UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

METODOLOGÍAS MODERNAS DE APRENDIZAJE PROFUNDO PARA LA DETECCIÓN DE FRUTA

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA
MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

LUIS GUSTAVO COSSIO MONTEFINALE

PROFESOR GUÍA:
JAVIER RUIZ-DEL-SOLAR SAN MARTÍN
PROFESOR CO-GUÍA
RODRIGO VERSCHAE TANNENBAUM

MIEMBROS DE LA COMISIÓN:
FELIPE ARTURO TOBAR HENRÍQUEZ
IVÁN ANSELMO SIPIRÁN MENDOZA

Este trabajo ha sido parcialmente financiado por:
Fondo FONDECYT 1201170
Fondo FONDEQUIP EQM170041

SANTIAGO DE CHILE
2023

RESUMEN DE LA TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA,
MENCIÓN ELÉCTRICA Y MEMORIA PARA OPTAR AL
TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: LUIS GUSTAVO COSSIO MONTEFINALE
FECHA: 2023
PROF. GUÍA: JAVIER RUIZ DEL SOLAR

METODOLOGÍAS MODERNAS DE APRENDIZAJE PROFUNDO PARA LA DETECCIÓN DE FRUTA

Esta tesis presenta una revisión del estado del arte en detección de frutas, considerando los distintos tipos de algoritmos, metodologías y tipos de frutos estudiadas. También se realiza un análisis del estado del arte en detección general de objetos, enfocado en los algoritmos de aprendizaje profundo y las metodologías de entrenamiento usadas en estos. A partir de estos análisis se observan ciertas diferencias entre las dos áreas, principalmente en la ausencia de ciertos algoritmos y arquitecturas que no han sido usadas en detección de fruta, a pesar de su prevalencia en detección de objetos. Para validar distintos algoritmos y arquitecturas se realiza una comparación exhaustiva de metodologías modernas en detección de objetos y metodologías tradicionales en detección de fruta. Los algoritmos fueron evaluados en dos bases de datos de frutas. La primera es MinneApple, una base de datos pública de manzanas. En segundo lugar, se construyó una base de datos de cerezas, llamada Cherry CO, para tareas de detección y segmentación de objetos. Se observó que redes modernas como Swin, YOLOR y Scaled-YOLOv4 ofrecen mejoras significativas, así como los métodos de aumento de datos Copy-Paste y Mosaic ofrecen mejoras consistentes, principalmente en términos de precisión y recall.

A mi padre

Agradecimientos

Este trabajo de tesis no habría sido posible sin el apoyo de distintas instituciones que me facilitaron recursos para investigar, por ello quiero agradecer el financiamiento del fondo FONDECYT 1201170, y al fondo FONDEQUIP EQM170041 por el acceso a equipos. De igual modo quiero reconocer la ayuda de proyecto FIC: Transferencia y Adopción de Tecnologías para la Gestión de Riesgo en el Proceso Productivo de la Cereza y al Centro de Estudios Avanzados en Fruticultura por apoyarme en el proceso de adquisición de datos.

Por sobre todo quiero agradecer a mi familia por el apoyo incondicional que me han dado a lo largo de mi tiempo en la universidad, no ha sido fácil y este apoyo ha sido invaluable. En segundo lugar, quisiera agradecer a la Universidad de Chile y a sus docentes, llegue con altas expectativas de formarme y lograr desarrollar todo tipo de proyectos y puedo decir que salgo listo para empezar estos. En este aspecto tengo que resaltar al profesor Javier Ruiz del Solar, que ha sido una gran influencia, un gran docente y guía.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

La tarea de detección y segmentación de objetos en imágenes ha adquirido cada vez más relevancia dado los altos niveles de precisión, velocidad y versatilidad que se han logrado, así como el gran número de aplicaciones que se pueden desarrollar gracias a estas. En la medida en la que se busca desarrollar todo tipo de aplicaciones semiautónomas o autónomas, el reconocimiento de objetos resulta un recurso fundamental para muchas de ellas. Una de las primeras aplicaciones fue el desarrollo de métodos de detección de rostros, como parte de algoritmos de identificación de personas en sistemas de seguridad. Otros ejemplos de aplicaciones basadas en detección son detección temprana de tumores cerebrales en base a imágenes de resonancias magnéticas, seguimiento de autos en videos de tráfico y también hay proyectos prometedores que buscan implementar vehículos autónomos en el futuro cercano basados en detección de peatones, autos y demás objetos que se pueden encontrar en zonas urbanas.

Para realizar las tareas de detección, y procesamiento de imágenes en general, se han empleado distintos algoritmos como detección basadas en características Haar Cascade o análisis de características SURF [1], sin embargo la familia de algoritmos que más éxito ha tenido en la última década son las Redes Neuronales (Neural Networks - NN), que son un tipo de algoritmo de aprendizaje de máquina, es decir, algoritmos cuyos parámetros se estiman usando datos de entrenamiento. Las redes neuronales se han aplicado a diversidad de problemas como análisis estadístico, robótica móvil y distintas aplicaciones en el procesamiento de imágenes, como clasificación de imágenes en distintas categorías, detección de objetos y generación de imágenes, entre otras. En el área de Visión Computacional el tipo de red más común son las Redes Neuronales Convolucionales (Convolutional Neural Network - CNN), dada su capacidad de inferir e interpretar información visual.

Dado su potencial no es sorpresa que la mayoría de las industrias hayan adoptado el uso de NN's en mayor o menor medida. Sin embargo, su adopción en el sector agrícola ha sido escasa. Esto se debe a las irregularidades presentes en ambientes agrícolas y a las dificultades que implica traer este tipo de tecnologías a instalaciones de este tipo. Esto hace desafiante aplicar cualquier sistema que sea robusto a las diversas condiciones que pueden aparecer y que simultáneamente presenta ventajas al trabajo con métodos convencionales. Por lo que una parte importante de las tareas agrícolas se desarrollan de forma manual o con maquinaria sencilla como tractores o aspersores. Este acercamiento aumenta los costos y el tiempo de ejecución de muchas tareas.

Muchos estudios han demostrado la viabilidad y alcance de múltiples técnicas de procesamiento de imagen en agricultura [2–4], también conocida como Agrovisión. Distintos autores han desarrollado algoritmos para una gran variedad de tareas, como monitoreo de estado de salud de frutos y plantas, detección de plagas y enfermedades, detección de frutos o semillas para tareas como conteo, cosechado o recolección.

Si bien estos métodos han demostrado resultados impresionantes en términos de precisión, velocidad y robustez, en una variedad de aplicaciones, aún hay espacio de mejora. Por lo que esta tesis busca desarrollar una serie de metodologías para la tarea de **Detección de Frutas** (DF), haciendo una comparación comprensiva entre distintas redes y metodologías de entrenamiento.

1.1. Detección y Segmentación de Objetos

La Detección de Objetos (DO) es el área de la visión computacional que busca poder caracterizar la ubicación de un objeto dentro de una imagen, donde a menudo esta ubicación se define como un contorno rectangular o Bounding Box (BB) que encierra un objeto de interés (ver Fig. 1.1 a)). Otras formas de detección también pueden incluir otros polígonos o algún otro tipo de información, pero lo más común es el uso de BB's.

Los algoritmos de detección de objetos aprenden a predecir las coordenadas de objetos en imágenes. Durante el entrenamiento requieren de múltiples muestras, donde cada muestra se compone de dos datos, una imagen I y una lista de las ubicaciones de cada objeto en dicha imagen. Las ubicaciones de cada objeto se definen como 4 coordenadas de la forma $o = ((x_1, y_1), (x_2, y_2))$. Potencialmente se puede especificar una categoría c por cada objeto, $o = ((x_1, y_1), (x_2, y_2), c)$, para distinguir objetos de distinta naturaleza, como perros de personas.

(a) (b) (c) (d)

Figura 1.1: Representaciones de objetos. a) Imagen original. b) Imagen con BB's. c) Imagen con Segmentaciones sobrepuestas. d) Máscara de segmentos.

Un detector es capaz de generar predicciones a partir de una imagen de la forma:

$$F(I) = ((x^*_{1j}, y^*_{1j}), (x^*_{2j}, y^*_{2j}), c^*_j)$$

Un buen detector es capaz de generar predicciones similares a las ubicaciones reales de objetos presentes en la imagen y que puedan clasificar correctamente dichos objetos.

Una tarea relacionada es la segmentación de objetos, que en lugar de definir una ubicación como un BB, define la pertenencia de un conjunto de píxeles en una imagen a un objeto, lo que permite detallar la forma de los objetos en la imagen de una forma mucho más precisa que con un BB. Dado que los algoritmos de segmentación requieren predecir segmentos, estos no utilizan un BB como dato de entrenamiento, sino que usan una máscara M, que define una representación de los segmentos de la imagen pixel a pixel (ver Fig. 1.1 d)).

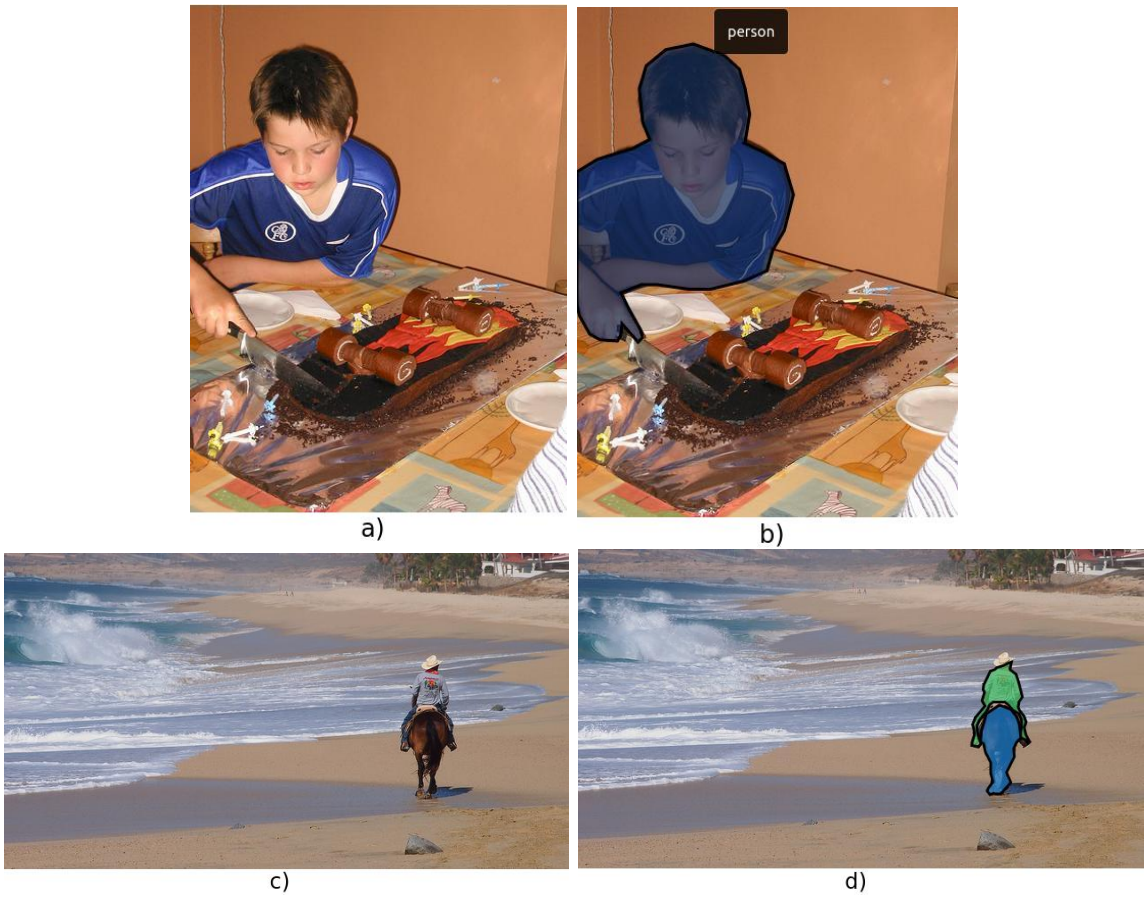


Figura 1.2: Ejemplos de imágenes de COCO. a) Imagen 1. b) Máscara 1. c) Imagen 2. d) Mascara 2

1.1.1. COCO Dataset

Dentro del área de detección de objetos existe una variedad de bases de datos públicas que sirven para poder comparar resultados entre distintos autores de forma imparcial. Estas bases de datos varían en tamaño, dimensiones y tareas que se buscan evaluar. Sin embargo, la base de datos más importante en el área de Detección de Objetos es la base de datos de Microsoft Common Objects in Context, también conocido como COCO [5]. La razón de

su importancia es la alta calidad en múltiples aspectos. En primer lugar, se compone de 164.000 imágenes de entrenamiento y 82.000 para tareas de validación y prueba/test, las cuales fueron extraídas de internet. Otro aspecto de gran importancia de COCO es su gran número de clases. En total existen 80 clases distintas de objetos presentes en las imágenes, tales como personas, perros, trenes, autos, botellas, caballos, bicicletas, etc. La base de datos de COCO está compuesto por etiquetas tanto para detección como para segmentación de objetos, por lo que sirve para evaluar ambos tipos de tareas. Esto permite que COCO sea una base de datos extremadamente desafiante, que tiene objetos en todo tipo de situaciones, tamaños y contextos. En la Fig. 1.2 se pueden observar dos ejemplos de COCO y sus segmentos.

En la actualidad esta base de datos se utiliza en la competencia MS-COCO dev-test donde distintos autores pueden comparar sus resultados de forma imparcial. Se ha observado que las redes neuronales y en particular las CNN's son los algoritmos que muestran el mejor desempeño para la detección y segmentación en COCO.

1.2. Automatización en Agricultura y Agrovisión

Históricamente, cosechar ha sido una actividad manual, con un uso acotado de maquinaria. El uso de computadores, grandes servidores, rovers o drones no parece tener lugar en la concepción general de una plantación. Sin embargo, ha habido una tendencia a mayor automatización y mayor uso de sistemas autónomos en cada vez más áreas de la producción agrícola. Ejemplos de estos son el uso de drones para la evaluación del estado de cultivos [3, 6], el uso de sensores para la medición activa de variables relevantes [6–8], tales como la humedad y la temperatura, así como la incorporación de sistemas inteligentes capaces de automatizar tareas como la irrigación, aplicación de fertilizantes, eliminación de hiervas y cosechado [9–12]. Si bien el alcance de estas tecnologías sigue siendo escaso, ya ha empezado un proceso de incorporación de estas en la industria. Muchos de estos avances se basan en distintas herramientas de procesamiento de imágenes, dado que la información visual de una planta es suficiente para determinar aspectos como su estado de madurez, su nivel hídrico, altura, masa cosechable, entre otras informaciones útiles. Por lo mismo el procesamiento de imágenes en agricultura, también conocido como Agrovisión, ha recibido mucha atención por parte de industrias e investigadores [2–4].

Existen distintas aplicaciones en Agrovisión [4], como monitoreo de la salud de salud y madurez, donde se utilizan cámaras, ya sea fijas o acopladas a robots para observar plantas o frutas, luego se utiliza un algoritmo para procesar dichas imágenes y extraer información como altura, estado hídrico, madurez, entre otros. Otra área de interés es la prevención y detección de enfermedades o plagas, donde se han desarrollado métodos para clasificar y/o detectar estas anomalías en frutas [13], árboles o hojas [14]. Finalmente existe el área de detección de material agrícola, como frutas, semillas, hierbas o flores. El caso de más interés ha sido la detección de fruta, donde se han investigado distintas aplicaciones de la Visión Computacional, tales como detección de fruta [15–17], conteo de fruta [18–20] y hasta aplicaciones para recolección [10, 11, 21].

1.3. Motivación

En la actualidad muchas tareas agrícolas como el conteo de fruta en una plantación o la recolección de ésta, entre otras, son tareas realizadas manualmente y por varias personas, lo que es lento y costoso, por lo que existe un gran interés en sistemas robustos que puedan realizar estas tareas. En términos generales, el conocimiento de la posición, cantidad o estado de las frutas tiene innumerables aplicaciones y el potencial de abaratar costos y mejorar la calidad de la fruta.

Distintos estudios han utilizado CNN's para la tarea de DF, tales como manzanas, naranjas, uvas, pimientos o tomates. El uso de CNN's ha demostrado tener altos niveles de precisión y velocidad, sin embargo aún existe margen para mejorar usando redes de detección en el estado del arte y usando distintas metodologías de entrenamiento. Para evidenciar el potencial de este acercamiento se desea evaluar las distintas metodologías y redes en distintas bases de datos desafiantes.

1.4. Hipótesis

En este trabajo se propone una metodología para la detección de frutas en imágenes, basada en algoritmos de redes neuronales modernas y métodos de aumento, con el propósito de demostrar que estas pueden superar significativamente en términos de precisión las metodologías convencionalmente usadas. Se proponen las siguientes hipótesis:

1. Se puede obtener mejoras significativas en tareas de detección de frutas al usar metodologías modernas usadas en detección de objetos. Dichas mejoras son similares observadas en competencias, tales como COCO, frente a acercamientos convencionales, por lo que redes más precisas en detección de objetos tenderán a ser más precisas en detección de fruta.

2. Dada las mismas condiciones de entrenamiento, las redes neuronales modernas exhiben una precisión y recall más altos, independiente del tipo de fruto estudiado.

3. Las metodologías de entrenamiento basadas en aumento de datos son capaces de aumentar la precisión de las redes entrenadas con estos métodos, con independencia del tipo de red o el tipo de fruta usada.

1.5. Objetivos

1.5.1. Objetivo General

El objetivo de este trabajo es el desarrollo de un sistema de DF robusto y altamente preciso, basado en el uso de métodos de aumento de datos para el entrenamiento de redes neuronales y evaluado en múltiples redes. Dicho método será definido a través de una comparación extensiva de distintas redes neuronales y métodos de aumento de datos, validados a través de distintas bases de datos.

1.5.2. Objetivos Específicos

Para lograr el Objetivo General de esta tesis se plantean los siguientes objetivos específicos:

- Realizar una revisión bibliográfica de métodos de detección de objetos en agricultura, con un énfasis en detección de frutas, para identificar desafíos y proponer soluciones.
- Identificar tendencias en los algoritmos de detección de frutas
- Hacer una revisión bibliográfica del estado del arte en detección de objetos, con un énfasis en Redes Neuronales Profundas.
- Implementar una librería para distintos algoritmos de aumento de datos.
- Desarrollo de una base de datos de cerezas para evaluar las redes.
- Realizar una comparación extensiva entre las redes evaluadas y métodos de aumento de datos, a través de distintas bases de datos de frutas.
- Identificar Fortalezas y debilidades de las distintas redes y metodologías evaluadas en tareas de detección de frutas.

1.6. Estructura de Tesis

El resto de la Tesis se organiza en los siguientes capítulos:
El Capítulo 2 consiste en el Marco Teórico de esta tesis, donde se describe a detalle las metodologías y recursos utilizados, tales como las Redes Neuronales, sus características más relevantes, como se evalúan y entrenan dichos algoritmos.

El Capítulo 3 describe el uso de algoritmos de detección de frutas, partiendo de las dificultades intrínsecas de esta tarea. Luego se procede a hacer una revisión de las distintas metodologías usadas para la detección de frutas, así como las principales aplicaciones para este tipo de algoritmos.

El Capítulo 4 hace un análisis de las tendencias en detección de objetos, haciendo un énfasis en los más recientes y relevantes avances en esta área.

El Capítulo 5 presenta la problemática principal señalando las principales diferencias observadas entre las áreas de detección de frutas y la detección de objetos. A partir de esto se señala la potencial utilidad en el uso de algoritmos modernos para la detección de frutas. Posteriormente se propone una metodología de evaluación de distintas redes neuronales y algoritmos de aumento de datos para demostrar la utilidad de estos, así como cuantificar su desempeño.

El Capítulo 6 presenta la base de datos Cherry CO, diseñado para la evaluación y entrenamiento de algoritmos de detección de frutas. Se describe los pasos realizados para su construcción, incluyendo la adquisición de datos, la anotación de las imágenes y se da un análisis estadístico de las anotaciones resultantes.

El Capítulo 7 describe la experimentación realizada. Se parte dando la descripción de los experimentos a realizar, detallando el uso de bases de datos, librerías, redes y demás recursos utilizados, así como la justificación de estas elecciones. Luego se presentan los resultados a través de los distintos experimentos. A partir de estos se comparan las distintas metodologías y se analiza la utilidad de estas en las dos bases de datos utilizadas, Cherry CO y MinneApple.

El Capítulo 8 detalla las conclusiones de los experimentos realizados, los cuales se contrastan con las hipótesis que se tienen sobre las redes y metodologías. Finalmente se presentan las implicaciones y recomendaciones de futuros desarrollos.

Capítulo 2

Marco Teórico

2.1. Aprendizaje Profundo

Las Redes Neuronales Artificiales o Artificial Neural Network (ANN) son algoritmos inspirados en el funcionamiento de las neuronas biológicas, utilizando como unidad base las neuronas artificiales. Similarmente a las neuronas biológicas estas neuronas artificiales pueden aprender a interpretar datos, emulando un comportamiento deseado, donde dado cierto estimulo se produce una respuesta concreta. Otra similitud que tienen con las neuronas biológicas es que suelen tener umbrales de activación en donde su respuesta a un estímulo es distinta si supera dicho umbral.

Cada neurona artificial es fundamentalmente una operación matemática, con una serie de parámetros, que se utiliza para realizar algún tipo de predicción. Las redes neuronales artificiales se componen de múltiples neuronas, dispuestas como una secuencia de operaciones lineales y/o paralelas. El aprendizaje profundo o Deep Learning (DL) es un área del Aprendizaje de Máquina que emplea Redes Neuronales Profundas, también conocidas como Deep Neural Network o DNN, las cuales están compuestas por millones de neuronas. Redes neuronales profundas pueden consistir en decenas o incluso cientos de millones de parámetros, lo que les da la capacidad de adaptarse a una gran variedad de problemas. Se ha observado que dado un número suficiente de datos de entrenamiento las DNN son la solución más precisa para un gran número de problemas, entre ellos el procesamiento de imágenes.

2.1.1. Módulos y Operaciones en Redes Neuronales

Se han desarrollado múltiples operaciones y módulos para mejorar el rendimiento de las Redes, ya sea en términos de precisión o de velocidad. Esta sección presenta los módulos más comunes en las ANN's.

