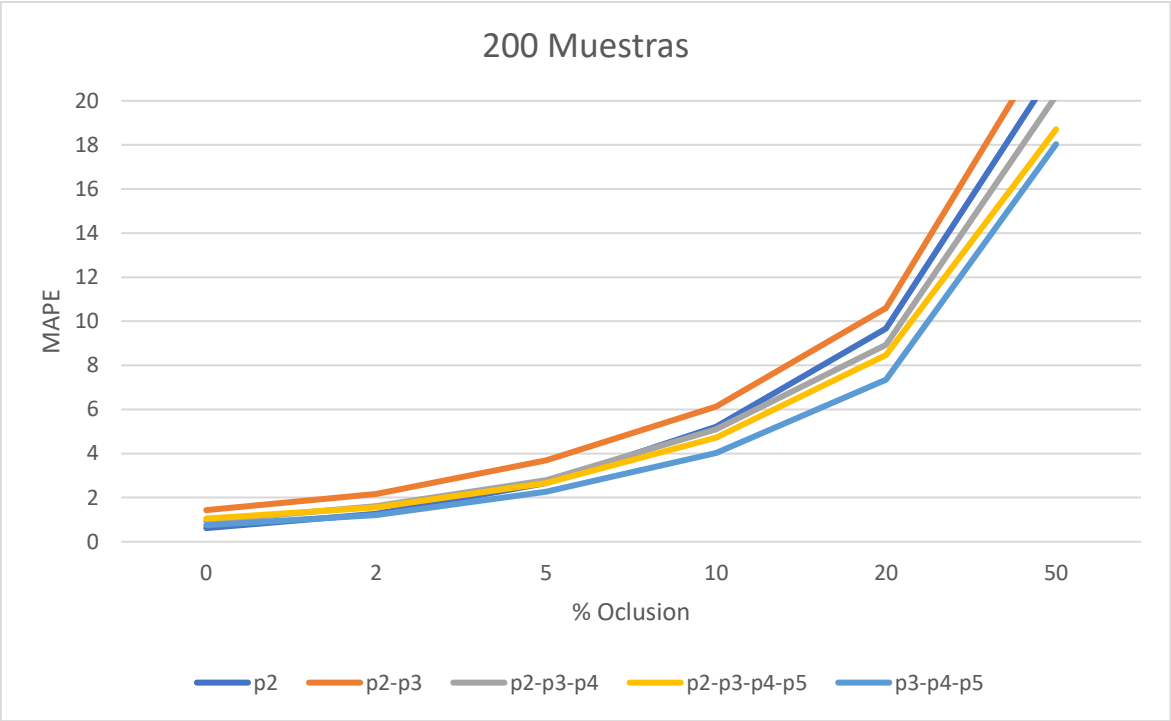

Tabla 15: Grafico de MAPE vs Oclusión para 200 muestras.

6.2.4) Comparación con otros algoritmos de conteo

Se llevó a cabo una comparación de resultados en función del MAPE con otros algoritmos de conteo de frutas descritos en la literatura. Dado que no todos los estudios proporcionan directamente esta métrica, el cálculo del MAPE se realizó con base en la información disponible sobre la cantidad de frutas señaladas en el ground truth y la cantidad detectada por los algoritmos propuestos en cada trabajo. En la tabla 16, se muestra la comparación de los resultados respecto al MAPE. Estudios que no proporcionan el numero bruto de frutas contadas o que no están relacionados al conteo de frutas no fueron considerados para esta comparación. En el siguiente párrafo, a continuación de esta tabla, se explica cómo se calculó el MAPE para cada estudio en particular.

Algoritmo de Conteo	Fruta de interés	MAPE
Leiying He et al. (2022) [38]	Camelias	15.79%
	Manzanas	5.55%
Zhang et al. (2022)[35]	Naranjas	5.28%
F. Gao et al. (2022) [39]	Manzanas	8.40%
MUHAMMAD RIZWAN et al. (2023) [40]	Inflorescencias de uva	4.53%
Zhang et al. (2024) [41]	Naranjas	4.36%
Algoritmo propio 1 (CD3-CD4-CD5)	**Arándanos**	**3.86%**
Algoritmo propio 2 (CD2-CD3-CD4-CD5)	Arándanos	4.5%
Algoritmo propio 3 (CD2)	Arándanos	4.77%

Tabla 16: Comparación de MAPE para algoritmos de conteo encontrados en la literatura vs los algoritmos desarrollados.