2.1.1.1. Capas Fully Connected

La unidad básica de las redes neuronales es la neurona artificial o perceptron [22], la cual se define como una función matemática con parámetros. Dicha operación consiste en una función lineal definida por los pesos/parámetros $\vec{w}$ y b. Los pesos $\vec{w}$ se usan para calcular el producto punto con un vector de datos de entrada $\vec{v_i} = [x_1, x_2, x_3, ..., x_n]$. Esto da un resultado escalar, al cual se le aplica lo que se conoce como una función de activación, que

comúnmente es una función no lineal ϕ.

$$v_o = \phi(< \vec{w}, \vec{v}_i > + b) \tag{2.1}$$

Inicialmente se utiliza la función escalón, es decir, si el resultado de la ecuación lineal supera 0 el resultado es $v_o = 1$, en caso contrario es 0. Esto permite generar neuronas con resultados binarios, que eran prácticas para tareas de clasificación.

Múltiples neuronas agrupadas generan una capa neuronal. En este caso los pesos se agrupan como una matriz $W \epsilon R^{n \times m}$ y un vector de $\vec{b} = [b_1, b_2, \ldots, b_m]$. El resultado de una capa neuronal es un vector $\vec{v}_o = [y_1, y_2, y_3, ..., y_m]$. Una capa de neuronas se conoce como una operación de conexión total o operación Fully Connected (FC).

$$v_o = \phi(W\vec{v}_i + \vec{b}) = FC(v_i) \tag{2.2}$$

Las capas FC se disponen en múltiples operaciones secuenciales, efectivamente generando una cadena de operaciones, generando así lo que se conoce como Redes Neuronales Artificiales Fully Connected. Se ha observado que redes más profundas (más de 20 capas) suelen entregar mejores resultados que redes con menos capas. Es por esto que la mayoría de los algoritmos en el estado del arte en multitud de tareas, como procesamiento de imágenes, procesamiento de lenguaje o sistemas de control emplean redes neuronales profundas.

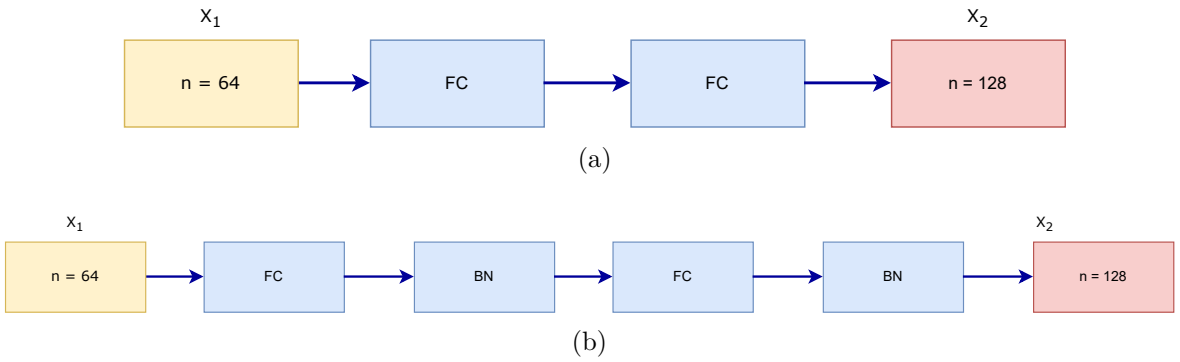


Figura 2.1: a) Sin batch normalization. b) Con Batch Normalization.

2.1.1.2. Normalización de Minibatches

El resultado de una operación en una red neuronal suele ser seguido por otra operación similar (ver Fig. 2.1 a)). Esta metodología puede llevar a que los valores en cada iteración aumenten en magnitud, lo que conlleva a valores más inestables en futuras operaciones. Para abordar este problema Ioffe et al. [23] propuso el método de Normalización por conjuntos o Batch Normalization (BN). El propósito de esto es normalizar el resultado de cada operación, permitiéndole a la red operar en valores en un rango acotado. Esta es una operación de normalización que se calcula usando un promedio $\bar{X}$ y una varianza σ_x:

$$X^* = \frac{X - \bar{x}}{\sigma_x} \tag{2.3}$$

El nombre de batch o conjunto, se refiere a que las redes neuronales se entrenan ingresando un subconjunto de datos, conocido como batch cuando se está entrenando a la red, en oposición

a usar la totalidad de datos de entrenamiento para ajustar los pesos. Los batches se ingresan de forma iterativa a la red para evaluar su actual resultado y modificar los pesos acordemente. Este conjunto suele ser de 8, 16 o 32 muestras, para que el costo computacional sea aceptable. Para obtener $\bar{X}$ y σ_x se calcula a partir de los batches. A partir de estas y las muestras anteriores se calcula la media móvil exponencial, así como la varianza correspondiente. La operación de BN se suele usar después de una operación, como una capa FC (Fig. 2.1 b)). Al ser una normalización las dimensiones antes y después de esta operación se mantienen constantes.

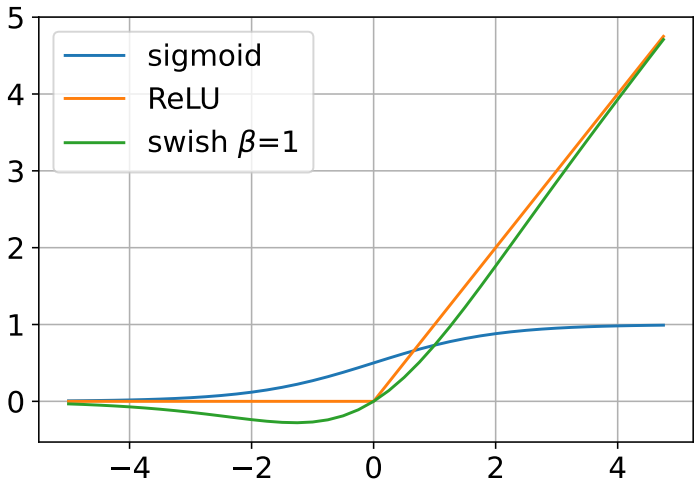


Figura 2.2: Funciones de activación.

2.1.1.3. Funciones de activación

Múltiples capas en ANN aplican operaciones lineales. Dado que las redes neuronales aplican capas de forma secuencial, redes que solo tengan operaciones lineales, podrán solo obtener resultados lineales. Esto es un problema dado que la mayoría de los problemas reales tienen un comportamiento no lineal. Es por esto que las capas FC, así como otras capas aplican operaciones no lineales también conocidas como funciones de activación.

Si bien es posible utilizar cualquier operación no lineal, las operaciones más comúnmente usadas son:

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

$$ReLU(x) = max\{x, 0\} \tag{2.5}$$

$$swish(x, \beta) = xsigmoid(x\beta) \tag{2.6}$$

Inicialmente la función de activación más común era la sigmoide, pero fue progresivamente desplazada por funciones como ReLU u otras similares, dado que se observó que las funciones de activación con un comportamiento similar a la identidad en ciertas regiones de su dominio entregan mejores resultados, tales como ReLU o swish (ver Fig. 2.2). Esta transición hacia este tipo de operaciones se debe a un fenómeno conocido como desvanecimiento del gradiente, el cual consiste en que las modificaciones realizadas a los pesos son ínfimas para las primeras

capas de la red debido a que el gradiente de ciertas capas tienen un valor muy bajo, lo que limita la capacidad de optimizar apropiadamente dichos parámetros. El gradiente de funciones con un comportamiento tipo ReLu converge a 1 en la zona lineal, en cambio el gradiente de la sigmoide converge a 0 cuando $|x| \to \infty$, debido a que la función es plana en estas zonas, lo que afecta al algoritmo de optimización usado en redes neuronales, que se explica en la sección 2.1.3.

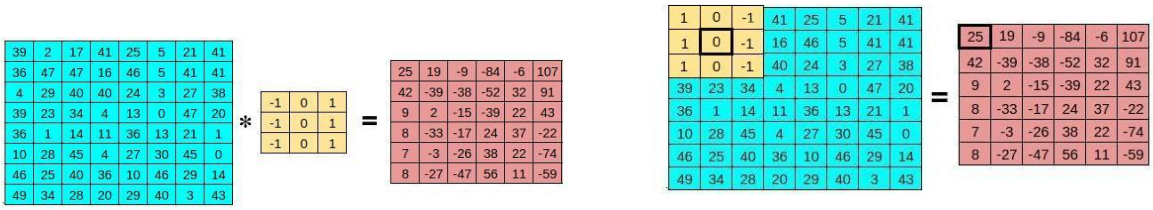
2.1.2. Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales son un tipo de red neuronal utilizada en el procesamiento de imágenes por su alta capacidad de interpretación de la información visual. Estas redes derivan su nombre de la operación de convolución en imágenes. Esta operación involucra dos matrices de datos, una matriz imagen $I \in R^{m \times n}$ y un kernel K $\in R^{k \times k}$. El resultado de la convolución es la matriz O que se define por 2.7:

$$O = I * K \tag{2.7}$$

$$O_{i,j} = \sum_{r=-k/2}^{k/2} \sum_{c=-k/2}^{k/2} I_{i-r,j-c} K_{k/2+r,k/2+c}$$
Con i=1, ... n; j = 1,....,m

Conceptualmente esta operación implica superponer una matriz traspuesta sobre la otra, multiplicando los elementos correspondientes, para finalmente sumar los resultados.



(a) (b)

Figura 2.3: a) Operación de Convolución. b) Ejemplo del cálculo de un elemento.

En caso de filtros de 3×3 el resultado de la imagen convolucionada es $O \in R^{(m-1)\times(n-1)}$, es decir, una imagen ligeramente más pequeña (ver Fig. 2.3 b)). Por ello es común extender en 1 el ancho y el largo de la imagen con $I^* \in R^{(m+1)\times(n+1)}$, lo que en consecuencia hace que la imagen resultante tenga las mismas dimensiones que la original. La extensión o padding implica incluir un valor de relleno en las ubicaciones agregadas que normalmente es un único valor, ya sea 0 o algún valor intermedio en la escala de 0-255.

La operación de convolución tiene una serie de propiedades importantes. En primer lugar, sus propiedades matemáticas incluyen Conmutatividad, Asociatividad, Distribución con respecto a la suma, entre otras, además de ser una operación lineal. Visualmente la convolución se ha usado para implementar filtros lineales, como filtros de bordes o filtros gaussianos, que permiten generar una nueva imagen que realza o disminuye una determinada característica de una imagen.

La aplicación de la operación de convolución en Redes Neuronales se hace en las llamadas

capas convolucionales. Una capa convolucional consiste en un módulo en una red donde participan 3 tensores, el tensor de entrada I de dimensiones $H \times W \times C_{in}$, el tensor de salida O de dimensiones $W_{out} \times H_{out} \times C_{out}$ y el tensor kernel K de dimensiones $k \times k \times C_{in} \times C_{out}$. La tercera dimensión definida por C_{in} y C_{out} para los tensores I y O se suele referir como los canales. Una capa convolucional realiza la siguiente operación:

$$\begin{aligned} O &= CL(I, K) \\ O_{\cdot,\cdot,i} &= \sum_{j=1}^{C_{in}} I_{\cdot,\cdot,j} * K_{\cdot,\cdot,j,i} \end{aligned} \tag{2.8}$$

Redes que emplean extensivamente capas convolucionales se conocen como Redes Neuronales Convolucionales o Convolutional Neural Network. Las CNN's se han convertido en un pilar fundamental en tareas de procesamiento de imágenes, donde la mayoría de las redes en el estado del arte emplean este tipo de redes. Uno de los usos mas significativos de redes convolucionales fue la implementación de la red Alexnet [24] en la competencia de ImageNet [25] una competencia de clasificación de imágenes altamente reconocida. Esto lo logró en gran parte por el uso de capas convolucionales y las propiedades de aprendizaje de las redes neuronales. Posteriores trabajos en esta competencia se enfocaron casi exclusivamente en redes CNN's.

2.1.2.1. Operación de Pooling

Las operaciones de Agrupamiento o *Pooling* es otro tipo de función ampliamente usada en redes neuronales. Esta operación genera una nueva matriz a partir de una matriz de entrada I. Para ello se aplica una función de agrupamiento sobre ventanas en distintas posiciones de la matriz I. Las ventanas pueden ser de dimensiones $n \times m$, con $n, m > 1$, pero lo normal es que sean de 3×3. En la Fig. 2.4 se observa un caso de Max pooling, es decir, que es un tipo de agrupamiento donde se extrae el máximo valor dentro de la ventana correspondiente. Por cada ubicación en la matriz de salida O, existe una coordenada de la ventana correspondiente. Para calcular cada coordenada de la matriz O, la ventana es desplazada en un espacio en la matriz I. Cabe mencionar que en la Fig. 2.4 la matriz O tiene dos filas y dos columnas menos que I. Esto se debe a que la ventana de 3×3 no se puede definir en los bordes de I. Para evitar perder el borde la información del borde de la imagen se suele usar una estrategia de padding, similar a la usada en las redes convolucionales.

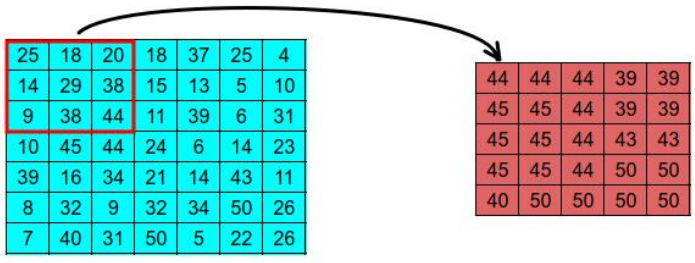


Figura 2.4: Max pooling con stride=1 en una ventana de 3×3.

El desplazamiento de la venta se conoce como stride y en el caso de la Fig. 2.4 el stride es de 1. Sin embargo es común emplear un stride = 2, para separar aún más las ventanas, lo que resulta en una imagen O que tiene la mitad de la resolución que la imagen original. Esto es importante en redes neuronales para poder reducir la resolución de forma iterativa,

pero manteniendo las características importantes. Esta reducción de resolución es uno de los principales usos que tiene la operación de pooling.

En las redes neuronales la operación de pooling se aplica canal a canal, es decir, se recibe un tensor de dimensiones $H \times W \times C$, y en cada uno de los C canales se aplica la operación de pooling con un stride = 2, lo que resulta en un tensor de dimensiones $\frac{H}{2} \times \frac{W}{2} \times C$.

2.1.2.2. Skip Connection

Es usual que ciertas CNN's tengan en uno o varios puntos una bifurcación entre sus operaciones secuenciales, donde se genera dos ramas, la rama de procesamiento y la rada de salto o Skip. La idea es poder aplicar múltiples operaciones sobre la capa de procesamiento, mientras que la rama de salto mantiene una copia del input original (ver Fig. 2.5). El resultado de ambas ramas es posteriormente combinado, a través de la suma de ambos tensores:

$$X_2 = X_1 + F(X_1) \tag{2.9}$$

Operaciones que siguen la ecuación 2.9, se conocen como Conexiones de Salto o *Skip Connections*. La rama de procesamiento puede consistir en operaciones de Convolución, Funciones de activación, Batch normalization u otras operaciones. Se ha observado que las Skip Connection permiten conservar información necesaria, pero permitiendo realizar modificaciones específicas, lo que le da más flexibilidad a la red durante el entrenamiento.

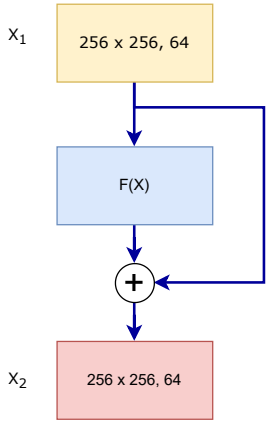


Figura 2.5: Diagrama de Skip connection.

Otras alternativas que se han evaluado han sido la fusión de ramas a través de la concatenación, en lugar de la suma. La concatenación es una operación que combina dos tensores de la misma resolución, agregando los canales de uno al otro. Si las dimensiones de X_1 es $H \times W \times C_1$ y las dimensiones de $F(X_1)$ son X_1 es $H \times W \times C_2$, entonces $X_3 \in R^{H \times W \times (C_1 + C_2)}$.

2.1.3. Entrenamiento de Redes Neuronales

Las ANN's son algoritmos de aprendizaje de maquina que son entrenadas de forma iterativa que reciben una entrada x para poder predecir una salida y. Para esto las redes se diseñan como una función $f(x) = y^*$, y el entrenamiento de esta implica usar múltiples ejemplos de la forma (x, y) para ajustar los parámetros de la red y así lograr que las predicciones se comporten como los ejemplos. El entrenamiento de las ANN consiste en realizar pequeñas correcciones en los pesos de la red. Para definir la corrección a los pesos se define una función

a minimizar, conocida como función de costo que es un resultado escalar del error de las predicciones:

$$\theta^* = \text{argmin}_{\theta \epsilon \Theta} C(\theta) = \text{argmin}_{\theta \epsilon \Theta} \frac{1}{N} \sum_{i=1}^{n} \mathcal{L}(y(i), f_\theta(x(i))) \tag{2.10}$$

En este caso se define $C()$ como la función a minimizar, tomando N ejemplos (entrada y salida), es decir, batch-size = N, y con parámetros/pesos θ. La parte más importante de 2.10 es la *Función de pérdida* $\mathcal{L}$ la cual tiene como rol definir la magnitud del error de cada predicción. Una de las funciones de pérdida más comúnmente usadas es la entropía cruzada.

Para modificar los pesos de la red se utiliza la técnica del *Descenso del Gradiente*. Este algoritmo modifica los parámetros de una función a través de determinar la dirección del gradiente de la función de costo, como función de los pesos. Esto se hace porque la dirección opuesta al gradiente de una función es también la dirección de máximo descenso de dicha función.

$$\theta_k = \theta_{k-1} - \mu \nabla_{\theta_{k-1}} C_{\theta_{k-1}}(x) \tag{2.11}$$

Dado que el gradiente sólo determina una dirección para modificar los pesos θ, basados en sus valores actuales, este algoritmo no asegura un mínimo global para la función de costo, sino que ofrece una constante reducción, lo que comúnmente lleva a mínimos locales.

Dado que el cálculo del gradiente sólo ofrece una dirección de máximo descenso, se debe escoger la magnitud del descenso. Por ello el gradiente es ponderado por un factor μ conocido como razón del gradiente o learning rate (LR). Este es un factor comúnmente pequeño ($\mu \sim 0.001$) para que la modificación de los pesos se mantenga en el rango donde la dirección del gradiente sea un indicador válido. De este modo se aplican pequeñas correcciones de forma iterativa a los pesos de la red. Un LR muy grande podría significar un proceso lento e inestable para llegar hasta el mínimo de la función (ver Fig. 2.6 a), por otro lado uno muy pequeño podría hacer el entrenamiento lento o prevenirlo de llegar al mínimo (Fig. 2.6 b). Es decir que el valor correcto de LR tiene un rol importante en el proceso de entrenamiento.

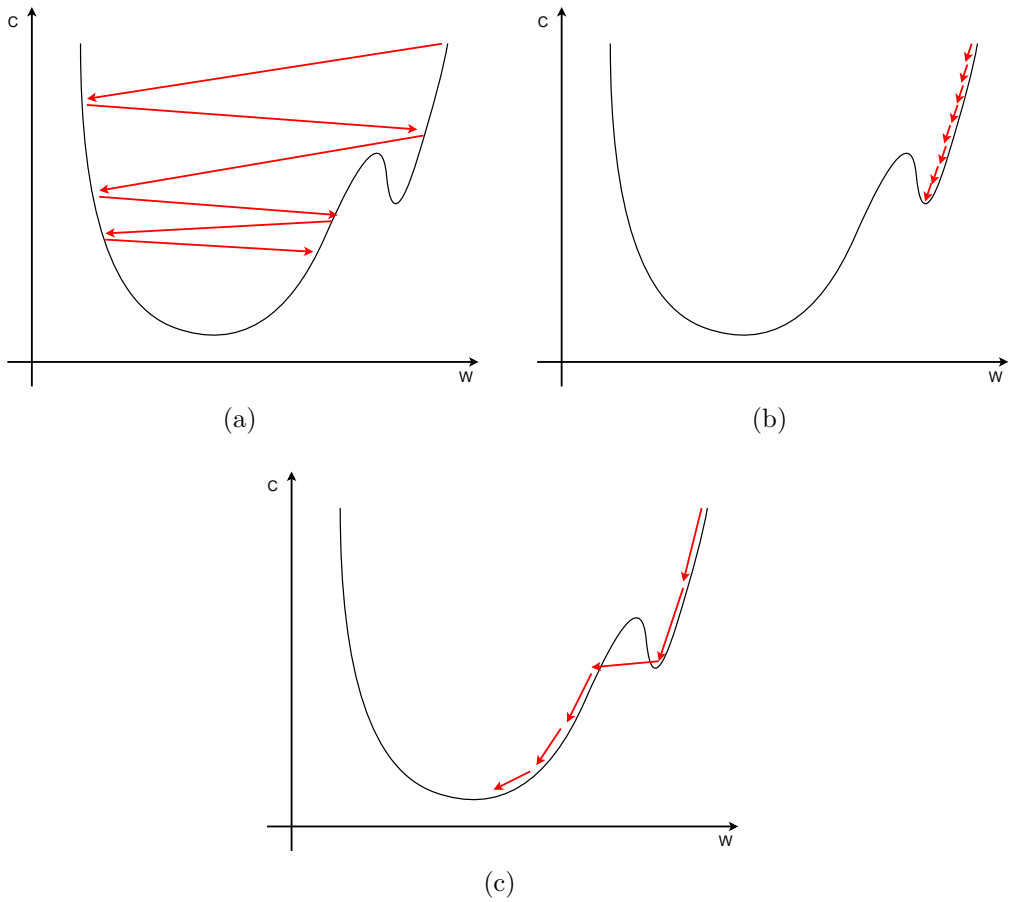


Figura 2.6: Ejemplos de descenso del gradiente con distintos learning rates.
a) LR alto. b) LR bajo. c) LR apropiado.

El cálculo del gradiente es una operación computacionalmente muy costosa, la cual a su vez depende del valor de x, de modo que cada vez que se hace una iteración se requiere re-calcular el gradiente. Para realizar las correcciones se utiliza el algoritmo de propagación hacia atrás, también conocido Back propagation (BP). Este algoritmo consiste en utilizar la regla de la cadena para determinar el gradiente de cada capa de forma iterativa. Dado que las redes neuronales se forman con operaciones secuenciales, es decir, como una composición de funciones de la forma $f_n(f_{n-1}(f_{n-1}(...f_1(x))))$ el gradiente de cada capa se puede expresar como una función del gradiente propio y el de la capa anterior:

$$\frac{\delta f_i(f_{i-1} \cdot f_{i-2} \cdots)}{\delta x}(x) = \frac{\delta f_i(f_{i-1})}{\delta f_{i-1}} \frac{\delta f_{i-1} \cdot f_{i-2} \cdots}{\delta x}(x) = \frac{\delta f_i(f_{i-1})}{\delta f_{i-1}} \frac{\delta f_{i-1}(f_{i-2})}{\delta f_{i-2}} \cdots \frac{\delta f_1}{\delta x}(x) \tag{2.12}$$

La ecuación 2.12 muestra porque funciones con gradiente cercanos a 0 perjudican al aprendizaje de las redes, dado que cualquier función con un gradiente muy bajo hará decaer significativamente la magnitud del gradiente en capas anteriores, lo que previene la modificación de los pesos en las capas inferiores, causando el desvanecimiento del gradiente.

La expresión 2.12 resulta de gran utilidad, dado que se puede calcular el gradiente capa por capa, partiendo desde la función de error y desde ahí retrocediendo a través de la red:

$$\nabla C_\theta(x) = \frac{\delta C_\theta(x)}{\delta x} = \frac{1}{N}\sum_{i=1}^{N} \frac{\delta \mathcal{L}(y(i), f_\theta(x(i)))}{\delta x} \tag{2.13}$$

Este acercamiento se suele referir como backpropagation por mini-batches, al usar solo un conjunto limitado de datos.

Se han desarrollado métodos basados en BP que mejoran el comportamiento del descenso del gradiente. Uno de los más importantes es el desarrollo del algoritmo Adam [26], el cual emplea back propagation, modificado la dirección de descenso con lo que se conoce como momentum adaptativo (Adaptive momentum). El efecto que esto tiene en el entrenamiento es que la actualización de los pesos no se ve sobre-sesgada por cada iteración, sino que las actualizaciones de BP siguen una dirección general de descenso.

2.2. Redes neuronales para detección

2.2.1. Estructura de una red neuronal para detección

Las DNN's para detección tienen dos estructuras principales, conocidas como backbone y cabeza de predicción, con roles específicos. El backbone es la primera etapa en el procesamiento de una imagen, la cual sirve para generar representaciones de la imagen que describan los objetos presentes en ellas, que clases de objetos son y en donde se ubicación. Un backbone recibe una imagen de dimensiones $H \times W \times 3$ y genera un tensor de salida de dimensiones $H' \times W' \times C$, donde a menudo la resolución $H' \times W'$ es una fracción de $H \times W$ (ver Fig. 2.7). Este tensor de salida se conoce como mapa de características el cual es usado por la cabeza de predicción que entrega una lista de predicciones, las cuales consisten en las coordenadas y la clase de los objetos detectados por la red.

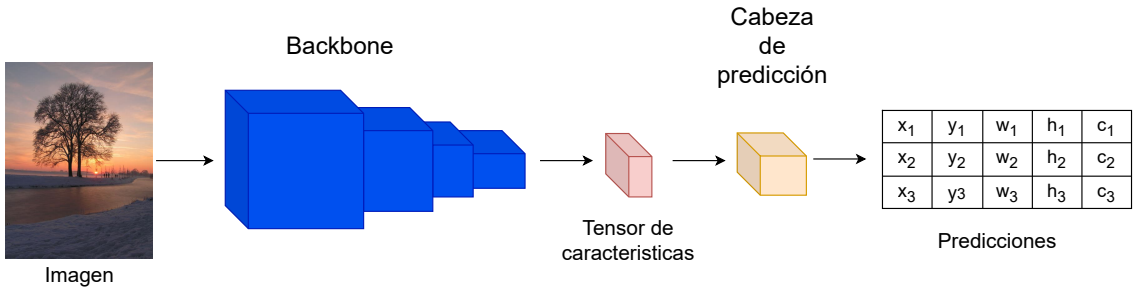


Figura 2.7: Esquema de red de detección de objetos.

2.2.2. Tipos de redes de detección

Existen dos categorías convencionales para la estimación de bounding boxes por parte de una red, y dependiendo de cuál método se utilice se clasifica a la red como Two-stage (de dos etapas) o One-stage (una etapa). Las redes Two-Stage se dividen en una primera etapa de generación de propuestas/candidatos a objetos. Un candidato se define como una zona rectangular equivalente a un BB donde posiblemente hay un objeto, conocida como 'anchor'. Estas zonas se seleccionan en posiciones aleatorias a lo largo de la imagen en diferentes tamaños y razones de aspecto. En general los anchors no van a encerrar un objeto, por lo que una red debe calcular la probabilidad de que que un anchor encierre un objeto. Los candidatos con la probabilidad más alta van a pasar a la siguiente etapa de detección, donde cada objeto será clasificado como alguna de las clases que se buscan identificar, o como parte del fondo de la imagen (background). Ejemplos de este tipo de redes son las redes Faster-RCNN [27], Mask-RCNN [28] y Hybrid task Cascade [29]. Por otro lado, las redes One-Stage generan una grilla de anchors que abarca la imágen completa. Cada coordenada en esta grilla tiene asociada diferentes candidatos con distintas escalas y relaciones de aspecto para poder modelar objetos de distinto tamaño y escala. La cabeza de predicción de la red define la probabilidad de pertenencia de cada anchor a una clase específica, con lo que anchors que se ajusten a un objeto tendrán un alto grado de pertenencia a una clase, mientras que grillas definidas en background tendrán una probabilidad cercana a nula. Ejemplos de este tipo de redes son SSD [30] o redes YOLO [31–35]. En ambos tipos de arquitecturas se calculan factores de off-set para darle más flexibilidad a las redes a la hora de ubicar un objeto, dado que un objeto puede que no se ajuste perfectamente a un anchor pre-definido.

2.3. Métricas de detección

Para cualquier aplicación de un algoritmo se requiere alguna métrica para poder cuantificar la capacidad de un determinado método de resolver el problema y poder compararlo con otros. En esta subsección se explican las métricas utilizadas para tareas de detección de objetos.

2.3.1. Matriz de confusión y sus componentes

Para distintas tareas como clasificación, búsqueda de elementos y detección se realizan predicciones sobre un conjunto de datos. Estas predicciones corresponden a la pertenencia de un elemento de este conjunto a alguna clase. En un caso binario donde cada elemento puede pertenecer a una clase positiva (P) y a una negativa (N), cada elemento se clasificará como una clase u otra, siendo esta clasificación la predicción de un algoritmo. En donde existen 4 escenarios posibles para cada elemento:

- Verdadero positivo (VP): Un elemento positivo fue clasificado como positivo
- Verdadero Negativo (VN): Un elemento negativo fue clasificado como negativo
- Falso positivo (FP): Un elemento negativo fue clasificado como positivo
- Falso Negativo (FN): Un elemento positivo fue calificado como negativo

Una matriz de confusión se conoce como un arreglo de 2×2 que muestra la distribución del conjunto de datos en estos (ver Fig. 2.8):

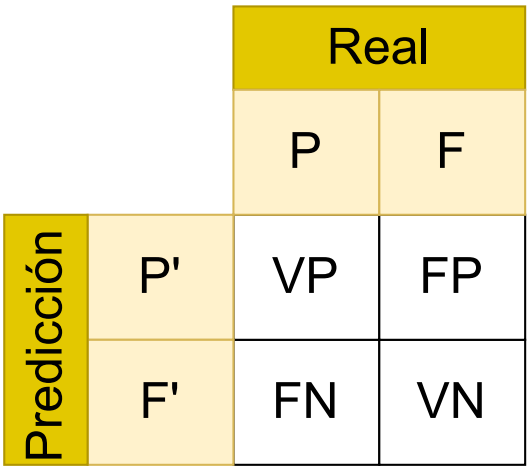


Figura 2.8: Estructura de una matriz de confusión binaria.

Cuando se evalúa un algoritmo se analizan principalmente dos valores relacionados, que son la Precisión y el Recuento (también llamado Recall):

$$Precision = \frac{VP}{VP + FP} \tag{2.14}$$

$$Recuento = \frac{VP}{VP + FN} \tag{2.15}$$

La precisión mide la razón entre los casos verdaderos clasificados correctamente en relación con la cantidad total de calificaciones positivas. El recuento, por otro lado, calcula la razón entre los VP y todos los positivos, por lo que sirve para tener una medida de completitud en cuanto a la clasificación de positivos. Ambas métricas están acotadas en el rango de $[0, 1]$ y es deseable que tengan el valor 1. Un modo trivial de obtener una precisión de 1 es clasificando solamente un elemento que sea positivo como tal, en cuyo caso FP = 0 y VP = 1. Sin embargo, esto puede ser sumamente perjudicial para el recuento, ya que, si había N = 100 elementos positivos y solo se clasificó uno como positivo FN = 99 y el recuento es 0.01. De igual modo, existe un modo trivial para obtener un Recuento = 1, que consiste en clasificar cada elemento como uno positivo, en cuyo caso FN = 0 y VP = 100. Asumiendo que también haya 100 ejemplos negativos esto causaría que FP = 100, lo que lleva a Precisión = 0.5. Este ejemplo muestra que estas dos métricas sirven como un contrapeso una de la otra y que aumentar ambas simultáneamente solo se puede lograr reduciendo FP o FN, es decir, reduciendo las clasificaciones incorrectas.

2.3.2. Intersección sobre la unión

Para definir detecciones positivas y negativas en tareas de detección de objetos se debe definir algún mecanismo de clasificación de las predicciones. Cada predicción se compone de una clasificación y coordenadas. Para poder definir el acierto de estas coordenadas se utiliza la Intersección sobre la unión o Intersección over Union (IoU), que se define como la razón entre la intersección de las área de la predicción (A_p) y el área real (A_r), con la unión de dichas áreas:

$$IoU = \frac{A_p \cap A_r}{A_p \cup A_r} \tag{2.16}$$

Dado que la intersección es siempre menor igual a la unión, esta métrica está definida en el rango [0,1]. Una buena predicción tendrá un valor cercano a 1, mientras que dos predicciones que no se interceptan obtendrán un IoU=0. De esta forma se define una medida de acierto para cada predicción.

Para definir una predicción como una predicción positiva o negativa se define primero un umbral, típicamente $\mu = 0.5$. Se considerará una predicción correcta solo si $IoU \geq \mu$.

2.3.3. Precisión promedio y Recuento promedio

En tareas de detección la métrica principal para comparar las redes es la precisión promedio o Average Precisión (AP). Esta métrica evalúa las distintas predicciones en comparación con las etiquetas correspondientes. Cada predicción está definida por sus coordenadas, la clase predicha y el grado de confianza de esta. Este último valor representa la seguridad con que la red percibe la presencia de un objeto. El cálculo de AP consiste en las siguientes etapas:

1. Asociar cada predicción a un objeto en la base de datos basado en su ubicación.
2. Calcular el IoU entre predicción y su objeto asociado y clasificar cada predicción p_i como VP o FP, representados por 1 y 0 respectivamente.

$$CL(p_i) = \begin{cases} 1 & IoU(p_i) \geq \mu \\ 0 & else \end{cases} \tag{2.17}$$

De este modo se obtiene un listado de clasificaciones para cada predicción C = [CL(p_0), $CL(p_1)$, ..., $CL(p_N)$].

3. Se ordenan las clasificaciones C según su confianza, C^{sort} (ver Tabla 2.1 columna Confianza).

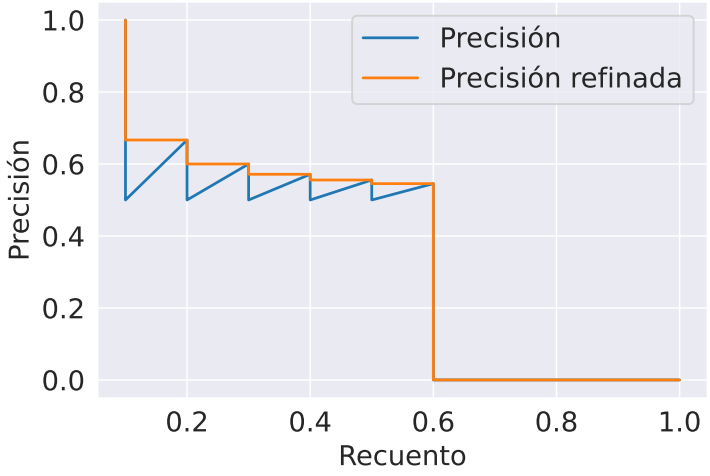
4. Se calcula el valor de precisión para la predicción i de mayor confianza como:

$$pr_i = \frac{1}{i}\sum_{j=1}^{i} CL(p_j) \tag{2.18}$$

5. Se calcula el valor del recuento para la predicción i:

$$rec_i = \frac{1}{N}\sum_{j=1}^{i} CL(p_j) \tag{2.19}$$

6. Se generan las curvas de precisión vs recuento. Posteriormente se corrige la curva de precisión para que sea decreciente y así evitar oscilaciones (ver Fig. 2.9).



(a)

Figura 2.9: Curvas ROC's.

7. Finalmente se toman los valores de la curva de precisión en 10 lugares, correspondientes a las coordenadas de recuento = $\{0.0, 0.1, 0.2, \ldots, 0.9, 1.0\}$. Estos valores se usan para calcular un promedio, conocido como AP.

Este procedimiento se realiza individualmente por cada clase predicha y luego se promedian dichos resultados. Existen distintas variantes de esta métrica. Por ejemplo, cuando se utiliza el umbral $\mu = 0.5$ se suele referir como AP@0.5, mientras que si μ =0.75, se le llama AP@0.75. Otra métrica derivada se conoce como Mean Average Precisión (mAP) y se calcula como el promedio de distintas AP con distintos μ's. La más conocida es la llamada mAP@0.50:0.05:0.95 que calcula el promedio de AP's para $\mu = \{0.50, 0.55, 0.60, \ldots, 0.95\}$.

Tabla 2.1: Ejemplo de calculo de Recuento y Precisión

Número de Detección	Confianza	VP	FP	Total VP	Precisión	Precisión refinada	Recuento
1	0.97	1	0	1	1.00	1.00	0.10
2	0.92	0	1	1	0.50	0.67	0.10
3	0.88	1	0	2	0.67	0.67	0.20
4	0.88	0	1	2	0.50	0.60	0.20
5	0.86	1	0	3	0.60	0.60	0.30
6	0.82	0	1	3	0.50	0.57	0.30
7	0.77	1	0	4	0.57	0.57	0.40
8	0.70	0	1	4	0.50	0.56	0.40
9	0.63	1	0	5	0.56	0.56	0.50
10	0.44	0	1	5	0.50	0.55	0.50
11	0.41	1	0	6	0.55	0.55	0.60
12	0.32	0	1	6	0.50	0.50	0.60
				Caso sin predicciones	0	0	1

2.3.3.1. F1-score

Como se observa en la imagen 2.9 las métricas recuento y precisión obtienen su máximo en el mínimo de la otra. Esto se debe a que ambas métricas sirven de contrapeso una de la otra, es decir, uno puede mejorar una métrica en perjuicio de la otra. Sin embargo, un buen detector tiene un alto recuento y una alta precisión, por ello se suele usar una métrica conocida como puntaje F1 (F1-score), el cual es la media armónica entre la precisión y el recuento:

$$F1 = \frac{2}{Recuento^{-1} + Precision^{-1}} \tag{2.20}$$

Esta expresión permite comparar distintos detectores sopesando ambas métricas, además de penalizar detectores desbalanceados, que tengan una métrica mucho más baja que la otra.

Capítulo 3

Detección y segmentación de frutas basado en aprendizaje profundo

Recientemente distintos sistemas inteligentes se han implementado en aplicaciones agrícolas con el fin de abordar los distintos desafíos presentes en métodos tradicionales de cosechado. Esto ha llevado a un número importante de investigaciones para estudiar la capacidad y viabilidad de distintos sistemas de visión computacional en la agricultura [3, 12, 36], y en particular en tareas asociadas con frutas dado el interés económico y nutricional que estas ofrecen. La detección de frutas ofrece la capacidad de poder observar, monitorear e interactuar con estas [10–12, 37, 38], lo que ofrece innumerables aplicaciones. Naturalmente se han generado una variedad de metodologías con altos niveles de precisión para la detección de distintos frutos. Un acercamiento inicial es utilizar características visuales como indicador para ubicar las frutas. Por ejemplo Puttemans et al. [39] utilizó características de Haar [40] entrenadas vías AdaBoost para detectar manzanas y frutillas. Este tipo de acercamiento, aunque capaz de detectar un gran número de frutas, suelen ser superados por algoritmos basados en aprendizaje profundo, los cuales se han convertido en la norma en detección de frutas (ver Fig. 3.1).

La mayoría de los estudios evalúa distintos acercamientos para mejorar el rendimiento de las redes, enfocados en aumentar la velocidad y precisión. En esta sección se hace una revisión sistemática de los desafíos presentes en la Detección de Frutas, el uso de redes, componentes y metodologías asociadas, y finalmente se presentan las principales aplicaciones basadas en DF.

3.1. Desafíos de la detección de frutas

Hay un gran número de problemas asociados con la detección de frutas, que hacen esta tarea desafiante en distintos aspectos. Los más relevantes son:

- Alta oclusión: En general una imagen en un ambiente agrícola tendrá algún objeto bloqueando parcial o totalmente la visión de la cámara. Las frutas pueden ser oclusionadas por hojas, ramas u otras frutas (ver Fig. 3.2 a).

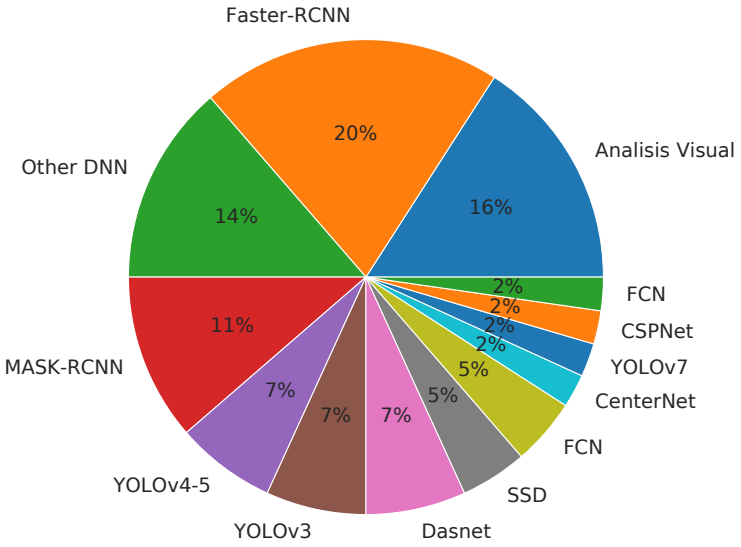


Figura 3.1: Distribución de redes usadas en estudios de detección de frutas revisados (ver Tabla 3.1).

(a) (b) (c)

Figura 3.2: Ejemplos de bases de datos de frutas. a) Apple Dataset Benchmark from Orchard Environment in Modern Fruiting Wall [41]. b) Cherry CO. c) MinneApple [42].

- Condiciones de iluminación desafiantes: La mayoría de las plantaciones de fruta están a la intemperie y por ende la iluminación de imágenes tomadas puede variar significativamente dada la hora del día, el clima y posición del árbol. Si bien es posible usar iluminación artificial durante la noche o en invernaderos, por razones prácticas esta metodología no suele ser usada en los estudios. Incluso dentro de una misma planta puede variar la iluminación, dado que sólo algunas hojas y frutas pueden estar directamente iluminadas.

- Alta densidad de instancias: Muchas especies de árboles producen una gran cantidad de frutas en cada cosecha. Además hay frutas que se dan a través de racimos como lo son las cerezas (ver Fig. 3.2 b) o las uvas, condición que conlleva a una gran acumulación de fruta en una misma ubicación. Esta alta concentración de fruta dificulta distinguir frutas individualmente.

- Instancias de tamaño pequeño: Frutas, semillas o flores suelen ser objetos pequeños, que por ende suelen verse pequeños en las imágenes. Si bien es posible disponer de cámaras a corta distancia de estos, como ocurre en la detección de hierbas y tallos [36], esto no es la norma para la mayoría de aplicaciones, por limitaciones prácticas de donde se puede disponer las cámaras, como estar a cierta distancia de la planta. De hecho en el caso de las frutas, bases de datos publicas como MinneApple [42] o ACFR [43] cada fruta ocupa en promedio un pequeño porcentaje del área total de la imagen, menor al 2[%] (ver Fig. 3.2 c).
- Falta de datos: Hay una escasez de bases de datos publicas para detección de frutas, las cuales tienen poca variedad de frutas, principalmente manzanas [41–43]. Esto obliga a que la mayoría de autores genere sus propias bases de datos, las cuales suelen tener un tamaño pequeño, que limita la precisión de cualquier algoritmo de aprendizaje de máquina.
- Ambiente de trabajo complejo: La mayoría de las instalaciones para la producción agrícola tienen limitaciones en las zonas con acceso a electricidad, ubicaciones alejadas de centros de investigación e irregularidades en la zona de la plantación misma que genera complicaciones para tomar datos, disponer equipos o hacer pruebas. Adicionalmente experimentos y tomas de datos están limitadas a épocas del año, lo que dificulta la investigación en esta área.

3.2. Estado del arte en Detección de Frutas

Las redes neuronales se han convertido en la principal familia de algoritmos para procesamiento de imágenes en agricultura, reemplazando en la mayoría de los casos algoritmos basados en análisis de características (ver Fig 3.1). Estudios que apliquen CNN's han servido para demostrar el alcance de estas tecnologías, indicando que distintas redes pueden alcanzar altos niveles de precisión [17, 34, 44, 45] y velocidad [44, 46, 47]. Uno de los primeros ejemplos de esto fue el estudio de Sa et al. [43] que evaluó la precisión de la red Faster-RCNN para la detección de manzanas, almendras y mangos, obteniendo un F1-score de 0.9. Trabajos posteriores exploraron distintas redes y metodologías para mejorar el rendimiento de algoritmos de detección, los cuales exploraron distintos aspectos relacionados con el uso de estructuras de redes y metodologías de entrenamiento.