En el primer estudio de Leiying He et al. (2022) , se abordan los conteos de frutos en plantas de Camelia y manzanas. Para las Camelias, se tomó la información presentada en la tabla 2, donde se mencionan 44 frutos contados por el algoritmo y 38 frutos en el ground truth. Con esos datos se calculó el MAPE. Para el caso de las manzanas, se recurrió a la tabla 6, que muestra el número de manzanas detectadas y la cantidad real de manzanas en los tres videos analizados.

El segundo estudio, Zhang et al. (2022) , se dedica al conteo automático de naranjas. Este trabajo ya incluye la métrica MAPE en la tabla 5, por lo que se tomó directamente ese valor para la comparación.

El tercer estudio, F. Gao et al. (2022) , se enfoca en el conteo de manzanas. Se utilizó la figura 10 para obtener el número de manzanas contabilizadas y la cantidad registrada en el ground truth

para cinco videos, calculando a partir de ahí el MAPE. A pesar de que en la parte final del estudio se incluyen 20 videos, no se especifican los valores exactos de manzanas detectadas por el algoritmo para ese conjunto mayor, así que no fue posible calcular el MAPE adicionalmente.

El cuarto estudio, MUHAMMAD RIZWAN et al. (2023), presenta información sobre la cantidad de inflorescencias de uva detectadas y su ground truth en la tabla 5, lo que permite estimar el MAPE a partir de los cinco videos que se describen en ese apartado.

En cuanto al quinto estudio, Zhang et al. (2024), se tomó directamente el MAPE indicado en el resumen (abstract) del artículo.

Para calcular el MAPE de los algoritmos propios, se utilizó el promedio obtenido a partir de los escenarios de simulación con un 10% de oclusión, que es el punto de corte seleccionado en esta investigación. Por ejemplo, para determinar el MAPE final de "Algoritmo propio 1 (CD3-CD4-CD5)", se consideraron los valores resultantes del 10% de oclusión en las simulaciones con 50, 100 y 200 arándanos por pantalla, y se calculó el promedio de esos tres resultados. El mismo criterio se aplicó a "Algoritmo propio 2 (CD2-CD3-CD4-CD5)" y "Algoritmo propio 3 (CD2)".

Como se observa, las métricas obtenidas por los algoritmos desarrollados resultan competitivas frente a otras propuestas encontradas en la literatura. En particular, el Algoritmo propio 1 (CD3-CD4-CD5) destaca por alcanzar un MAPE de 3.86%, un valor inferior al de algunos trabajos previos. Los otros dos algoritmos desarrollados presentan resultados comparables, con un MAPE de 4.5% y 4.77%, respectivamente.

Estos hallazgos indican que los modelos propuestos ofrecen un desempeño competitivo y, en ciertos casos, superan a algoritmos previamente reportados. Además, sugieren que los modelos son capaces de realizar predicciones con alta precisión, reduciendo el margen de error en comparación con otros algoritmos existentes.

Conclusiones

La evidencia reunida en esta tesis sugiere que modificar la arquitectura de YOLOv8 para mejorar la detección de objetos pequeños, mediante la incorporación de la cabeza de detección de objetos muy pequeños (CD2), no beneficia al rendimiento del detector de arándanos. El motivo principal radica en que el mapa de características P2 consiste en un mapa de alta resolución muy similar a la imagen original, lo que introduce ruido y afecta negativamente las métricas de detección. En contraste, la inclusión de la cabeza de detección de objetos grandes (CD5), junto con su mapa de características P5, favorece la detección de objetos pequeños. Esto sucede porque la red aprende a distinguir mejor el fondo de la imagen y, gracias a la estructura PANet, puede refinar la información en las capas especializadas en objetos pequeños. Así, la capa P5 aporta un contexto que ayuda a diferenciar eficazmente los arándanos del fondo, permitiendo una detección más precisa incluso en presencia de oclusiones o complejidades en la escena.

En esta tesis también se desarrolló un programa de simulación que permite ajustar diversos parámetros para recrear las condiciones de presencia de fruta y oclusión que se encuentran en una plantación real. Este programa facilita la evaluación de las modificaciones realizadas a la arquitectura de YOLOv8 durante el desarrollo de esta investigación, permitiendo probar de manera sencilla distintos enfoques para mejorar el conteo de arándanos. Con el objetivo de contribuir al avance en este campo, el programa de simulación será compartido en un repositorio público, proporcionando una herramienta accesible para futuros investigadores interesados en evaluar diferentes métodos de conteo y algoritmos de detección que desarrollen.

Para las simulaciones realizadas, el algoritmo de conteo de arándanos desarrollado durante esta tesis, junto con las tres modificaciones en la arquitectura YOLOv8 del modelo de detección de objetos, alcanza métricas de desempeño similares a las presentadas por distintos algoritmos de conteo de frutas en la literatura. En particular, el algoritmo que emplea las cabezas de detección para CD3-CD4-CD5 consigue resultados que incluso superan los reportados anteriormente, lo cual es aún más relevante si se considera que el problema de conteo de arándanos aborda racimos con frutas pequeñas que suelen ocluirse parcialmente entre sí. Esto contrasta con los estudios previos, enfocados mayoritariamente en frutas de gran tamaño, donde la visibilidad se mantiene amplia aun cuando existe cierto nivel de oclusión.

Otro aspecto destacado en esta tesis es la importancia de adaptar los algoritmos de detección al dominio específico del problema. En el caso de los modelos basados en la arquitectura YOLOv8, fue necesario modificar la capa de entrada de la red, duplicando su tamaño original para ajustarse al tamaño de los arándanos. De no haberlo hecho, la configuración por defecto habría arrojado resultados poco coherentes. Asimismo, se evaluaron diferentes configuraciones de “cabezas de entrada” para determinar la que mejor se ajustaba a las necesidades de la tarea de detección y conteo de arándanos.

Dicho esto, y dado los resultados positivos de los experimentos, se considera que se ha cumplido con los objetivos de la tesis.

Bibliografía

[1] «“Global state of the Blueberry - Industry Report 2023”», Blueberry International Organization. Accedido: 24 de junio de 2024. [En línea]. Disponible en: https://www.internationalblueberry.org/2023-report/

[2] Mundoagro, «Los desafíos de la campaña 2023-2024 de arándanos chilenos - Mundoagro». Accedido: 24 de junio de 2024. [En línea]. Disponible en: https://mundoagro.cl/los-desafios-de-la-campana-2023-2024-de-arandanos-chilenos/

[3] Mundoagro, «Transformación de la agricultura chilena de la mano del Internet de las Cosas - Mundoagro». Accedido: 30 de marzo de 2025. [En línea]. Disponible en: https://mundoagro.cl/transformacion-de-la-agricultura-chilena-de-la-mano-del-internet-de-las-cosas/

[4] MUNDOAGRO, «Buscan incorporar Inteligencia Artificial para el conteo de fruta fresca de exportación - Mundoagro». Accedido: 30 de marzo de 2025. [En línea]. Disponible en: https://mundoagro.cl/buscan-incorporar-inteligencia-artificial-para-el-conteo-de-fruta-fresca-de-exportacion/

[5] E. Elyan *et al.*, «Computer vision and machine learning for medical image analysis: recent advances, challenges, and way forward», *Art Int Surg*, 2022, doi: 10.20517/ais.2021.15.

[6] Samahitha Kaliyuru Ravi, Sameera Kaliyuru Ravi, y A. Hema Prabha, «Advent of machine learning in autonomous vehicles», *Int. J. Sci. Res. Arch.*, vol. 13, n.º 1, pp. 1219-1226, sep. 2024, doi: 10.30574/ijsra.2024.13.1.1760.