En esta tesis se realizó una revisión sistemática de los algoritmos y tareas asociadas con detección de frutas en imágenes. Se evaluaron estudios relacionados con la tarea de detección de frutas en ambientes agrícolas, es decir, frutas en arbustos o árboles, a menudo en periodos de tiempo donde la fruta ya está madura o cercana a esta etapa. Como el enfoque es en frutas, no se consideran trabajos que realicen tareas de procesamiento de imágenes de semillas, tallos, hojas, verduras o flores. También se excluyeron etapas posteriores en la producción de frutas, como el empaquetado de frutas en ambientes industriales, ya que, los desafíos y tareas asociadas difieren significativamente de los presentes en cultivos. La Tabla 3.1 presenta un resumen de las características de los estudios revisados.

Esta sección hace una revisión de las principales áreas de desarrollo observadas en detección de frutas desde la perspectiva de la visión computacional. En total se identificaron 6 áreas de desarrollo que son:

- Representaciones Multi-escala: Arquitecturas que generan representaciones invariantes al tamaño de objetos se les llama multi escala, y permiten aumentar la capacidad de inferencia de la red. Una de las operaciones más comúnmente usadas es un módulo FPN presentado por Lin et al. [48] que combina distintos niveles de resolución. Como se observa en la Fig. 3.3 en la estructura Bottom-Up una imagen de resolución $H \times W$ se usa para generar mapas de características de menor resolución, comúnmente dividiendo la resolución a la mitad. Este proceso se repite varias veces para obtener distintos niveles de resolución. Los niveles de menor resolución tienen mapas de características más procesadas, con más canales y un mayor campo receptivo, por lo que pueden identificar características complejas de la imagen y objetos de gran tamaño. Sin embargo, la información necesaria para identificar objetos pequeños se suele perder al disminuir la resolución. Para generar representaciones invariantes al tamaño se utiliza un mecanismo de fusión de información de distintos niveles de resolución (Top Down pathway), que expanden los tensores de menor resolución a través de una función de extrapolación, para luego sumarlos al nivel correspondiente, o en su defecto concatenarlos. Esta red ha sido la base para arquitecturas multi-escala y la mayoría de las redes tienen un diseño similar a FPN, generalmente denominado como cuello de la red. Uno de los primeros ejemplos de su uso en la detección de frutas fue en [49] con un Resnet101 modificado, usando un cuello FPN [50] y operaciones ASPP [51] para manejar las diferentes escalas de manzanas. Actualmente la mayoría de las redes en detección de fruta incluyen operaciones multi-escala [13, 44, 47, 52–56].

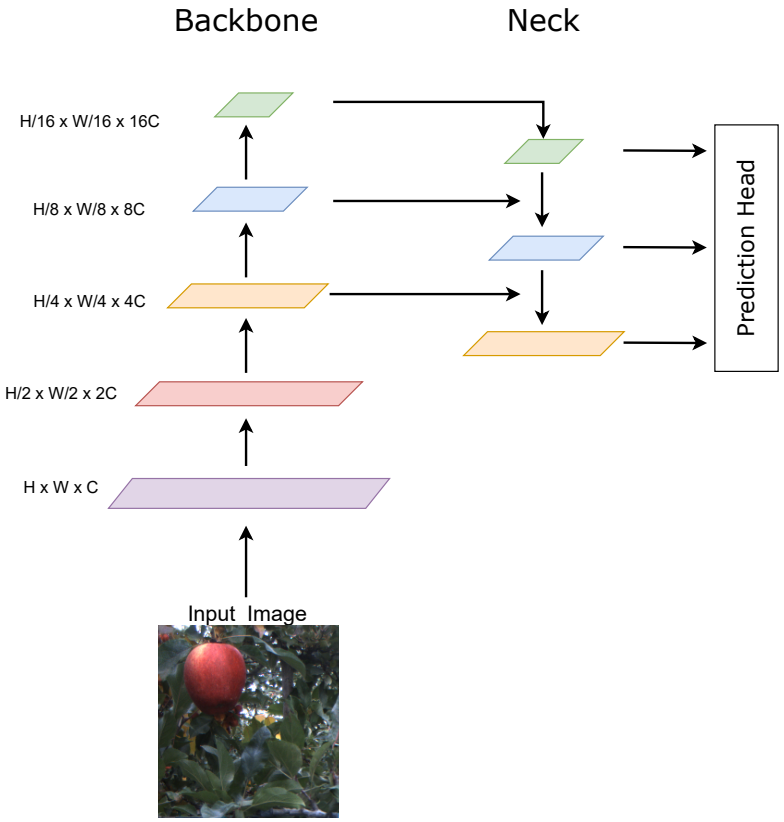
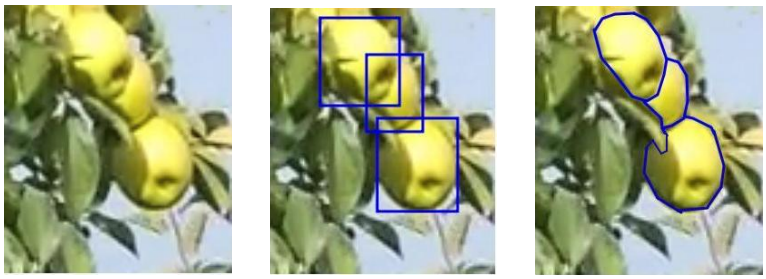


Figura 3.3: Arquitectura tipo FPN [48] para la generación de características Multi-escala.

- Módulos Ligeros: La aceleración de las redes ha sido un tema ampliamente investigado dado el gran número de aplicaciones que requieren respuestas en tiempo real. Para lograr esto se han estudiado múltiples operaciones que reducen el coste computacional, sin perjudicar el rendimiento de la red. Este tipo de operaciones permite aumentar la velocidad o aumentar la precisión de la red, manteniendo los requerimientos de hardware. En el contexto de detección de frutas hay aplicaciones que requieren de respuestas en tiempo real, como la recolección. Zheng et al. [17] presentó un análisis de su conjunto de datos CropDeep evaluando varias redes ligeras como SSD, YOLOv2 y YOLOv3, entre otras. Se concluyó que YOLOv3 fue más preciso en la mayoría de los cultivos (31 cultivos) que las otras dos redes mientras mantenía una velocidad comparable o mejor. Un módulo importante en redes modernas Ligeras es la conexión de salto parcial cruzada (CSP por sus siglas en ingles) [57]. La mayoría de las redes ligeras modernas usan este tipo de operación, tales como YOLOv4, YOLOR, CSPNet y YOLOv5. Fan et al. [55] empleo la red YOLOv5 en la detección de fresas. Se compararon diferentes versiones de YOLOv5 con otras redes livianas tales como SSD y EfficientDet, y se concluyó que las redes YOLOv5 tienen una mejor relación entre velocidad y precisión. Se obtuvieron resultados similares en [53] usando Light-CSPNet para detectar manzanas, naranjas y tomates. Se han empleado otras redes eficientes, como EfficientNet [58] utilizada por Wu et al. [47] como backbone de una red YOLOv4 para detectar manzanas. En comparación con otras redes como Faster-RCNN, YOLOv3 y YOLOv4, esta YOLOv4 modificada logró mejores tiempos y precisión en las tres bases de datos.

- Segmentación: La incorporación de redes de Segmentación ha presentada múltiples ventajas para la tarea de detección de frutas. En primer lugar se ha observado que entrenar una red para más de una tarea, en este caso detección y segmentación de objetos (DSO), mejora el rendimiento de esta. Una de las redes más usadas para tareas de DSO es Mask-RCNN [28] que consiste en una extensión de la red Faster-RCNN para realizar segmentación. Callum [15] comparó la precisión de Faster-RCNN y Mask-RCNN con un backbone Resnext101 y esta última presentó mayor precisión en la base de datos de MinneApple [42]. Una segunda ventaja en el uso de etiquetas de segmentación es que permiten abordar el problema de alta oclusión entre frutas mejor que etiquetas rectangulares, dado que las coordenadas de los BB's de frutas aledañas a menudo se sobreponen, mientras que las segmentaciones no (ver Fig. 3.4). Finalmente, la segmentación permite aplicaciones como estimación de pose de fruta cuando se usa en combinación con imágenes RGB-D, dado que permite estimar la profundidad asociada a cada píxel de una imagen. Al obtener la profundidad por píxel de una fruta vía segmentación se puede obtener la profundidad y posición estimada de la fruta [38, 44].



(a) (b) (c)

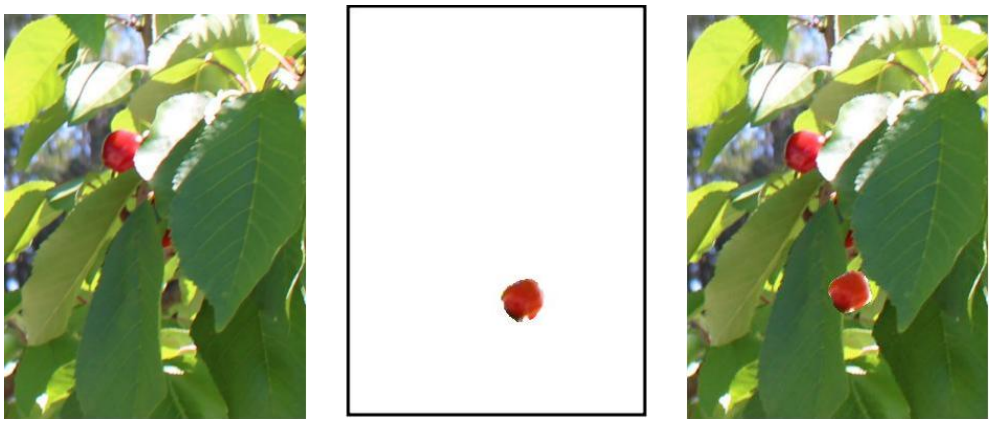
Figura 3.4: Oclusión en fruta. a) Imagen Original b) Etiquetas rectangulares c) Etiquetas de segmentos.

- Extra-Sensores: La inclusión de sensores visuales adicionales presenta múltiples ventajas para abordar problemas de detección de frutas. Por ejemplo cámaras que operan en el espectro Near Infrared (NIR) o cámaras Hiperespectrales han sido utilizadas en aplicaciones agrícolas para observar características que no son visibles por cámaras convencionales [4, 59]. Una de las primeras aplicaciones en detección de frutas fue realizada por Sa et [60] usando una red Faster-RCNN para la detección de pimentones en un invernadero, incorporando una imagen NIR en conjunto con una imagen RGB para obtener información adicional de la fruta. Se concluyó que la fusión entre información RGB y NIR entregaba resultados más precisos que usar solo uno de los dos tipos de imagen. Otras tecnologías ampliamente usadas son las cámara RGB-D para adquirir información de profundidad. La primera utilidad de este tipo de datos es que mejora el rendimiento de las redes cuando se usan como un cuarto canal, como lo muestra Gené-Mola et al. [61]. Otros autores no han usado el canal D como uno adicional en la tarea de detección, sino que lo utilizan para ubicar frutas en el espacio. Kang et al. [38] estudio el uso de imágenes RGB-D para la aplicación de un sistema de recolección de manzanas, a través de tres etapas: detección, estimación de pose y agarre. En este trabajo se emplearon las imágenes RGB para realizar detección y segmentación de fruta, para luego emplear la red PointNet [62] y estimar la posición dada la ubicación en píxeles de los segmentos de frutas. Otros estudios emplearon metodologías similares, aplicando una red para detección y segmentación y algún otro método para estimar la pose de frutas [54, 63].

- Modificación específicas basada en tareas: Un acercamiento común para abordar problemas de detección es emplear una red base y modificarla ligeramente en función del problema. Una característica de interés en DF es la implementación de detectores que pueden funcionar en tiempo real para que los robots puedan interactuar con las plantas. Por ello múltiples autores han desarrollado backbones o módulos para maximizar la velocidad. Este fue el caso de Zhang et al. [53] que empleó una versión más rápida de CSPNet para la detección de frutas. En total se evaluaron 3 conjuntos de datos, tomates, naranjas y manzanas, donde se observó que la red Ligth-CSPNet era más rápida que YOLOv3 y YOLOv4, a pesar de obtener un AP más alto que las otras. Otras áreas sujetas a modificaciones son los módulos de predicciones y funciones de pérdidas. Aspectos como tamaño y relación de aspecto han mejorado de manera consistente el rendimiento, al adaptar los anclajes a los objetos superpuestos y pequeños que se suelen encontrar en estos conjuntos de datos [34, 64, 65]. Por ejemplo Behera et al. [65] utilizó una versión modificada de IoU en Faster-RCNN que responde mejor ante objetos

altamente superpuestos.

- Aumento de datos: Este es un tipo de metodología que generan nuevos datos etiquetados para poder ser usados en el entrenamiento de las redes. Es común usar distintos algoritmos de *Aumento de datos* para poder enseñarle a la red distintos tipos de escenarios en los que se puede presentar un mismo tipo de objeto. Acercamientos convencionales son las transformaciones geométricas, como flipping, rotación, escalamiento y traslación de imágenes en el conjunto de entrenamiento. Otro tipo de transformaciones muy usadas son aquellas de carácter visual donde se modifica el contraste, la iluminación o se realizan transformaciones HSV, entre otras. Estos acercamientos le permiten a las redes aprender sobre las diferentes condiciones en las que se pueden encontrar objetos, tomando en cuenta variaciones en el tamaño, posición, inclinación, color y brillo. Este tipo de métodos convencionales ha sido la norma en tareas de detección. Siguiendo esta lógica también se han desarrollado métodos para crear nuevas imágenes realistas. Tian et al. [13] entrenó una red YOLOv3 para la detección de lesiones en manzanas. Para mejorar el rendimiento de la red se generaron imágenes adicionales de manzanas con lesiones usando una red CycleGAN [66]. La base de datos aumentada mostró un F1 score de 0.816, mientras que sin el aumento de CycleGan solo se obtuvo 0.764.



(a) (b) (c)

Figura 3.5: Oclusión en fruta. a) Imagen Original b) Etiquetas rectangulares c) Etiquetas de segmentos.

Otra metodología para crear nuevas imágenes es usando máscaras de imágenes etiquetadas para incorporar etiquetas adicionales en imágenes. Esto permite crear nuevas imágenes etiquetadas a partir de una imagen base y una imagen donante (ver Fig. 3.5). Pawara et al. [20] empleo un subset de máscaras (20 máscaras de manzanas), y se observó una mejora en su método de conteo de frutas. Wu et al. [47] entrenó una red YOLOv4 con la base de datos aumentada incorporando máscaras de hojas en imágenes de manzanas en posiciones aleatorias. Esto permitió generar imágenes con más oclusión. Se observó que F1 score mejoró un 2,08 [%] usando esta metodología.

3.3. Aplicaciones de detección de frutas

El cultivo de fruta consiste en múltiples procesos, tales como plantación, irrigación, fertilización, monitoreo y cosecha. Muchas de estas tareas pueden ser abordadas o facilitadas a través de sistemas de detección de fruta. Como resultado se han estudiado múltiples sistemas autónomos para abordar estas tareas. Las principales aplicaciones de DF son monitoreo de madurez y salud, estimación de producción y recolección.

3.3.1. Monitoreo de madurez y salud

Para el desarrollo de cualquier actividad agrícola es necesario evaluar el estado de los cultivos de forma continua. Tareas cómo estimar el grado de madurez de cultivos permite anticiparse al proceso de cosecha y asi tener las preparaciones logísticas correspondientes. Por otro lado, detectar condiciones inusuales como enfermedades, presencia de plagas o estrés hídrico permite tomar acciones y minimizar el daño. Normalmente este tipo de tareas requiere de mucho personal evaluando continuamente el estado de la plantación, y dada la extensión de la mayoría de estas se vuelve una tarea costosa, lenta y a menudo imprecisa. Dado que aspectos como la madurez o el estado de salud se pueden estimar con precisión a partir de características visuales se han desarrollado múltiples algoritmos de procesamiento de imágenes para evaluar el estado de plantas y cosechas [4]. En el área frutícola se han desarrollado múltiples estudios de sistemas de reconocimiento de madurez. Halstead et al. [67] entrenó una red Faster-RCNN para detectar pimentones en tres etapas distintas de madurez, green (verde), mix (inmaduro) and red (maduro), tomando cada una como una clase. La matriz de confusión mostró un accuracy de 94,0[%], 62,7[%] y 89,5[%] para los pimentones green, mix y red, respectivamente. Fan et al. [55] comparó la precisión de distintas versiones de la red YOLOv5 en la tarea de detección de frutillas, tomando 4 etapas de madurez como clases, immature, close to maturity, mature and bad/damaged. En promedio, el 83,5[%] de las detecciones fueron correctamente clasificadas.

La detección de plagas y enfermedades también es una tarea de gran importancia para asegurar el bienestar y calidad de una cosecha. Muchos procesos de cosecha tienen etapas de control de calidad post-cosecha [68–70], sin embargo aplicar esta opción tan tarde en el proceso de producción permite que plagas y enfermedades se esparcen de forma innecesaria. Lo que ha llevado a la investigación y desarrollo de sistema de reconocimiento temprano de enfermedades [71]. Por ejemplo, Liu et al. [14] desarrolló un sistema multispectral 3D para la detección de invertebrados en hojas. Similarmente Tian et al. [13] desarrolló un sistema de detección de lesiones causadas por enfermedades en manzanas. Se observó que usando la red YOLOv3 con el backbone Densenet era capaz de detectar la mayoría de las lesiones, obteniendo un F1 score de 0.82 y superando a la versión original de YOLOv3 que obtuvo 0.80.

3.3.2. Estimación de producción

La mayoría de los agricultores firma acuerdos de venta meses antes de la cosecha. Lo que a menudo requiere de acuerdos que subestiman la producción total, para evitar comprometerse más allá de sus capacidades. Esto causa pérdidas para los agricultores, por ende, generar estimaciones precisas de la producción ha sido un tema de alto interés [3], que generalmente se ha realizado en base producciones anteriores y mediciones de distintas variables, como temperatura, irrigación, usó de fertilizantes, usó de pesticidas, entre otros factores. Sin embargo,

a día de hoy las estimaciones aún tienen graves errores, dada la gran cantidad de factores fuera del control del agricultor, como el clima, la presencia de plagas y enfermedades, entre otras.

En la industria frutícola, la detección de frutas presenta una alternativa económica, no invasiva y precisa para la estimación de la producción, ya que los algoritmos pueden contar frutas que es un dato que permite estimar producción directamente. Un sistema autónomo de detección y conteo de frutas que pueda dar una estimación precisa sería de gran ayuda para los agricultores. Para ello se han desarrollado múltiples sistemas, que generalmente utilizan cámaras de vídeo para tener un registro continuo de secciones de una plantación. Estos métodos utilizan un sistema de seguimiento para contar los frutos observados y evitar problemas de doble conteo. Las imágenes de vídeo también aseguran la visibilidad de todas las frutas. Este tipo de método consiste en general debe resolver tres tareas:

1. Detección: La fruta se detecta cuadro por cuadro usando algún algoritmo de detección, con lo que se tiene un registro de coordenadas en cada cuadro del vídeo.

2. Seguimiento: Dado que la posición de cada fruta cambia a lo largo del vídeo, es importante desarrollar un sistema que haga un seguimiento de la posición de cada fruta, considerando la cinemática que se ha observado en esta. En esta etapa es habitual incorporar datos adicionales de sensores como datos de movimiento respecto a la plataforma que transporta la cámara (Inercia, velocidad y orientación) para tener en cuenta el movimiento de la cámara o datos de distancia (vía cámaras de profundidad o LIDAR) para hacer una reconstrucción 3D de la posición de la fruta.

3. Asociación: Se aplica un sistema de asociación de las detecciones actuales con las frutas registrados previamente. La idea es identificar frutas individualmente a lo largo del vídeo, para poder contabilizar estas.

Halstead et al. [67] empleó un sistema de detección y conteo de pimentones. Este método define a una fruta con una identificación y una ubicación en la imagen (coordenadas 2D). Cuando se recibe una nueva detección D_j esta se asocia a una antigua fruta F_i en base al cálculo de IoU entre estas. Este acercamiento es sencillo y solo requiere de información visual para la detección y conteo, sin embargo, demostró una alta precisión con un error absoluto promedio del 4,1[%]. Otros métodos aplican métodos de seguimiento más complejos que incluyen mapear las frutas en un plano 3D. Häni et al. [18] evaluó este tipo de sistemas en 4 bases de datos de manzanas que cubren distintos grados de iluminación y madurez. Se utilizó la red Faster-RCNN para la detección de manzanas y se usó un software de reconstrucción 3D para ubicar las manzanas detectadas en filas de manzanos. La reconstrucción 3D se realizó tomando como vídeos de la cara frontal y trasera de cada fila. Los resultados indican que la precisión de conteo en los 4 bases de datos fue de 96,97[%], 96,72[%], 94,81[%], 91,97[%] respectivamente. Liu et al. [19] empleó un sistema de tres etapas para mantener registro de las frutas y así poder contabilizarlas. La primera consiste en una detección frame a frame usando la red FCN. A continuación, se empleó un sistema de seguimiento para modelar el movimiento de la fruta en vídeos, basado en la reconstrucción 3D de las hileras, además de usando datos de inercia capturados por una plataforma robótica durante la toma de datos. Finalmente se emplea el Algoritmo Húngaro [72] para asociar las detecciones con las coordenadas de frutas previamente vistas. El método propuesto se evaluó en una base de datos de manzanas y naranjas donde se observó que el error de estimación de producción tuvo una desviación estándar de 7,8[%] y 4,1[%] en las bases de datos de naranjas y manzanas, respectivamente.

3.3.3. Recolección

En la mayoría de los países desarrollados se ha observado una escasez de mano de obra para tareas agrícolas, y sumado al aumento de la población esto puede presentar severos problemas para asegurar un suministro de comida a la población. Consecuentemente se ha expandido la investigación de sistemas autónomos de cosecha [8] con el fin de minimizar costos, aumentar la producción y asegurar estándares de calidad. En el contexto frutícola un sistema autónomo de cosecha consiste en un robot móvil capaz de detectar, recoger y almacenar la fruta en un compartimento de contención. Este tipo de sistemas requieren resolver múltiples tareas, asociadas a tres etapas:

- Toma de datos: Consiste en la adquisición de datos para poder determinar características del ambiente necesarias para navegar e interactuar en él. Un robot tiene que resolver problemas de navegación, los cuales suelen ser resueltos con cámaras y se pueden apoyar de sensores de distancia tales como LIDAR's o sistemas de navegación GPS. Para la detección de frutas las cámaras suelen ser el sensor más usado, aunque para recolección se suele aplicar cámaras RGB-D para obtener información de la pose de frutas (posición y forma).