[7] F. Alvi, «Computer Vision and Image Processing: Understanding the Distinction and Interconnection», OpenCV. Accedido: 31 de marzo de 2025. [En línea]. Disponible en: https://opencv.org/blog/computer-vision-and-image-processing/

[8] E. F. Morales y H. J. Escalante, «A brief introduction to supervised, unsupervised, and reinforcement learning», en *Biosignal Processing and Classification Using Computational Learning and Intelligence*, Elsevier, 2022, pp. 111-129. doi: 10.1016/B978-0-12-820125-1.00017-8.

[9] F. Rosenblatt, «The perceptron: A probabilistic model for information storage and organization in the brain.», *Psychological Review*, vol. 65, n.º 6, pp. 386-408, 1958, doi: 10.1037/h0042519.

[10] D. H. Hubel y T. N. Wiesel, «Receptive fields of single neurones in the cat’s striate cortex», *The Journal of Physiology*, vol. 148, n.º 3, pp. 574-591, oct. 1959, doi: 10.1113/jphysiol.1959.sp006308.

[11] D. H. Hubel y T. N. Wiesel, «Receptive fields and functional architecture of monkey striate cortex», *The Journal of Physiology*, vol. 195, n.º 1, pp. 215-243, mar. 1968, doi: 10.1113/jphysiol.1968.sp008455.

[12] K. O'Shea y R. Nash, «An Introduction to Convolutional Neural Networks», 2015, *arXiv*. doi: 10.48550/ARXIV.1511.08458.

[13] A. Krizhevsky, I. Sutskever, y G. E. Hinton, «ImageNet Classification with Deep Convolutional Neural Networks», en *Advances in Neural Information Processing Systems*, F. Pereira, C. J. Burges, L. Bottou, y K. Q. Weinberger, Eds., Curran Associates, Inc., 2012. [En línea]. Disponible en: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[14] «Imagenet Challenge - an overview | ScienceDirect Topics». Accedido: 29 de marzo de 2025. [En línea]. Disponible en: https://www.sciencedirect.com/topics/computer-science/imagenet-challenge

[15] «ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC2012)». Accedido: 29 de marzo de 2025. [En línea]. Disponible en: https://image-net.org/challenges/LSVRC/2012/results.html

[16] K. He, X. Zhang, S. Ren, y J. Sun, «Deep Residual Learning for Image Recognition», 2015, *arXiv*. doi: 10.48550/ARXIV.1512.03385.

[17] P. Viola y M. Jones, «Rapid object detection using a boosted cascade of simple features», en *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, dic. 2001, p. I-I. doi: 10.1109/CVPR.2001.990517.

[18] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, y S. Belongie, «Feature Pyramid Networks for Object Detection», 2016, *arXiv*. doi: 10.48550/ARXIV.1612.03144.

[19] S. Liu, L. Qi, H. Qin, J. Shi, y J. Jia, «Path Aggregation Network for Instance Segmentation», 2018, *arXiv*. doi: 10.48550/ARXIV.1803.01534.

[20] R. Girshick, J. Donahue, T. Darrell, y J. Malik, «Rich feature hierarchies for accurate object detection and semantic segmentation», 2013, *arXiv*. doi: 10.48550/ARXIV.1311.2524.

[21] R. Girshick, «Fast R-CNN», 2015, *arXiv*. doi: 10.48550/ARXIV.1504.08083.

[22] S. Ren, K. He, R. Girshick, y J. Sun, «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks», 2015, *arXiv*. doi: 10.48550/ARXIV.1506.01497.

[23] W. Liu *et al.*, «SSD: Single Shot MultiBox Detector», 2015, doi: 10.48550/ARXIV.1512.02325.

[24] J. Redmon, S. Divvala, R. Girshick, y A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection», 2015, *arXiv*. doi: 10.48550/ARXIV.1506.02640.

[25] A. Dosovitskiy *et al.*, «An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale», 2020, *arXiv*. doi: 10.48550/ARXIV.2010.11929.