- Procesamiento y toma de decisiones: Esta etapa procesa los datos de los sensores para extraer información de utilidad y así definir cómo controlar los actuadores. Estos requieren de resolver dos tareas, desplazamiento hacia la planta y recolección de la fruta. Para poder cosechar fruta es necesario llevar el robot hacia esta, lo que requiere de ubicar al robot en un campo de cultivo, determinar la trayectoria a seguir para alcanzar un árbol con frutas. Una vez ahí el vehículo queda estático y el sistema de recolección comienza a cosechar (ver Fig. 3.6). Este proceso en primer lugar requiere de la detección de las frutas, usualmente con una red ligera para asegurar detecciones en tiempo real, usualmente en algún procesador con capacidades limitadas en el robot. Posteriormente se debe estimar la posición de la fruta usando información de profundidad. Luego de estimar la posición de la fruta este tipo de sistemas debe llevar el brazo robótico hacia esta. Este problema es uno cinemática inversa que debe considerar el alcance del brazo, obstáculos como ramas y otras frutas, donde es posible que frutas visibles estén fuera del alcance. Una vez el brazo está en posición, se debe activar una pinza y llevar la fruta hasta un contenedor en el robot. El proceso de recolección se repite hasta que toda la fruta al alcance fue recolectada. Entonces la plataforma debe moverse a la siguiente ubicación.

  Se han desarrollado múltiples sistemas que realizan estos procesos. Por ejemplo, Li et al. [63] desarrolló un sistema de estimación de pose de manzanas de 4 etapas, partiendo de la detección y segmentación de manzanas a partir de la red YOLOACT. Luego a partir del BB de la fruta y el valor de profundidad obtenido con una cámara RGB-D se calcula el frustum que encerraba a dicha fruta, con lo que se obtenía una nube de puntos donde la fruta estaba encerrada. Posteriormente se aplica el algoritmo de clusterización DBSCAN para extraer los puntos correspondientes a la fruta. Finalmente se ajustan la nube de puntos resultantes a una esfera, para así obtener una ubicación y radio de esta. Otros autores realizan una estimación de la posición con una red neuronal, como lo hizo Kang et al. [38] que realizo la estimacion en dos etapas, la primera fue consiste en usar la red Mobile Dasnet para obtener segmentos de frutas presentes en árboles de manzanas, para luego usar el canal de profundidad de una cámara RGB-D y dichos

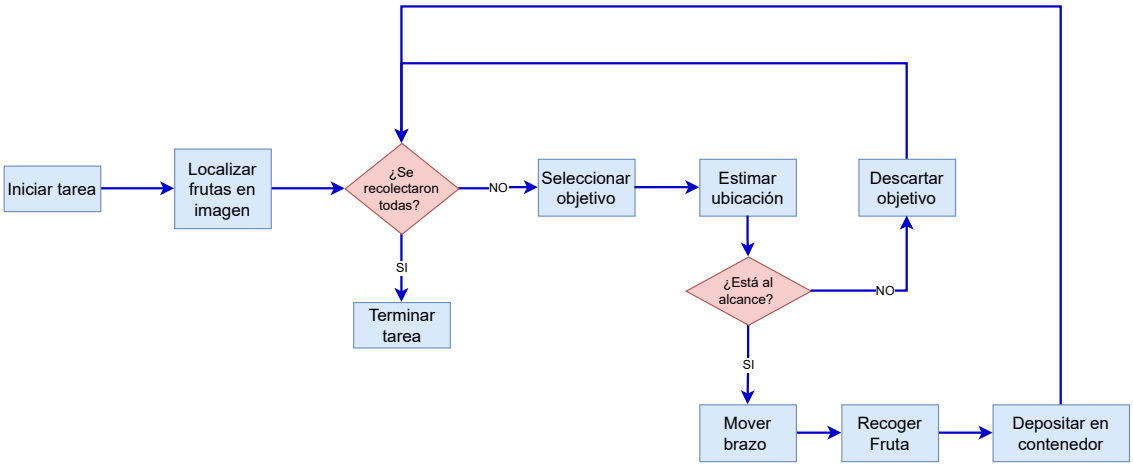


Figura 3.6: Proceso de recolección de fruta.

segmentos como entrada de la red PointNet [62] para estimar el elipsoide que define la manzana. Finalmente se usa un brazo robótico para recoger la fruta. El sistema de recolección demostró tener un porcentaje de éxito de 85[%] y 80[%] en ambientes de invernadero y exteriores, respectivamente.

- Actuación Robótica: Posterior a la toma de decisiones se ejecutan las órdenes en los actuadores. Un sistema convencional estará compuesto por un brazo robótico, un mecanismo de agarre (MA) y una plataforma para mover el sistema. Las plataformas suelen ser vehículos autónomos de 4 ruedas, donde se monta todo el sistema. Los brazos llevan los MA hasta la fruta para poder separarlas del árbol. Es usual usar brazos con 6-8 grados de libertad o degrees of freedom (DoF's) de tipo rotativo, como el brazo usado por Onishi et al. [21] para desarrollar un sistema de recolección de manzanas capaz de cosechar una manzanas en 16 [s] utilizando una pinza como actuador. Si bien brazos con varios DoF's son los más comunes por su versatilidad y robustez, es posible utilizar sistemas con menos DoF's. Zhang et al. [11] desarrolló un sistema de recolección de manzanas con un brazo de 3 DoF, generados a partir de un riel de desplazamiento recto y dos uniones rotativas en las orientaciones pitch y yaw. En este estudio se utilizó un sistema de succión como mecanismo de agarre, lo que permite recoger la fruta desde cualquier ángulo.

El desarrollo de sistemas de este tipo sigue estando en una etapa temprana por lo que la mayoría de los estudios se enfocan en solo una etapa de la recolección como lo es la detección o estimación 3D de la fruta (ver Tabla 3.1). En los casos donde se han desarrollado robots de recolección, la mayoría ha estudiado su rendimiento en condiciones de laboratorio, debido a las dificultades intrínsecas de esta tarea. Sin embargo, hay algunos autores que han evaluado prototipos en terreno, como lo hizo Arad et al. [10], que desarrolló un sistema recolección de pimentones y lo evaluó en un invernadero comercial. Se evaluó el sistema en múltiples filas de pimentones, evaluando cultivos normales y podados (para disminuir la oclusión). Se evaluó la razón de cosecha (RC) en estos dos tipos de cultivos:

$$RC = \frac{\text{Fruta recogida}}{\text{Fruta total}} \tag{3.1}$$

Los resultados de [10] indican un tiempo promedio de cosecha de 24[s] por fruta y una

RC de 0.61 y 0.18 para los cultivos modificados y normales respectivamente. Los autores argumentan que mejoras significativas se pueden lograr en términos de RC y velocidad, enfocándose en modificar la capacidad del sistema en detectar frutas, aumentar la visibilidad de estas con otras metodologías de cultivo y modificar el gripper para que sea más robusto y ligero.

Tabla 3.1: Resumen de redes, métodos, bases de datos y aplicaciones en estudios revisados.

Artículo	Año	Red	Backbone	Fruta	Multi escala	Métodos de Aumento Modernos	Módulos Ligeros	Modificaciones específicas	Segmentación	Aplicaciones
TomatoDet: Anchor-free detector for tomato detection [73]	2022	CenterNet	DLA34	Tomates	X			X		DETECCIÓN
Adaptive Active Positioning of Camellia oleifera Fruit Picking Points: Classical Image Processing and YOLOv7 Fusion Algorithm [74]	2022	YOLOv7	YOLOv7	Camelia Oleifera	X		X			DETECCIÓN
A fast and efficient green apple object detection model based on Foveabox [45]	2022	Foveabox	EfficientNetV2-S	Manzana	X		X	X		DETECCIÓN
Localizing Small Apples in Complex Apple Orchard Environments [75]	2022	Attention Mask	ResNet	Manzana					X	DETECCIÓN
Detection and Segmentation of Mature Green Tomatoes Based on Mask R-CNN with Automatic Image Acquisition Approach [56]	2021	MASK-RCNN	SENet154-vd-FPN	Tomates	X	X			X	DETECCIÓN
Strawberry Maturity Recognition Algorithm Combining Dark Channel Enhancement and YOLOv5 [55]	2021	YOLOv5	YOLOv5	Frutilla	X	X	X			MONITOREO DE MADUREZ
3D shape sensing and deep learning-based segmentation of strawberries [54]	2021	SegNet	VGG16	Frutilla	X	X		X	X	DETECCIÓN
Occluded Apple Fruit Detection and Localization with a Frustum-Based Point-Cloud-Processing Approach for Robotic Harvesting [63]	2021	YOLACT++	ResNet101	Manzana					X	DETECCIÓN
Lightweight Fruit-Detection Algorithm for Edge Computing Applications [53]	2021	CSPNet	CSPNet	Manzana, Tomates, Naranja	X		X	X		DETECCIÓN
Apple Detection in Complex Scene Using the Improved YOLOv4 Model [47]	2021	YOLOv4	EfficientNetB0	Manzana	X	X	X	X		DETECCIÓN
Fruits yield estimation using Faster R-CNN with MIoU [65]	2021	Faster-RCNN	VGG16	Mango, Manzana, Tomates y otras				X		ESTIMACIÓN DE PRODUCCIÓN
A Real-Time Apple Targets Detection Method for Picking Robot Based on Improved YOLOv5 [34]	2021	YOLOv5	YOLOv5	Manzana	X		X	X		DETECCIÓN
Development of a sweet pepper harvesting robot [10]	2020	Visual Análisis		Pimenton						RECOLECCIÓN
System Design and Control of an Apple Harvesting Robot [11]	2020	MASK-RCNN	ResNet101-FPN	Manzana					X	RECOLECCIÓN
A visual detection method for nighttime litchi fruits and fruiting stems [76]	2020	YOLOv3	Darknet53	Lychee	X		X		X	DETECCIÓN
Tomato Fruit Detection and Counting in Greenhouses Using Deep Learning [77]	2020	MASK-RCNN	ResNext101	Tomates					X	MONITOREO DE MADUREZ
Deep Learning with Data Augmentation for Fruit Counting [20]	2020	Faster-RCNN	ResNet50	Manzana, Limon, Pear y otras		X				ESTIMACIÓN DE PRODUCCIÓN
ON TREE GUAVA FRUIT DETECTION AND YIELD ESTIMATION [78]	2020	Visual Análisis		Guava						DETECCIÓN
Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association [79]	2020	MASK-RCNN	ResNet101	Uvas	X				X	DETECCIÓN
Robust Cherry Tomatoes Detection Algorithm in Greenhouse Scene Based on SSD [46]	2020	SSD	InceptionV2	Tomates cherry			X			DETECCIÓN
Detection and Characterization of Cherries: A Deep Learning Usability Case Study in Chile [16]	2020	Faster-RCNN	InceptionV2	Cereza						DETECCIÓN
Passion fruit detection and counting based on multiple scale Faster R-CNN using RGB-D images [52]	2020	MS-FRCNN	ResNet101	Fruta de la pasión	X			X		DETECCIÓN
Applying Faster R-CNN and Mask R-CNN on the MinneApple Fruit Detection Challenge [15]	2020	MASK-RCNN	ResNext101	Manzana					X	DETECCIÓN
Real-Time Fruit Recognition and Grasping Estimation for Robotic Apple Harvesting [38]	2020	Mobile Dasnet	MobileNet	Manzana	X		X	X	X	RECOLECCIÓN

Artículo	Año	Red	Backbone	Fruta	Multi escala	Métodos de Aumento Modernos	Módulos Ligeros	Modificaciones específicas	Segmentación	Aplicaciones
Fruit Detection, Segmentation and 3D Visualisation of Environments in Apple Orchards [44]	2019	Dasnetv2	lw-ResNet	Manzana	X		X		X	DETECCIÓN
MinneApple: A Benchmark Dataset for Apple Detection and Segmentation [42]	2019	Faster-RCNN	ResNet	Manzana					X	DETECCIÓN
CropDeep: The Crop Vision Dataset for Deep-Learning-Based Classification and Detection in Precision Agriculture [17]	2019	YOLOv3	Darknet53	Tomates, Limon y otras	X		X			DETECCIÓN
Lychee Fruit Detection Based on Monocular Machine Vision in Orchard Environment [80]	2019	Visual Análisis		Lychee						DETECCIÓN
Detection of Apple Lesions in Orchards Based on Deep Learning Methods of CycleGAN and YOLOV3-Dense [13]	2019	YOLOv3	Densenet	Plaga	X	X				MONITOREO DE MADUREZ
Multi-modal Deep Learning for Fruit Detection Using RGB-D Cameras and their Radiometric Capabilities [64]	2019	Faster-RCNN	VGG16	Manzana				X		DETECCIÓN
Fruit Detection and Segmentation for Apple Harvesting Using Visual Sensor in Orchards [49]	2019	Dasnet	ResNet101	Manzana	X		X		X	DETECCIÓN
An automated fruit harvesting robot by using deep learning [21]	2019	SSD	VGG16	Manzana			X			RECOLECCIÓN
An adaptable approach to automated visual detection of plant organs with applications in grapevine breeding [81]	2019	FCN	VGG16	Uvas			X	X	X	MONITOREO DE MADUREZ
Fruit Quantity and Ripeness Estimation Using a Robotic Vision System [67]	2018	Faster-RCNN	VGG16	Pimenton						MONITOREO DE MADUREZ ESTIMACIÓN DE PRODUCCIÓN
Robust Fruit Counting: Combining Deep Learning, Tracking, and Structure from Motion [19]	2018	FCN	VGG16	Manzana					X	ESTIMACIÓN DE PRODUCCIÓN
Apple Counting using Convolutional Neural Networks [18]	2018	Faster-RCNN	Alexnet	Manzana						ESTIMACIÓN DE PRODUCCIÓN
Green Grape Detection and Picking-Point Calculation in a Night-Time Natural Environment Using a Charge-Coupled Device (CCD) Vision Sensor with Artificial Illumination [37]	2018	Visual Análisis		Uvas						DETECCIÓN
Crop yield estimation using deep learning [82]	2017	Inceptionv3	Inceptionv3	Manzana and Uvas					X	DETECCIÓN
Early Yield Prediction Using Image Analysis of Apple Fruit and Tree Canopy Features with Neural Networks [83]	2017	Visual Análisis		Manzana						ESTIMACIÓN DE PRODUCCIÓN
Counting Apples and Oranges with Deep Learning: A Data Driven Approach [84]	2017	FCN	VGG16	Manzana Naranja						ESTIMACIÓN DE PRODUCCIÓN
Deep Fruit Detection in Orchards [43]	2017	Faster-RCNN	VGG16	Manzana, Mango and Almendras						DETECCIÓN
DeepFruits: A Fruit Detection System Using Deep Neural Networks [60]	2016	Faster-RCNN	VGG16	Pimentons, Manzana, Naranja y otras						DETECCIÓN
Automated visual fruit detection for harvest estimation and robotic harvesting [39]	2016	Visual Análisis		Manzana and Frutilla						DETECCIÓN
Automatic fruit recognition and counting from multiple images [85]	2014	Visual Análisis		Pimenton						ESTIMACIÓN DE PRODUCCIÓN

Capítulo 4

Estado del arte en detección de objetos basado en Deep Learning

Los algoritmos de detección de objetos son capaces de identificar y localizar objetos en imágenes, lo que permite el desarrollo de aplicaciones que puedan interpretar la información visual. Una de las bases de datos más relevantes para la evaluación de algoritmos de detección es la base de datos de COCO [5]. Esto se debe a que esta base de datos tiene un alto número de imágenes etiquetadas, múltiples clases (91 categorías), abarca una gran variedad de escenarios y es de carácter público, lo que facilita la discusión y comparación de algoritmos de detección. Esto ha servido para mejorar el rendimiento de algoritmos de detección de objetos, y en particular de redes neuronales. A modo de ejemplo la red Faster-RCNN [27], publicada en 2016, presento un puntaje mAP de 34,9[%] en la base de datos de COCO, el cual demostró ser el mejor puntaje entonces, sin embargo el actual primer lugar, InternImage [86], obtuvo un 65,4[%]. Este gran progreso se dio a partir de una serie de desarrollos en el uso de operaciones, arquitecturas y metodologías de entrenamiento de DNN.

La mayoría de las redes y metodologías usadas en detección de frutas han sido primero desarrolladas para la detección de objetos en COCO. Es por ello que esta sección presenta una revisión de los algoritmos más relevantes en tareas de procesamiento de imágenes, con un énfasis en aquellos empleados en detección de objetos. Para esta revisión se evaluaron múltiples trabajos en el área de DO's, considerando su relevancia en otros estudios así como los beneficios que ofrecen en términos de velocidad y precisión para detección de objetos.

Este capítulo se divide en cinco áreas de desarrollo en DO, las cuales son: Estructura de backbone, Diseño de redes, Mecanismos de atención, Modelamiento de Predicciones y Metodologías de entrenamiento (ver Fig. 4.1).

4.1. Estructura de backbones

El diseño de las redes debe tener en consideraciones propiedades que permitan a la red extraer información significativa de las imágenes, por lo que se deben definir estructuras a nivel micro y macro en los backbones. Las principales propiedades utilizadas para mejorar el rendimiento en las redes modernas son *Módulos Eficientes* y *Representaciones multi-escala*.

- Módulos Multi-escala: Todas redes modernas incorporan estructuras que pueden extraer información de distintos niveles de resolución, para así poder reconocer objetos de distin-

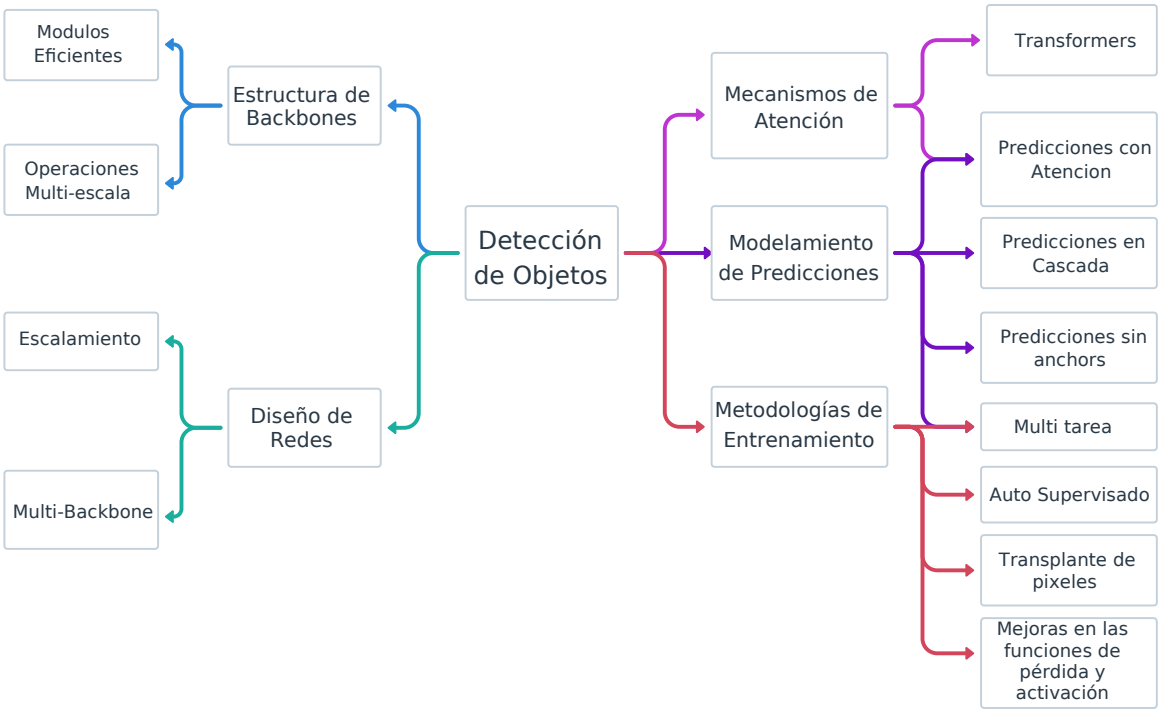


Figura 4.1: Tendencias en detección de objetos en la competencia de COCO.

tas escalas [33, 87–91]. La mayoría utiliza algún método de combinación de resolución similar al usado en FPN [48], pero se han implementado otros módulos multi-escala, como las operaciones de Deconvolución y Atrous Convolution [51]. A modo de ejemplo, se empleó una versión modificada de esta última operación en la red DetectoRS [91], cosa que mejoró su mAP en COCO en 3[%].

- Módulos Eficientes: La demanda de predicciones en tiempo real ha aumentado dada su utilidad para todo tipo de aplicaciones en robótica, IoT y sistemas autónomos en general. De igual modo es importante mantener o aumentar la precisión de las redes, los que son objetivos difíciles de conseguir simultáneamente. En respuesta a esta doble necesidad se han desarrollado múltiples operaciones eficientes, las cuales ofrecen mejoras en términos de velocidad y precisión. Esto se logra al utilizar operaciones con menor costo computacional que las operaciones tradicionales, pero sin sacrificar capacidad de inferencia. Ejemplo de esto son la operación de convolución agrupada empleada por Resnext [92] o operaciones Depth-wise separable convolutions (DWSC) usada por EfficientDet [93]. A modo de ejemplo, si se compara el costo computacional de una capa convolucional (C^{conv}) con el costo de una capa DWSC (C^{DWSC}) se tiene que:

$$\begin{aligned} C_{conv} &= C_{in}C_{out}HW3^2 \\ C_{depth} &= C_{in}HW3^2 + C_{in}C_{out}HW \end{aligned} \tag{4.1}$$

En este caso para C_{out} mucho mas grande que 3, el costo computacional es aproximadamente un noveno.
El tipo de operación eficiente más relevante en la actualidad son los módulos CSP [57].

Este tipo de arquitectura fue presentada el 2019 por Wang et al. [57] y consiste en reemplazar un bloque operacional de una red por su versión CSP. Dado un capa tensorial de N canales esta se divide en dos (ver Fig. 4.2).

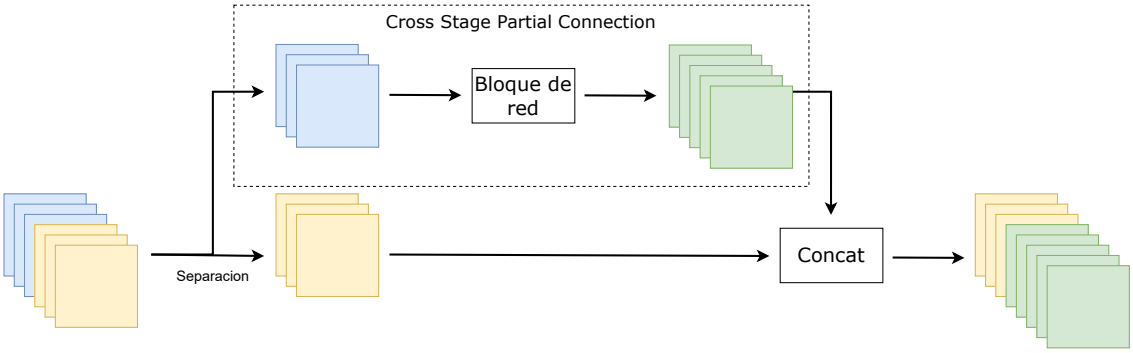


Figura 4.2: Esquema de bloque CSP [57].

La primera rama, conocida como rama Cross-Stage, se le aplica el bloque operacional para generar un nuevo tensor, el cual es concatenado con la segunda rama. Normalmente se aplicaría el bloque de red sobre la totalidad del tensor, pero este sutil cambio ha demostrado disminuir el costo computacional y mejorar la capacidad de inferencia. Los autores tomaron como ejemplo la red Resnet y reemplazaron los módulos Resnet por módulos CSPResnet, y observaron una mejora tanto en precisión como en velocidad. Módulos CSP combinan una skip connection parcial con otras operaciones, y se ha observado que tiene la capacidad de reducir el costo computacional de una red hasta en un 20 [%], en la medida en que aumenta la capacidad de inferencia de esta. La versión más importante en términos de velocidad son redes basadas CSPDarknet, usadas por ScaledYOLOv4 [33], YOLOR [94], YOLOv5 [34] y YOLOv7 [35].

4.2. Diseño de redes

El desarrollo de módulos eficientes o el uso de transformers [95] han servido para avanzar las capacidades de detectores. Aunque estos módulos son muy importantes en el desarrollo de una red, el uso de un módulo en particular no define la totalidad de la red. Esta tarea requiere elegir el tamaño, número de capas, dimensiones por capa y otros parámetros en la red, lo cual no es una decisión trivial. Los métodos convencionales relegan esta decisión al criterio del autor, métodos de prueba y error u otros criterios convencionales que buscan mejorar la precisión y/o velocidad de la red. En este sentido, múltiples autores desarrollan diferentes modelos de una misma red, para ofrecer distintos niveles de rendimiento, lo que resulta útil para distintas aplicaciones con distintos niveles de capacidad computacional. El método más común para hacerlo es mediante la iteración del mismo macro-módulo, modificando diferentes hiper parámetros en cada uno, como la resolución o número de canales. Algunos ejemplos de esto son las redes ResNet [96] y Swin [89]. Sin embargo, el aumento directo no siempre se refleja en mejores resultados, por ejemplo, la red ResNet-152 ofrece mejores resultados en ImageNet que la red ResNet-201. Otros autores han optado por mecanismos de crecimiento basados en escalado o *Scaling* [33, 35, 58, 97]. El escalado es una metodología para diseñar una nueva red basada en una red modelo. Esto se hace aumentando su tamaño en múltiples dimensiones, ancho (número de canales), resolución y profundidad (número de capas),

simultáneamente según alguna heurística. Tan et al. [58, 93] propusieron un sistema en el que la red crecía en sus tres dimensiones, ancho, resolución y profundidad, en función de un factor de escalado compuesto . De manera similar, Scaled-YOLOv4 propuso un método similar para ofrecer diferentes versiones de YOLOv4 [33] aumentando simultáneamente sus tres dimensiones.

Otra forma de mejorar el rendimiento de una red es el uso de múltiples backbones. DetectoRS [91] usa un bakbone de Resnext recursivamente, tomando las salidas de cada nivel de resolución como una entrada adicional en el siguiente ciclo. Por otro lado, Liang et al. [98] propusieron el uso de múltiples redes troncales pre entrenadas conectadas a través de conexiones compuestas. Este último método se utilizó para desarrollar la red Dual Swin, compuesta por dos backbones Swin-L, superando así a la red Swin-L.

4.3. Mecanismos de Atención

Uno de los avances más importantes en Aprendizaje profundo ha sido el desarrollo de operaciones de atención. Este tipo de operaciones consiste en módulos que combinan información de distintas secciones del tensor, en la medida en que atenúan la influencia de ciertos componentes poco relevantes, mientras que resaltan otros. Este proceso de atenuación y énfasis se realiza en base a la información de la misma imagen, lo que permite un procesamiento especifico por imagen, en oposición a usar los mismos pesos para cada imagen. Uno de los primeros ejemplos fueron las operaciones Squeeze and Excitation (SE) [99]. Este módulo pondera canal a canal el mapa de características, usando un vector de tamaño N=Número de canales. Dicho vector se genera a partir del mismo mapa de características que se pondera, el cual se compone de valores entre [0,1]. Este acercamiento genera un vector que pondera la información de cada canal. En la actualidad la operación más importante de atención es la operación de self-attention [95]. Self-Attention originalmente se diseñó para tareas de procesamiento de Lenguaje Natural, al considerar cada palabra como un vector de características llamados token. Este módulo se ha expandido al uso en imágenes al considerar imágenes como un conjunto de ventanas y considerar dichas ventanas como tokens. El mecanismo de self-attention primero genera un embedding por cada token en la forma de un vector $E_{i,j}$. Dada una imagen de $H \times W$ tokens, se calcula 3 vectores a partir del embedding, $Q_{i,j}$, $K_{i,j}$ y $V_{i,j}$, denominados Query, Key y Value, utilizando operaciones Fully Connected.

Luego se procede a calcular el mapa de atención correspondiente de cada token $A^{i,j}$ (ver Fig. 4.3). Dicho mapa es una matriz de dimensiones $H \times W$, donde cada elemento se define como el resultado de una función entre el query $Q_{i,j}$ y la key $K_{k,l}$ correspondiente de algún token. Comúnmente se usa el producto punto entre K y $Q_{i,j}$ para luego aplicar softmax y así obtener valores en el rango de [0,1], los cuales representa un valor de atención que el token $T_{i,j}$ le atribuye al token $T_{k,l}$:

$$A^{i,j}_{k,l} = Softmax(F(K_{k,\bullet}, Q_{i,j})_l) \tag{4.2}$$

Posteriormente se calcula el embedding de salida como una suma ponderada de los valores $V_{i,j}$ según su coordenada correspondiente en el mapa de atención:

$$E^{out}_{i,j} = \sum_{k=1,l=1}^{H,W} A^{i,j}_{k,l} V_{k,l} \tag{4.3}$$

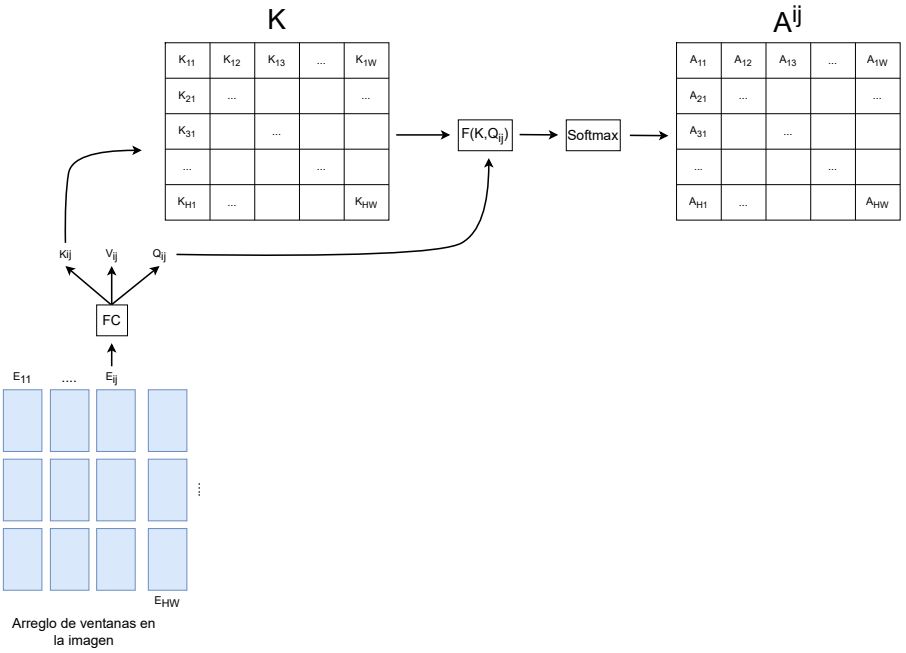


Figura 4.3: Diagrama de mapas de atención en operaciones de self-attention.

La característica más significativa de este tipo de operaciones es la capacidad de asociar zonas de la imagen con otras que contengan al mismo objeto o información relevante de forma selectiva. Esto se puede observar en la ecuación 4.3 donde un mapa de atención solo tendrá valores significativos en ubicaciones relacionadas con el token $T_{i,j}$. La Fig. 4.4 es un ejemplo de mapas de atención de la red DETR entrenada en COCO, donde se muestran las ubicaciones de 4 tokens distintos y sus respectivos mapas de atención, donde se observa que cada token presta atención mayoritariamente a tokens del mismo objeto, en este caso de la misma vaca de una red DETR.

Este tipo de operaciones se han expandido y empleado en distintos tipos de módulos, como el módulo Transformer [95] que combina una estructura Encoder-Decoder con múltiples operaciones de self-attention. Este tipo de módulos han tenido una gran relevancia en el SOTA de detección y segmentación de objetos, a tal grado que a la fecha los 10 primeros lugares en la competencia COCO los emplean, tales como Swin [89], Swinv2 [90], DINO [88] e InternImage [86]. Cabe mencionar que los primeros 5 lugares en la competencia COCO-dev test usan algún módulo Transformer, principalmente basados en la cabeza de predicción DINO.

4.4. Modelamiento de Predicciones

La cabeza de predicción de una red genera predicciones en base al mapa de características generado por el backbone. Se predicen las coordenadas y clase del objeto, así como otro tipo de datos asociados como máscaras de cada objeto. El cómo realizar esta tarea ha sido un tema altamente estudiado dado el aumento de precisión que esto puede conllevar. Los principales avances en esta área se dividen en 4 sub-áreas:

- Predicciones basadas en atención: Las operaciones de atención también puede ser usadas para la etapa de predicción y se ha observado una gran utilidad en este aspecto. La red

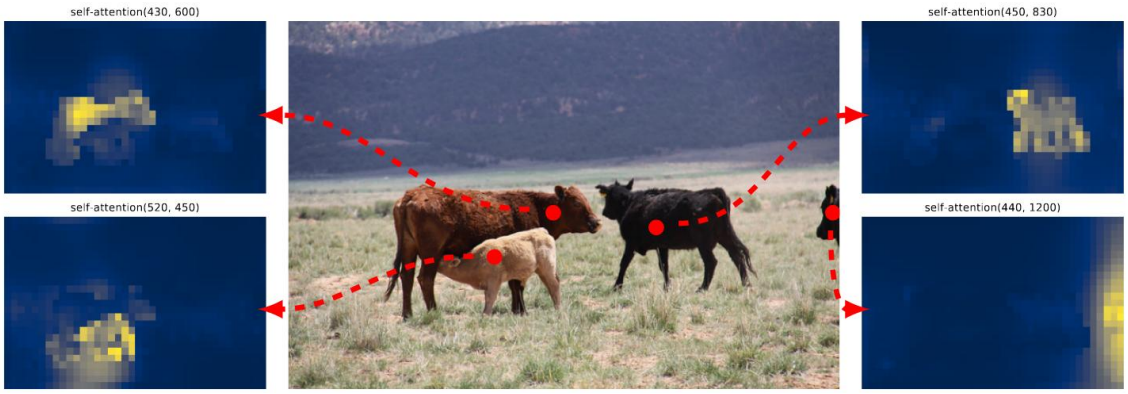


Figura 4.4: Ejemplo de mecanismo de atención en DETR [100]

DyHead [101] utiliza una cabeza de predicción que aplica módulos de atención scale-aware, spatial-aware y task-aware, con un mecanismo similar a SE. Otras arquitecturas usan módulos transformers para generar predicciones, principalmente aquellas basadas en DETR [86–88, 100]. En particular DINO [88] es la cabeza más precisa en la competencia COCO dev-test, la cual es una versión mejorada de DETR. Dicha arquitectura es usada por la mayoría de las redes en el top 10 de la competencia, incluyendo el actual primer lugar, InternImage [86].

- Predicciones en cascada: Cai et al. [102] desarrolló un método de detección de objetos en múltiples etapas sucesivas, donde cada etapa genera predicciones a partir de candidatos, similarmente a como lo hacen las redes Two Stage, pero en este caso se considera como candidatos a las predicciones generadas por módulos anteriores. De este modo se generan predicciones en cascadas. Etapas posteriores se les demanda una mayor precisión dado que sus candidatos ya han pasado por etapas de filtrado. Ejemplos comunes de este tipo son las redes Cascade-RCNN [102] y Hybrid Task Cascade [29] (HTC).

- Predicciones sin anchors: Existen dos acercamientos convencionales para la estimación de BB's, y estas son redes Two-stage y One-stage. Sin embargo, recientemente se han desarrollado una serie de algoritmos que no dependen de anchors pre-definidos, conocidos como redes sin anchors. Estas redes pueden generar BB's de cualquier tamaño y razón de aspecto, permitiendo así una mayor flexibilidad. Ejemplo de este tipo de redes son CenterNet [103] y PyCenterNet [104] que realizan la estimación de BB's usando keypoints, que son 3 tipos de ubicaciones en la imagen que definen un BB. Estos son una esquina superior izquierda (TL), una esquina inferior derecha (BR) y un centro del BB. Una predicción se define como una tripleta de keypoints en las tres categorías, TL, BR y Center. Otra familia de redes sin anchors, son aquellas que realizan una regresión directamente para estimar la coordenada de los objetos a partir de todas las características. Las redes más conocidas de este tipo son aquellas de la familia DETR [87, 88, 100].

- Predicciones de múltiples tareas [1]: Las redes tienen una sección denominada como el backbone, el cual es usado para procesar y extraer características de la imagen, las cuales son posteriormente usadas para realizar predicciones. Se ha observado que realizar predicciones de múltiples tareas, ayuda que el backbone sea capaz de inferir información

[1] Esta área de desarrollo es parte de Modelamiento de predicciones y Metodologías de entrenamiento, por lo que se omite de esta última sección.

más detallada de la imagen. En consecuencia, múltiples redes que realizan tareas de detección y segmentación presentan mejores resultados que aquellas que solo realizan una. Redes de detección y segmentación tienen acceso a más información durante el entrenamiento, y la afinidad entre ambas tareas permite el uso de las mismas características para identificar objetos. También se ha observado que otras tareas pueden ofrecer mejoras significativa, tales como Multi-label Classification y keypoint detection [94, 101]. En particular, tareas asociadas con el procesamiento de lenguaje han logrado mejoras significativas. A modo de ejemplo, Li et al. [105] uso un método de pre-entrenamiento basado en el emparejamiento de texto e imágenes, conocido como phrase grounding para mejorar el rendimiento de la red Dyhead [101] con lo que obtuvo una mejora de 0.9[%]. Phrase grounding consiste en asociar palabras en una oración con objetos en una imagen, detectando dichos objetos. Esto permite a la red asociar información lingüística con información visual durante el entrenamiento. Múltiples trabajos han demostrado que definir tareas que involucren ambos dominios mejora consistentemente el rendimiento de DNN's [106–109].

4.5. Metodologías de entrenamiento

Típicamente el entrenamiento de las redes consiste en emplear datos etiquetados y algún optimizador para aplicar el algoritmo de backpropagation, y así ajustar los pesos de la red. Estos elementos son la base para cualquier entrenamiento, sin embargo, se han logrado importantes avances modificando distintos aspectos del proceso de entrenamiento. Se han desarrollado sofisticadas metodologías y funciones asociadas con las imágenes usadas, el tipo de etiquetas o el flujo de gradiente. De estos se pueden destacar tres áreas de desarrollo enfocadas en el entrenamiento; Mejoras en funciones de pérdida y activación, Trasplante de píxeles y Entrenamiento auto supervisado.

4.5.1. Mejoras en las funciones de pérdida y activación

El proceso de entrenamiento a través de backpropagation es una de las etapas más estudiadas en el aprendizaje profundo, dada las mejoras en rendimiento que se pueden lograr. Uno de los aspectos más importantes de backpropagation es la propagación del flujo del gradiente durante el entrenamiento, ya que, este determina la forma en que el gradiente afecta a los pesos de la red a través de las distintas capas. Una de las primeras funciones de activación para DNN's fue la operación ReLu, pero esta función tiene el problema de que tiene zonas de gradiente nulo y una derivada discontinua, lo que genera un comportamiento irregular. Por ello múltiples funciones se han evaluado para tener un comportamiento similar a ReLu, pero con una derivada continua para generar para generar comportamientos regulares, como lo son las funciones Swish [110], Mish [111] y GELU [112].

Funciones de pérdida con un carácter discontinuo como NMS o IoU, tienen el problema de que consideran a múltiples predicciones como incorrectas si no están por encima de cierta cota. Por ejemplo, NMS considera solo una predicción correcta para cada objeto en la imagen, aun cuando hay otras predicciones que pueden ser altamente precisas, lo que entrega información inconsistente a la red. Acercamientos como Soft-NMS [113] solucionan este problema al considerar grados de superposición. Algo similar ocurre con IoU y generalized IoU [114]. Por otro lado, una función de pérdida importante para arquitecturas One-stage

es Focal Loss [115]. Se ha observado que las arquitecturas de una etapa sufren de un imbalance entre background y objetos reales, siendo estos últimos mucho menos numerosos. Esto impacta negativamente al entrenamiento, ya que, esto les da un peso significativo en la función de costo a secciones de la imagen de poco interés. Focal Loss minimiza el impacto de estas secciones de la imagen y se ha observado que mejoran significativamente la precisión de cualquier arquitectura de una etapa. En la actualidad las redes más importantes de una etapa emplean esta función, como YOLOv7 y YOLOR.

4.5.2. Trasplante de píxeles

El aumento de datos consiste en generar mas datos etiquetados para mejorar la capacidad de inferencia de la red. Acercamientos convencionales son las transformaciones geométricas y visuales, que permiten a las redes aprender sobre las diferentes condiciones en las que se pueden encontrar objetos, tomando en cuenta que estos pueden variar su tamaño, posición, inclinación, color y brillo. Este tipo de metodos permite generalizar la capacidad de inferencia de las redes sin la necesidad de etiquetar más datos, la cual es una tarea costosa y lenta.

(a) (b) (c) (d) (e)

Figura 4.5: Ejemplo de mecanismo de trasplante de píxeles. a) Imagen Base. b) Imagen donante. c) Cut-Mix. d) Copy-Paste. e) Mosaic

Siguiendo esta línea de trabajo se han desarrollado distintos mecanismos para generar más datos, donde los más relevantes actualmente son aquellos a los que me referiré como trasplante de píxeles. Este tipo de algoritmos consiste en emplear una imagen base (Ver Fig. 4.5 a) y combinarla con otra o varias otras imágenes ya etiquetadas, para crear nuevas imágenes etiquetadas. Uno de los primeros ejemplos de Pixel Transplant se vio en el algoritmo Cut-

Mix, el cual consiste en generar una nueva imagen utilizando una imagen base y una imagen donante, donde esta última entrega un recorte rectangular que es sobrepuesto en la imagen base. Si este recorte llegase a tener un objeto etiquetado este también se ve reflejado en la imagen resultante (Ver Fig. 4.5 c). Múltiples cut-mixes pueden ser incorporados en la misma imagen base. Otro ejemplo fue presentado en 2020 por Ghiasi et al. [116] conocido como el método *Copy Paste Data Augmentation*, el cual se asemeja a Cut-Mix, pero en lugar de incorporar cortes rectangulares, toma segmentos de objetos en la imagen donante y los pega en una ubicación aleatoria de la imagen base (Fig. 4.5 d)). Esto permite generar imágenes más realistas que con Cut-mix. Por otro lado, Bochkovskiy et al. [32] desarrollo un método llamado *Mosaic Data Augmentation* para entrenar la red YOLOv4, que consiste en combinar las esquinas de 4 imágenes distintas para generar una nueva (Ver Fig. 4.5 e), manteniendo las etiquetas correspondientes. A todos estos métodos se les puede incorporar transformaciones convencionales tanto a la imagen donante como a la imagen base, para obtener imágenes radicalmente distintas a las imágenes base, pero semánticamente relevantes.

4.5.3. Entrenamiento Auto-Supervisado

Dadas el gran costo en el etiquetado manual de imágenes en términos económicos y de tiempo, los entrenamientos se hacen a partir de una base de datos pública, tales como COCO [5], WIDER-FACE [117] o Open Image [25], en lugar de etiquetar imágenes cada vez que se entrena una red. Sin embargo, el interés por utilizar imágenes no etiquetadas ha aumentado recientemente. Un primer acercamiento es el conocido como Self-Training. Este método entrena un detector con imágenes etiquetadas y luego lo usa para que detecte objetos en imágenes no etiquetadas, para usar dichas predicciones como etiquetas adicionales o pseudo-etiquetas. Durante el entrenamiento estas etiquetas generadas se usan como etiquetas normales. Una segunda red es entonces entrenada usando un mix de las imágenes etiquetadas y las pesudo-etiquetadas. Este método ha sido usado por varios autores [116, 118] mostrando mejoras significativas frente a un entrenamiento puramente supervisado. El uso de algoritmos auto supervisados presenta un gran potencial dado que la cantidad de imágenes no etiquetadas similares a la base de datos de COCO es virtualmente infinita.

En 2021 Xu et al. [119] propuso el algoritmo Soft-Teacher learning, un mecanismo para etiquetar durante el entrenamiento imágenes no etiquetadas con una red Soft-Teacher y entregar las detecciones generadas como pseudo-etiquetas a una red estudiante. Se utiliza un promedio exponencial móvil de la red estudiante como el Soft-Teacher. Este arreglo permite que las pseudo-etiquetas mejoren en la medida en que la red student aprende, lo que a su vez mejora las pseudo-etiquetas, generando una retroalimentación positiva. Este acercamiento fue usado junto con la red Swin-L con lo que se obtuvo una mejora de 2,4[%] en puntaje mAP.