[26] V. Miles, F. Gurr, y S. Giani, «Camera-Based System for the Automatic Detection of Vehicle Axle Count and Speed Using Convolutional Neural Networks», *Int. J. ITS Res.*, vol. 20, n.º 3, pp. 778-792, dic. 2022, doi: 10.1007/s13177-022-00325-1.

[27] R. E. Kalman, «A New Approach to Linear Filtering and Prediction Problems», *Journal of Basic Engineering*, vol. 82, n.º 1, pp. 35-45, mar. 1960, doi: 10.1115/1.3662552.

[28] A. Bewley, Z. Ge, L. Ott, F. Ramos, y B. Upcroft, «Simple online and realtime tracking», en *2016 IEEE International Conference on Image Processing (ICIP)*, Phoenix, AZ, USA: IEEE, sep. 2016, pp. 3464-3468. doi: 10.1109/ICIP.2016.7533003.

[29] N. Wojke, A. Bewley, y D. Paulus, «Simple Online and Realtime Tracking with a Deep Association Metric», 2017, *arXiv*. doi: 10.48550/ARXIV.1703.07402.

[30] Y. Du *et al.*, «StrongSORT: Make DeepSORT Great Again», 2022, *arXiv*. doi: 10.48550/ARXIV.2202.13514.

[31] H. Luo *et al.*, «A Strong Baseline and Batch Normalization Neck for Deep Person Re-Identification», *IEEE Trans. Multimedia*, vol. 22, n.º 10, pp. 2597-2609, oct. 2020, doi: 10.1109/TMM.2019.2958756.

[32] X. Liu *et al.*, «Monocular Camera Based Fruit Counting and Mapping With Semantic Data Association», *IEEE Robotics and Automation Letters*, vol. 4, n.º 3, pp. 2296-2303, jul. 2019, doi: 10.1109/LRA.2019.2901987.

[33] M. Tkachenko, M. Malyuk, A. Holmanyuk, y N. Liubimov, «Label Studio: Data labeling software». 2020. [En línea]. Disponible en: https://github.com/heartexlabs/label-studio

[34] G. Jocher, J. Qiu, y A. Chaurasia, *Ultralytics YOLO*. (enero de 2023). [En línea]. Disponible en: https://github.com/ultralytics/ultralytics

[35] Z. Zheng *et al.*, «An efficient online citrus counting system for large-scale unstructured orchards based on the unmanned aerial vehicle», *Journal of Field Robotics*, vol. 40, n.º 3, pp. 552-573, may 2023, doi: 10.1002/rob.22147.

[36] G. Jocher, «YOLOv8 P2 and P6 variants for small object detection». [En línea]. Disponible en: https://github.com/orgs/ultralytics/discussions/8227#discussioncomment-8485540

[37] Z. Zheng *et al.*, «An efficient online citrus counting system for large-scale unstructured orchards based on the unmanned aerial vehicle», *Journal of Field Robotics*, vol. 40, n.º 3, pp. 552-573, may 2023, doi: 10.1002/rob.22147.

[38] L. He, F. Wu, X. Du, y G. Zhang, «Cascade-SORT: A robust fruit counting approach using multiple features cascade matching», *Computers and Electronics in Agriculture*, vol. 200, 2022, doi: 10.1016/j.compag.2022.107223.

[39] F. Gao *et al.*, «A novel apple fruit detection and counting methodology based on deep learning and trunk tracking in modern orchard», *Computers and Electronics in Agriculture*, vol. 197, 2022, doi: 10.1016/j.compag.2022.107000.

[40] M. R. Khokher *et al.*, «Early Yield Estimation in Viticulture Based on Grapevine Inflorescence Detection and Counting in Videos», *IEEE Access*, vol. 11, pp. 37790-37808, 2023, doi: 10.1109/ACCESS.2023.3263238.

[41] Z. Zheng *et al.*, «A robust and efficient citrus counting approach for large-scale unstructured orchards», *Agricultural Systems*, vol. 215, p. 103867, mar. 2024, doi: 10.1016/j.agsy.2024.103867.