Capítulo 5

Mejorando la Detección de Frutas: Un enfoque moderno

A partir del análisis de las líneas de desarrollo en DF presentado en el Capítulo 3 se mostró la presencia de distintos acercamientos modernos aplicados a frutas. Por ejemplo el uso de las redes YOLO han sido frecuentemente usadas en estudios [47, 55, 74]. También se puede observar un uso generalizado de mecanismos multi-escala (ver Tabla 3.1). Por otro lado, el método propuesto por Pawara et al. [20] es similar a copy-paste [116], pero con un alcance más limitado, dado que utilizó un número acotado de máscaras para generar nuevas imágenes.

A pesar de las metodologías y modelos modernos mencionados, existe un alto número de acercamientos exitosos en DO que han sido escasamente explorados en DF. Por ejemplo, redes que realicen predicciones sin anchors o en cascada han estado claramente ausentes en los distintos trabajos evaluados. De igual modo las metodologías de entrenamiento descritas en la Sección 4.5 y los módulos transformer están ausentes en la mayoría de los estudios. Arquitecturas convencionales como Faster-RCNN y Mask-RCNN son las redes más relevantes en la mayoría de estudios en desmedro de otras alternativas más recientes (ver Fig. 3.1). Esta tendencia a usar métodos convencionales es aún más notoria cuando se considera los backbones. Los backbones más usados son VGG-16 [120], Inceptionv2 [23] y Resnet, [96], los cuales fueron publicados hace al menos 6 años y comprenden el 54[%] de los backbones evaluados. En ese tiempo han surgido alternativas de backbones tales como Renext, EfficientDet y Swin, los cuales han demostrado mejores resultados tanto en ImageNet como en COCO.

Los distintos algoritmos relevantes en DO discutidos en el Capítulo 4 han demostrado un gran potencial, y es razonable esperar resultados similares a los obtenidos en la base de datos COCO si se aplican en DF. Los estudios donde se han comparado redes modernas y redes tradicionales avalan esta hipótesis [53, 55, 74, 121]. Esta diferencia entre el uso de ciertos acercamientos se puede deber parcialmente a que aplicar algún algoritmo de detección general de objetos a una tarea específica toma tiempo, por lo que es razonable esperar un desfase, sin embargo, la mayoría de los estudios evaluados optan por usar redes convencionales, aun después de uno o más año de la publicación de alternativas más precisas.

A partir de estas observaciones esta tesis busca hacer un análisis comparativo de metodologías modernas y tradicionales. Para esto se propone una evaluación exhaustiva entre distintos métodos de aumento de datos y redes en distintas bases de datos de frutas para va-

lidar los beneficios que estas ofrecen. El resto del capítulo describe la metodología propuesta. La Sección 5.1 describe las redes a evaluar y la razón de su elección. La Sección 5.2 describe los métodos de aumento de datos a evaluar en las distintas redes. Finalmente la Sección 5.3 describe las bases a evaluar.

5.1. Redes Neuronales

Para evaluar el estado del arte de módulos y redes, se propone la evaluación de múltiples redes de alta relevancia en la competencia COCO. Las redes escogidas son; Scaled YOLOv4, YOLOR y Swin. Las redes Scaled YOLOv4 y YOLOR son una de las redes One-Stage más precisas en esta competencia al momento de escribir esta tesis, al mismo tiempo que mantienen una velocidad más alta que la mayoría de otras redes modernas. Ambas están basadas en la red YOLOv4, y cada una incorpora mejoras en los módulos Darknet, que incluyen módulos CSP, uso de focal loss, entre otras. En el caso de YOLOR esta fue pre-entrenada para COCO y a su vez incorpora componentes de conocimiento implícito [94], para mejorar su capacidad de inferencia. La red Swin fue por un tiempo la red más precisa en COCO por su uso de módulos transformers. También fue la base para múltiples trabajos posteriores tales como DyHead [101], DINO [88] y SwinV2 [90].

Para contrastar el rendimiento entre estas redes en el SOTA y las redes convencionales se evaluará en igualdad de condiciones a las redes Faster-RCNN y Mask-RCNN, usando el backbone Resnet101-FPN. Para poder hacer comparables distintas arquitecturas se optó por considerar el número de parámetros como métrica para nivelar las capacidades de las redes y que las diferencias en resultados fueran basadas en el tipo de operaciones y no en el tamaño de estas. Por ello se evaluaron redes con alrededor de 100 Millones de parámetros. En el caso de SWIN esto implica usar la versión Small de esta red, mientras que las redes Scaled YOLOv4 y YOLOR usaran sus versiones P6 y E6 respectivamente.

5.2. Metodologías de entrenamiento

Dado el potencial de las metodologías de aumento de datos descritas en la Sección 4.5 se decidió evaluar tres metodologías de aumento de datos. Las metodologías de aumento de datos permiten mejorar la capacidad de inferencia de cualquier red independiente de su arquitectura o aplicación, y sin incurrir en costo computacional adicional durante la inferencia. Se optó por el uso de algoritmos de trasplante de píxeles, como los discutidos en la Sección 4.5.2. En particular se usan los algoritmos de Mosaic y Copy-Paste, siendo estos los más relevantes en la actualidad. Otro método de aumento de datos que se evaluará fue el de Self-Train, descrito en las Sección 4.5.3.

Los tres algoritmos fueron implementados desde cero en Python. Estos fueron desarrollados para funcionar con etiquetas de tipo COCO, para facilitar su uso y flexibilidad en múltiples escenarios. Todo el código para esto se dejó en un repositorio público en github denominado AggressiveDataAugmentation[2].

[2] Hacer click en el nombre del repositorio.

5.3. Bases de datos

Para comparar las distintas metodologías y redes propuestas se propone la evaluación de estas en dos base de datos, MinneApple [42] y Cherry CO. Estas bases de datos se componen de imágenes de manzanas y cerezas, respectivamente, capturadas en instalaciones agrícolas, por lo que presentan características realistas y típicas de este tipo de ambientes.

- MinneApple: MinneApple fue producido por el Robotic Sensor Network Laboratories (RSN) de la Universidad de Minnesota, USA [42]. Está compuesto por 1000 imágenes de manzanos. Dichas imágenes fueron tomadas a partir de una grabación de un celular Samgsung Galaxy S4 de las filas de manzanos, seleccionando y etiquetando solo un subconjunto de las grabaciones. La base de datos muestra múltiples especies de manzanas, distintas condiciones de iluminación, distintos tamaños de manzanas, lo que ofrece una gran variabilidad de colores, tamaños y grados de oclusión. Cada imagen tiene asociada una máscara, donde están marcados los segmentos de cada fruta con un número único que permite identificar frutas individualmente (ver Fig. 1.1 d)). El conjunto de test no contiene etiquetas disponibles, en su lugar existe una competencia en CodaLab, donde se pueden evaluar predicciones sobre este conjunto y obtener puntajes en distintas métricas.
- Cherry CO: Este es una base de datos de cerezas generado para esta tesis. Está compuesto por 3006 imágenes etiquetadas de cerezos, las cuales fueron adquiridas con una cámara fotográfica en un centro de estudios frutícolas. Cada imagen tiene asociada las coordenadas, los contornos y los grados de madurez de cada cereza individual. El Capítulo 6 contiene una descripción detallada de la adquisición de imágenes y la construcción de la base de datos.

Estas bases de datos de manzanas y cerezas, respectivamente, se componen de imágenes tomadas en instalaciones agrícolas a la intemperie, con distintos niveles de iluminación, oclusión, número de frutas, grados de madurez y posición del árbol con respecto a la cámara. Es decir, son bases de datos desafiantes, que engloban las dificultades asociadas con detección de fruta en ambientes agrícolas. Ambas bases de datos contienen etiquetas para Segmentación y Detección de Objetos.

Tabla 5.1: Información de bases de datos de frutas

Nombre	Fruta	Número de imágenes	Resolución	Partición Train	Partición val	Partición Test
MinneApple	Manzana	1000	1280×720	536	133	331
Cherry CO	Cereza	3000	1328×1328	1800	600	600

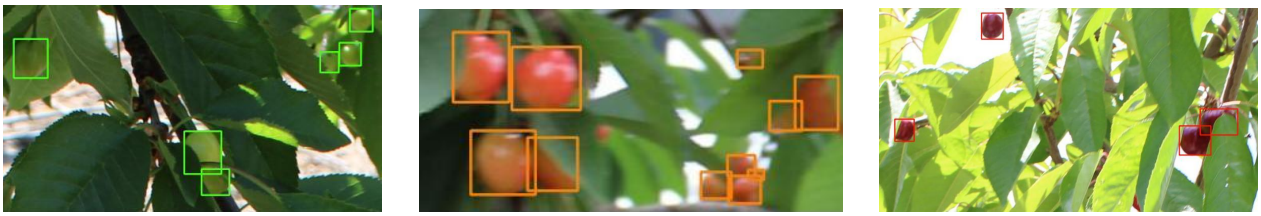
Capítulo 6

Diseño y Construcción de Base de Datos Cerezas

Dada la escasez y baja variedad de bases de datos de frutas se desarrolló una base de datos de cerezas para entrenar y evaluar las distintas redes y metodologías propuestas. Esta base de datos se construyó como una colaboración entre la Universidad de Chile y la Universidad de O'Higgins, acuñada como Cherry CO (Cherry Chile-O'Higgins). Este es una base de datos pública de cerezas, compuesto por 3006 imágenes etiquetadas. Este capítulo describe la adquisición, anotación y características de Cherry CO.

6.1. Adquisición de datos

Las imágenes utilizadas para este conjunto de datos fueron tomadas del Centro de Estudios Frutícolas Avanzados (CEAF), Rengo, Chile, que apoyó esta investigación. El periodo de adquisición fue durante los días 8, 14 y 24 de noviembre de 2021. Este periodo comprende el proceso de maduración de las cerezas cuando el fruto ya está formado. Los grados de madurez capturados se pueden observar en la Figura 6.1. Las imágenes se tomaron entre las 10 a. m. y las 4 p. m. durante los días soleados y nublados, lo que genera diferentes grados de iluminación en el conjunto de datos. Las fotos fueron tomadas con una cámara comercial de mano (Canon rebel Ti7) desde una distancia de entre 1 y 1,3 metros del tronco del árbol, y desde distintos ángulos y alturas. Las cerezas pueden crecer en cualquier rama del árbol, y en consecuencia estas pueden aparecer más cerca o más lejos de la cámara que el tronco del árbol. Naturalmente la existencia de hojas, ramas y otras cerezas causa que la mayoría de las cerezas visibles están parcialmente ocluidas, dejando ver sólo una sección de estas. La cámara se configuró con ajustes automáticos de enfoque, lo que significa que los ajustes de velocidad de obturación, apertura y enfoque se configurarán automáticamente en cada toma, para enfocar específicamente una sección de la imagen, comúnmente algún plano con cerezas. Naturalmente otras secciones de la imagen no tienen el mismo grado de enfoque. Todo esto generó una gran variedad en términos de color, iluminación, oclusión, tamaño, nitidez y ángulo del cual se tomaron las fotos.



(a) (b) (c)

Figura 6.1: Ejemplos de madurez de cerezas. a) Verde. b) Inmadura. c) Madura.

6.2. Anotación de imágenes

Las cerezas en este conjunto de datos incluyen ejemplos desde la etapa de ovarios verdes hasta cerezas completamente maduras. Con el fin de distinguir los diferentes grados de madurez, se establecieron tres categorías:

- Verde: Esta categoría refiere a las cerezas de color verde amarillento (ver Fig. 6.1 a), que es la primera etapa de maduración presente en la base de datos.
- Inmadura: Esta categoría refiere a las cerezas que están en un periodo de transición entre frutas verdes a frutas maduras. Se caracterizan por tener un color naranja, pero pueden tener secciones de la fruta de un color verde o rojo.
- Madura: Esta categoría contiene a todas las frutas de color rojo (6.1 c), indicando un grado de madurez suficiente para ser consumidas. Estas frutas incluyen un rojo claro y un rojo oscuro.

Los cerezos a menudo se organizan en varias filas, por lo que una foto a menudo incluye árboles en segundo plano. Para varias aplicaciones agrícolas, las frutas en segundo plano no son un objetivo deseable, ya que eso podría generar problemas de doble conteo o no ser de interés práctico. Por lo tanto, se generaron sub-clasificaciones, en las categorías de segundo plano (SP) o de primer plano. Finalmente, se asignó una categoría a las cerezas que, por cualquier motivo, no eran comestibles (frutos picoteados, frutos caídos, etc.). Con esto en mente se definieron en total 6 clases distintas. Comenzando con cereza madura, inmadura y verde, para cerezas en primer plano, con los grados de madurez correspondientes. En segundo lugar, se definió dos categorías de frutas en segundo plano, SP madura y SP inmadura. Cabe mencionar que las cerezas verdes en segundo plano se categorizaron como SP inmadura, debido al escaso número de muestras de cerezas inmadura y verde en segundo plano y dado que estas son difíciles de distinguir. Finalmente, la fruta no comestible se clasificó como dañada. Cada cereza fue etiquetada con anotaciones de segmentación como se muestra en la Fig. 6.2.

(a) (b)

Figura 6.2: Ejemplos de frutas anotadas. a) Imagen original. b) Imagen segmentada.

En total se adquirieron 5500 imágenes, de las cuales 501 fueron seleccionadas al azar para el proceso de etiquetado, utilizando la plataforma www.labelbox.com. En cada imagen se anotaron las 6 clases de objetos con etiquetas de segmentación individual para cada fruta, teniendo cuidado de anotar solo las secciones visibles de las cerezas. El proceso de etiquetado duró aproximadamente 3 meses y fue realizado por 3 personas, dos anotadores contratados para seguir las guías de etiquetado descritas anteriormente, así como el r. Un mes después yo realice el proceso de revisión de etiquetas, el cual consiste en verificar que cada anotación estuviera debidamente realizada, en términos de forma y clase, además de etiquetar cualquier cereza faltante, de esta forma asegurando estándares de calidad y uniformidad en toda la base de datos.

6.3. Post-Procesamiento

Dado que las imágenes etiquetadas eran $2,656 \times 3,984$ $[pixels^2]]$, lo cual es demasiado grande para la mayoría de las redes neuronales profundas, cada imagen se dividió en una cuadrícula de 2×3, generando 6 imágenes etiquetadas de $1,328 \times 1,328$, contabilizando para 3,006 imágenes etiquetadas. Finalmente, se filtraron todas las anotaciones con un área segmentada inferior a 270 $[pixels^2]$, para quitar objetos demasiado pequeños y generar un criterio uniforme. En total, Cherry CO tiene más de 15,000 anotaciones, en 3,006 imágenes.

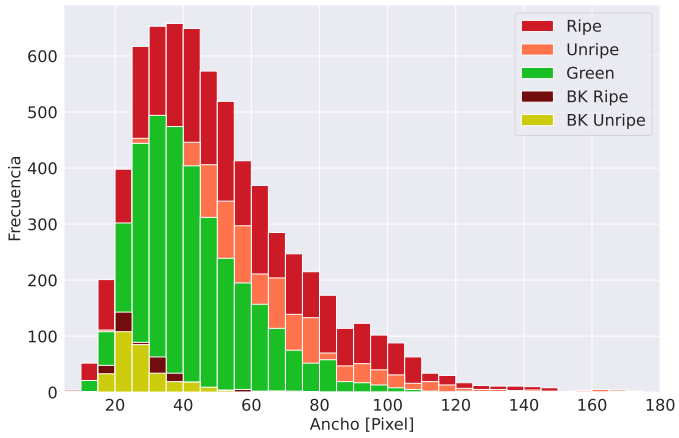
6.4. Análisis estadístico

La Tabla 6.1 muestra una descripción de las diferentes clases. La clase madura es la más numerosa, seguida de las clases inmadura y verde, lo que muestra un desequilibrio entre las tres clases de primer plano. Estas tres clases componen más del 95 % de las anotaciones, lo que indica que las dos clases de segundo plano están subrepresentadas, lo que dificulta la distinción precisa de dichas clases.

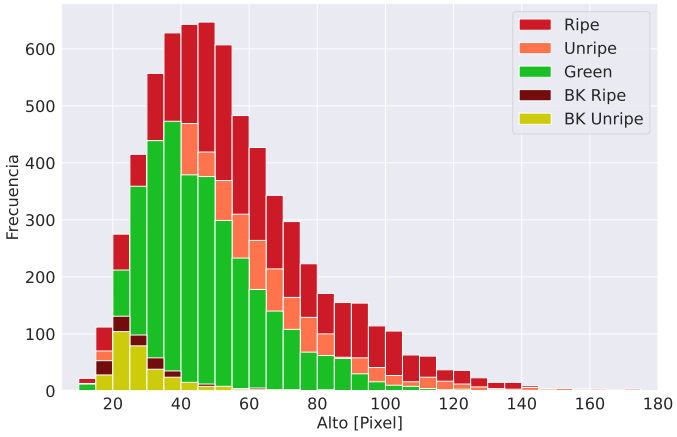
Las cerezas en este conjunto de datos se muestran como objetos pequeños, cuya área segmentada representa menos de 0.12 % del área total de la imagen. Sin embargo, la mayoría de las cerezas son lo suficientemente grande como para identificarse claramente, ya que todas las BB tienen un lado de al menos 10 píxeles, y casi todas tienen dimensiones de BB de más de 20×20, como se muestra en la Fig. 6.3.

Tabla 6.1: Comparación de anotaciones por clase. Se muestran el número de anotaciones totales, el área segmentada promedio y las dimensiones de BB's.

	Anotaciones totales	Promedio de anotaciones por imagen	Ancho promedio BB [p]	Altura promedio BB [p]	Area segmentada promedio [p^2]	Porcentaje del area en la imagen [%]
Madura	6,684	2.22	50.96	55.14	2101.38	0.12
Inmadura	4,211	1.40	48.13	50.53	1856.41	0.11
Verde	3,524	1.17	42.14	45.67	1482.11	0.08
S.P. Madura	420	0.14	28.02	28.33	590.27	0.03
S.P. Inmadura	312	0.10	27.44	28.67	601.84	0.03
Total	15,151	5.04	47.00	50.37	1816.49	0.10



(a)



(b)

Figura 6.3: Distribución de ancho y alto de BB, separados por clase. a) Histograma de Ancho. b) Histograma de Altura.

6.5. Configuraciones de Cherry CO

Las aplicaciones de detección de frutas tienen diferentes objetivos, por ejemplo, un sistema de recolección debe detectar y recolectar solo frutas maduras [10, 11, 38]. Por otro lado, monitorear la madurez de la fruta requiere distinguir diferentes grados de madurez [55, 67, 77]. Por ende, los algoritmos de entrenamiento y evaluación para cada aplicación requieren de

distintos objetivos. Por ello se definieron tres configuraciones para Cherry CO:

- Maduraz: Esta configuración consiste en las cerezas maduras, y evalúa la capacidad de las redes para diferenciar la fruta madura del resto. Está compuesto por una única clase, que incluye cerezas maduraz en el primer y segundo plano.
- Madurez: Este conjunto de datos considera tres niveles de madurez como tres clases diferentes, por lo tanto, evalúa la capacidad de detectar y distinguir los tres grados de madurez. Las clases de segundo plano se juntaron con sus respectivas clases en primer plano.
- Combinada: Esta configuración considera todas las cerezas visibles como una sola clase, por ende, evalúa la capacidad de detección global de cerezas, sin distinción de madurez.

Todas estas configuraciones se dividieron en conjuntos para entrenamiento, validación y prueba, en una proporción de 3:1:1, respectivamente.

Capítulo 7

Estudio comparativo: Metodología, Resultados y Discusión

Como se mencionó en la sección 5, el uso de metodologías modernas de detección para aplicaciones agrícolas es limitado, y generalmente se prefieren acercamientos convencionales. Con el propósito de realizar un análisis comparativo entre redes y métodos en igualdad de condiciones, se realizaron estudios comparativos para determinar el efecto de estos. Este Capítulo se divide en dos secciones, la primera es 7.1 Experimentos, que describe y justifica los experimentos realizados para evaluar las redes y metodologías propuestas. La sección 7.2 presenta los resultados de los experimentos y entrega un análisis de los resultados obtenidos.

7.1. Experimentos

Los experimentos propuestos buscan una comparación completa e imparcial de las capacidades de las redes y métodos de entrenamiento evaluados. Todas las redes descritas en 5.1 fueron entrenadas en los mismos conjuntos de entrenamiento correspondientes a los distintos métodos de aumento, de esta forma se realizaron experimentos comparables entre las distintas redes en cada uno de las bases de datos. De igual modo, se evaluaron las redes resultantes de los entrenamientos en los conjuntos de validación y test, para obtener métricas del desempeño de estas redes. Los métodos de aumento a evaluar se definieron como:

- Base: Cada entrenamiento emplea aumento de datos convencionales basados en transformaciones de escala, de traslación, modificaciones de brillo, contraste y color.
- Self-train: Estos entrenamientos consisten en que cada red se usa para producir pseudo-etiquetas de un conjunto de datos no etiquetados, para luego entrenar desde cero el mismo tipo de red usando este conjunto de datos, combinado con el conjunto de entrenamiento original. Es preferible usar redes altamente precisas para generarlas pseudo-etiquetas, de modo que para esta tesis se emplea redes entrenadas con otros métodos de aumento de datos, como copy-paste, para generar las pseudo-etiquetas, en lugar de usar el caso base. A modo de ejemplo, se utilizó el método copy-paste para entrenar la red YOLOR, para luego usar esa red y etiquetar un conjunto de datos no etiquetado. Un segundo entrenamiento se realiza desde 0 con una red YOLOR, utilizando el conjunto de datos extendido
- Mosaic: Este método de aumento genera una nueva imagen combinando 4 esquinas de imágenes tomadas al azar del conjunto de entrenamiento (ver Fig. 4.5 e). Se realizaron

dos modificaciones al método original descrito por [32]. La primera es que en el método original las esquinas de las imágenes correspondientes no están asociadas la misma esquina en la imagen resultante. Esto se modificó y se mantuvo la posición de las esquinas, es decir, la esquina superior derecha de una imagen donante se utilizó como la esquina superior derecha en la imagen resultante. Esto se modificó para poder asegurar imágenes realistas para este contexto, donde el cielo esté en las esquinas superiores y el suelo en las inferiores. Otro cambio importante es que los recortes de las esquinas se realizan en ubicaciones que no corten por la mitad algún objeto, dado que objetos pequeños podrían ser difíciles de distinguir una vez fueran recortados.

- Copy-Paste: Este método, descrito en [116] añade recortes de imágenes a una imagen base, para generar una nueva (ver Fig. 4.5 d), donde el número de objetos a copiar y pegar es un parámetro. En este caso se implementó una versión de copy-paste que produce un número al azar dentro de cierto rango para definir el número de objetos a copiar. El numero de frutas a pegar en cada imagen generada es eelgido al azar en el rango 1-15 para MinnaApple y 1-5 en Cherry CO.

Cabe mencionar que no se evaluará el método de Self-train en MinneApple al no existir una base de datos de características similares disponible y con el tamaño correcto.

7.1.1. Recursos

El codigo y pesos pre-entrenados utilizados provienen de los repositorios públicos de los respectivos autores para las redes scaled YOLOv4, YOLOR y SWIN. Los frameworks de las redes Scaled YOLOv4 y YOLOR están definidas en pytorch, donde se utilizó la versión 1.10.0. Mientras que tanto SWIN como las redes Faster-RCNN y Mask-RCNN están definidos en mmdetection, un frameworks basado en pytorch que ofrece distintas arquitecturas pre-hechas y una serie de herramientas de entrenamiento, validación y análisis.

7.1.2. Condiciones de evaluación

Esta sección describe las distintas condiciones de entrenamiento en las distintas redes. También se justifica las distintas decisiones concernientes al entrenamiento de las redes, en función de optimizar la precisión de las redes y mantener condiciones homogéneas entren los distintos experimentos.

7.1.2.1. Hiper-parámetros de Redes

Se realizaron múltiples experimentos para evaluar los hiper-parámetros de las redes. En primer lugar, se determinó el número ideal de épocas de entrenamiento. Para las redes YOLOR y YOLOv4 se optó por realizar 125 épocas de entrenamiento para el entrenamiento en los casos base (sin aumento) y 90 épocas de entrenamiento en entrenamientos con aumento de datos. Cada entrenamiento se realizó con el optimizado Adam usando parámetros $\beta_1 = 0,937$, $\beta_2 = 0,999$. Cabe mencionar que se utilizó una política de reducción de learning rate durante el entrenamiento llamada One-Cycle scheduling [122], que reduce el learning desde un valor $l_{r0} = 0,001$ en la primera época hasta un valor $lr = 0,001 l_{rf2}$, en la última. En este caso se utilizó $l_{rf2} = 0,01$. Este tipo de acercamiento permite hacer ajustes mas precisos al final del entrenamiento. Finalmente se optimizó el batch-size de cada red dentro de los límites del hardware empleado, en este caso una Nvidia Tesla-V100. Para las redes YOLOR y YOLOv4

se utilizó un batch-size de 13 y 12, respectivamente (para el entrenamiento de Cherry CO). La Tabla 7.1 resume el resto de hiper-parametros utilizados.

Tabla 7.1: Hiper-parámetros usados en los entrenamientos de las redes.

Red	Batch size	Learning rate	Weigth decay	Optimizador	Reduccion de learning rate	Minimo LR	Epocas caso base	Epocas con aumento
Faster-RCNN	10	0,0003	0,0001	Adam	Step decay	3,00E-07	50	39
Mask-RCNN	10	0,0003	0,0001	Adam	Step decay	3,00E-07	50	39
YOLOR	13	0,0010	0,0005	Adam	One-Cycle	1,00E-05	125	90
YOLOv4	12	0,0010	0,0005	Adam	One-Cycle	1,00E-05	125	90
Swin	5	0,0003	0,0001	Adam	Step decay	3,00E-07	50	39

Las redes Swin, Mask-RCNN y Faster-RCNN son redes que emplean una arquitectura Two-Stage y se entrenan en el mismo framework (mmdetection), por lo que se optó por emplear los mismos parámetros de entrenamiento en estas redes en la mayoría de los aspectos, los cuales están descritos en la Tabla 7.1. En las tres redes se utilizó una política de reducción de learning rate conocida como decaimiento por escalones, donde se disminuye el learning rate l_r en l_{rf} en distintas épocas. En este caso se redujo en las épocas [9,24,39] en una magnitud de $l_{rf} = 0,1$.

Es importante resaltar que, aunque los entrenamientos realizados con las redes Swin, Mask-RCNN y Faster-RCNN tienen menos épocas que las realizados por las redes YOLO, las redes más precisas se obtuvieron en épocas en torno a la mitad del entrenamiento en la mayoría de los casos, indicando que este tipo de redes requieren de menos épocas de entrenamiento.

7.1.2.2. Evaluación de resolución

Para mantener aspectos visuales homogéneos entre distintos entrenamientos se empezó por definir la resolución sobre la cual todas las redes se evaluaran en la base de datos correspondiente. Para esto se entrenó y evaluó los resultados de distintas resoluciones en una sola red, buscando optimizar AP@0.5 en el proceso. Por ejemplo, en el caso de Cherry CO, se evaluó su configuración Maduraz entrenando la red YOLOR para distintas resoluciones. Al mismo tiempo se maximiza el batch-size en cada resolución.

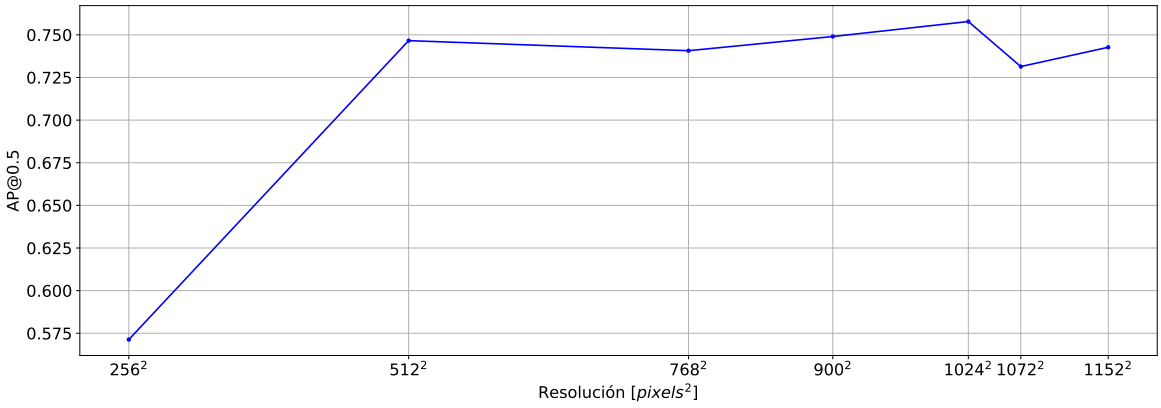


Figura 7.1: Gráfico de Resolución vs Precisión.

Se concluyó que se maximiza la precisión con una resolución de 1024×1024, aun cuando la resolución original de las imágenes en esta base de datos es de 1328×1328. Esto se debe a que en un mismo hardware el aumento de la resolución implica una disminución del batch-size, el cual es un parámetro de gran importancia para que las redes aprendan.

El mismo análisis fue realizado en la base de datos MinneApple y se concluyó que la mejor resolución para este conjunto es de 1280×720. En ambas bases de datos la resolución se mantendrá constante para todas las redes y se optimizará individualmente el batch-size en cada caso.

7.1.2.3. Evaluación de aumento de datos

Para mantener en igualdad de condiciones los distintos algoritmos de aumento de datos se impuso que todos deben generar la misma cantidad de datos, para que los entrenamientos sean comparables. Para elegir el tamaño de aumento se evaluó con la red Scaled YOLOv4 cuanto variaba la precisión para distintos tamaños de la base de datos de entrenamiento en MinneApple. En la Fig. 7.2 se observa la precisión de la red como función del tamaño de la base de datos, como múltiplo del tamaño original. De modo que la base de datos original es equivalente a 1, y se evaluó hasta un tamaño de entrenamiento equivalente a 4 veces el tamaño original, es decir, que se crearon y agregaron imágenes equivalentes a tres veces el tamaño de la base de datos de entrenamiento. Como se observa en la Fig. 7.2 al aumentar el tamaño del conjunto de entrenamiento mejora la precisión. Considerando que el aumento no mejora significativamente posterior a n=3, se escogió este tamaño para todos los algoritmos, dado que tamaños más altos también aumentan el tiempo de entrenamiento.

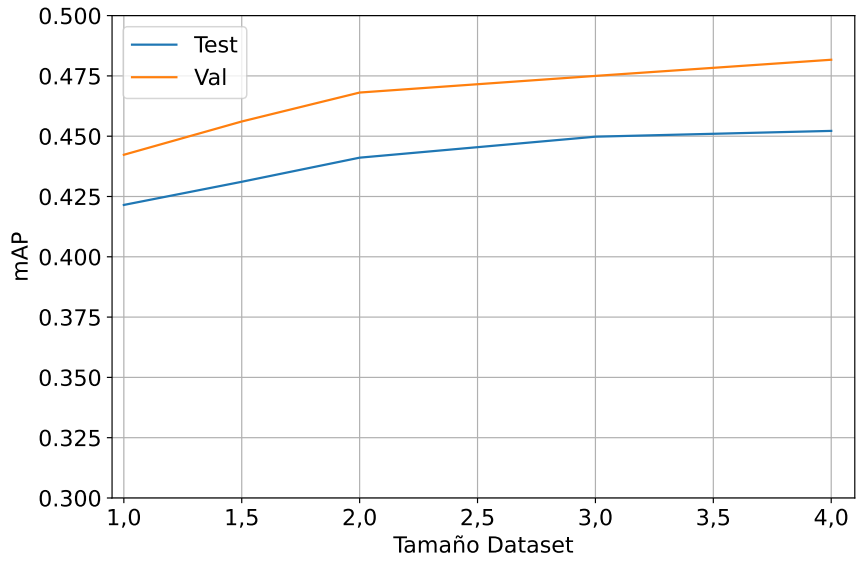


Figura 7.2: Gráfico de Tamaño del conjunto de entrenamiento vs Precisión en MinneApple.

7.2. Resultados y Análisis

Esta sección presenta los experimentos realizados, mostrando los resultados de las redes y métodos evaluados. Se realizaron pruebas en ambas bases de datos, MinneApple y Cherry CO, en las cuales se evaluaron las métricas AP@0.5 y mAP tanto para el conjunto de validación como de prueba, siendo esta última métrica la más significativa para evaluar el rendimiento de las redes. Adicionalmente también se toma en consideración la velocidad de las redes.

La sección 7.2.1 presenta los resultados de los distintos experimentos en la base de datos Cherry CO. Dado que este conjunto contiene 3 configuraciones, los resultados están divididos en las configuraciones Maduraz, Combinada y Madurez. Dado que esta es una base de datos aún sin publicar, no hay otros trabajos con los que compararlos, por lo que solo se consideran los resultados de las redes estudiadas en esta tesis. Posteriormente se evalúan los distintos métodos y se describen los resultados en función de las distintas métricas evaluadas, para luego hacer un análisis de los resultados en las distintas configuraciones.

La sección 7.2.2 expone los resultados de los métodos y redes evaluadas en MinneApple. Se evalúan las métricas de Precisión y Recuento evaluadas en Cherry CO y se comparan con los resultados de otros autores en la competencia MinneApple. De igual modo se hace un análisis de los distintos métodos y redes, señalando la efectividad de estos.

La sección 7.3 hace un análisis de las fortalezas y debilidades de las redes. Estas se vinculan con las aplicaciones en las que mejor se desempeñan las redes y métodos estudiados.

7.2.1. Cherry CO

7.2.1.1. Resultados de experimentos

Las Tablas 7.2-7.4 muestran los resultados de los experimentos realizados en Cherry CO, donde los resultados que se observan son aquellos con el mAP VAL mas alto. A partir de estos experimentos se puede observar ciertas cualidades de las redes y los métodos de aumento de datos:

- Buen rendimiento de redes modernas: En los diferentes experimentos de aumento realizados en la configuración Madurez, Swin y YOLOR muestran las mejores puntuaciones de la prueba mAP cuando se comparan los distintos métodos de aumento, mientras que YOLOR y YOLOv4 obtuvieron los mejores puntajes de Recuento. Las redes convencionales tienden a tener menor precisión. Por ejemplo, en la configuración Ripe, Mask-RCNN tiene un promedio de 3,8 % puntos menos que Swin en mAP Test, mientras que Faster-RCNN tiene 6,8 % menos. Estos resultados también se pueden observar en el conjunto de datos de validación.

Tabla 7.2: Comparación de métodos y redes en Cherry CO en configuración Madurez.

Red	Aumento	AP@0.5 Val	AP@0.5 Test	mAP Val	mAP Test	Recuento Test	F1-Score
Faster-RCNN	Base	75.4	84.4	47.3	53.2	63.8	58.0
Faster-RCNN	Self-train	75.1	84.0	47.5	52.6	60.4	56.2
Faster-RCNN	Mosaic	73.8	83.4	46.9	53.0	64.2	58.1
Faster-RCNN	Copy-Paste	76.0	84.4	49.2	54.7	65.6	59.7
Mask-RCNN	Base	66.6	83.1	40.3	52.6	63.0	57.3
Mask-RCNN	Self-train	71.8	82.4	45.4	53.8	63.5	58.2
Mask-RCNN	Mosaic	74.1	83.0	47.1	53.1	64.2	58.1
Mask-RCNN	Copy-Paste	75.7	83.5	49.1	54.9	65.8	59.9
YOLOv4	Base	68.8	80.1	44.0	52.0	70.3	59.8
YOLOv4	Self-train	70.3	80.7	46.7	53.9	**71.0**	61.3
YOLOv4	Mosaic	71.4	81.3	49.4	55.9	70.7	62.4
YOLOv4	Copy-Paste	71.3	81.1	48.5	56.1	70.9	62.6
YOLOR	Base	72.4	81.5	48.4	55.4	70.0	61.9
YOLOR	Self-train	71.3	80.7	48.4	55.2	68.8	61.3
YOLOR	Mosaic	74.5	83.0	50.2	57.1	70.5	63.1
YOLOR	Copy-Paste	71.3	82.8	51.3	57.7	70.8	**63.6**
Swin	Base	73.3	**87.3**	45.9	55.4	66.6	60.5
Swin	Self-train	78.2	86.5	50.2	55.9	66.3	60.7
Swin	Mosaic	77.6	86.3	49.6	55.9	66.2	60.6
Swin	Copy-Paste	**79.5**	86.3	**52.0**	**57.9**	67.8	62.5

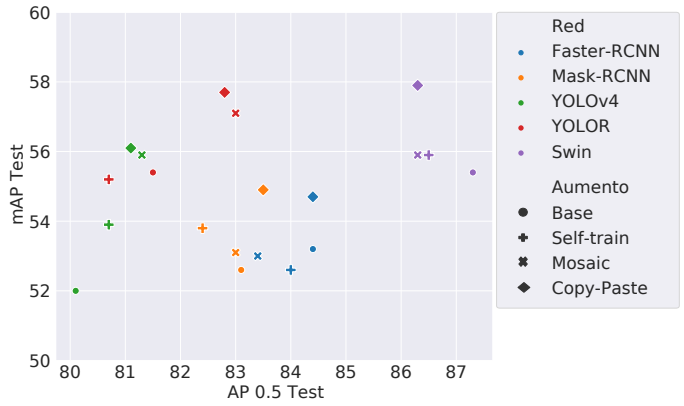
Tabla 7.3: Comparación de métodos y redes en Cherry CO, configuracion Maduraz.

Red	Aumento	AP@0.5 Val	AP@0.5 Test	mAP Val	mAP Test	Recuento Test	F1-Score
Faster-RCNN	Base	75.7	77.0	46.2	46.6	61.0	52.8
Faster-RCNN	Self-train	75.0	75.0	44.4	45.0	58.9	51.0
Faster-RCNN	Mosaic	75.4	74.8	46.0	45.4	58.5	51.1
Faster-RCNN	Copy-Paste	75.4	75.9	46.8	46.0	60.4	52.2
Mask-RCNN	Base	78.2	78.3	47.5	48.2	61.6	54.1
Mask-RCNN	Self-train	79.1	78.5	49.3	48.8	60.6	54.1
Mask-RCNN	Mosaic	79.5	79.1	49.3	48.7	61.3	54.3
Mask-RCNN	Copy-Paste	77.6	80.1	49.6	50.9	62.8	56.2
YOLOv4	Base	71.8	68.3	45.1	43.9	65.9	52.7
YOLOv4	Self-train	75.5	67.9	45.8	45.5	67.2	54.3
YOLOv4	Mosaic	78.1	75.7	51.6	51.0	68.8	58.6
YOLOv4	Copy-Paste	75.5	73.1	49.6	49.3	67.7	57.1
YOLOR	Base	77.9	77.0	49.5	48.8	64.2	55.5
YOLOR	Self-train	79.7	73.7	50.5	50.0	63.9	56.1
YOLOR	Mosaic	79.2	76.5	51.0	52.1	67.3	58.7
YOLOR	Copy-Paste	76.1	77.7	**54.4**	53.2	**69.6**	**60.3**
Swin	Base	76.9	79.9	46.6	50.0	62.1	55.4
Swin	Self-train	82.3	81.4	53.6	53.1	66.9	59.2
Swin	Mosaic	**82.8**	81.9	53.6	53.0	63.5	57.8
Swin	Copy-Paste	82.7	**83.0**	53.3	**54.0**	66.2	59.5

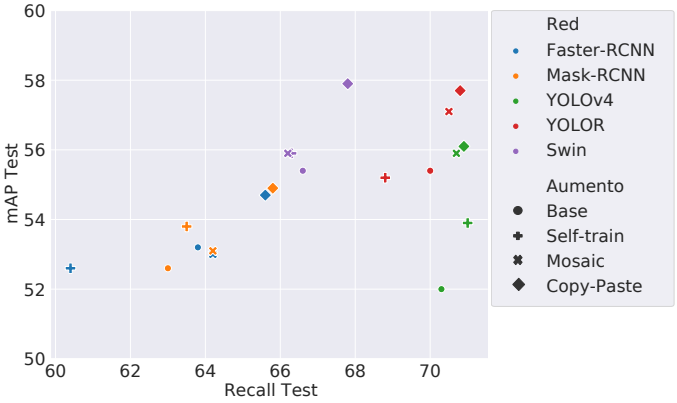
Tabla 7.4: Comparación de métodos y redes en Cherry CO, configuracion Combinada.

Red	Aumento	AP@0.5 Val	AP@0.5 Test	mAP Val	mAP Test	Recuento Test	F1-Score
Faster-RCNN	Base	87.7	87.1	57.6	57.5	64.0	60.6
Faster-RCNN	Self-train	86.9	86.0	56.9	56.7	60.1	58.4
Faster-RCNN	Mosaic	86.6	85.6	56.6	55.4	62.0	58.5
Faster-RCNN	Copy-Paste	86.5	85.8	56.9	57.0	63.2	59.9
Mask-RCNN	Base	88.0	87.6	57.5	57.2	64.0	60.4
Mask-RCNN	Self-train	87.2	87.5	57.1	56.0	63.0	59.3
Mask-RCNN	Mosaic	87.1	86.3	56.9	56.4	63.1	59.6
Mask-RCNN	Copy-Paste	88.0	86.1	57.7	57.4	64.2	60.6
YOLOv4	Base	86.2	85.4	58.6	59.5	**69.6**	64.2
YOLOv4	Self-train	85.5	83.8	58.3	57.4	65.1	61.0
YOLOv4	Mosaic	86.6	84.0	**59.8**	59.8	68.3	63.8
YOLOv4	Copy-Paste	84.9	81.3	58.5	58.1	67.8	62.6
YOLOR	Base	87.2	85.8	58.6	59.7	69.5	64.2
YOLOR	Self-train	86.1	84.1	58.4	58.5	67.4	62.6
YOLOR	Mosaic	87.4	86.4	59.4	60.0	69.1	64.2
YOLOR	Copy-Paste	87.4	85.0	**59.8**	**60.2**	**69.6**	**64.6**
Swin	Base	**89.6**	**89.4**	57.5	58.0	65.1	61.3
Swin	Self-train	89.5	88.5	57.5	57.6	65.5	61.3
Swin	Mosaic	89.2	89.3	57.6	58.2	65.4	61.6
Swin	Copy-Paste	89.3	88.7	58.6	59.3	66.2	62.6

- Diferencias entre métricas: La figura 7.3 a) muestra que las redes convencionales tienen una puntaje AP@0.5 Test más alto que YOLOR y YOLOv4, y simultáneamente tienen un puntaje mAP Test más bajo. Esto sucede cuando los detectores producen predicciones imprecisas en torno a un objeto, las cuales se consideran correctas cuando IoU=0.5, pero se consideran incorrectas en umbrales más altos. La figura 7.3 b) muestra que las redes YOLO tienen el Recuento test más alto, mientras que las redes convencionales tienen el más bajo.



(a)



(b)

Figura 7.3: Comparación de métodos y redes en la conf. Madurez. a) mAP Test vs AP@0.5 Test. b) mAP Test vs Recuento Test.

- Efectividad de los métodos de Aumento de datos: Se observó que la mayoría de las redes mejoran su puntaje mAP Test utilizando métodos de Aumento. Como se muestra en la Fig. 7.4, la mayoría de las redes aumentan su puntaje mAP en comparación a los casos base. En cada configuración las redes que obtienen el mejor puntaje son Swin o YOLOR usando Copy-Paste. Entrenamientos usando Self-train muestran mejoras de mAP en la mayoría de las redes, sin embargo este método es superado por otros y mostró un rendimiento deficiente en el conjunto Combined, al reducir mAP Test en todas las redes. No obstante, la mayoría de los experimentos muestran un aumento en mAP utilizando métodos de aumento modernos (tanto en test como en validación). Copy-Paste demostró ser el mejor método en este sentido, seguido de Mosaic.

- Recuento: Las redes con mayor recuento son YOLOR y YOLOv4, seguidas por Swin. La mayoría de las redes disminuye su recuento al usar Self-train, salvo por Swin. Similarmente a mAP, los métodos modernos de aumento de datos sirven para aumentar el recuento como se muestra en la Tabla 7.5, siendo el caso más significativo el de la red YOLOR usando Copy-Paste, el cual aumento en 5.4 % en la configuración Ripe. Sin embargo, las redes convencionales no se benefician tanto de estos métodos, especialmente Faster-RCNN.

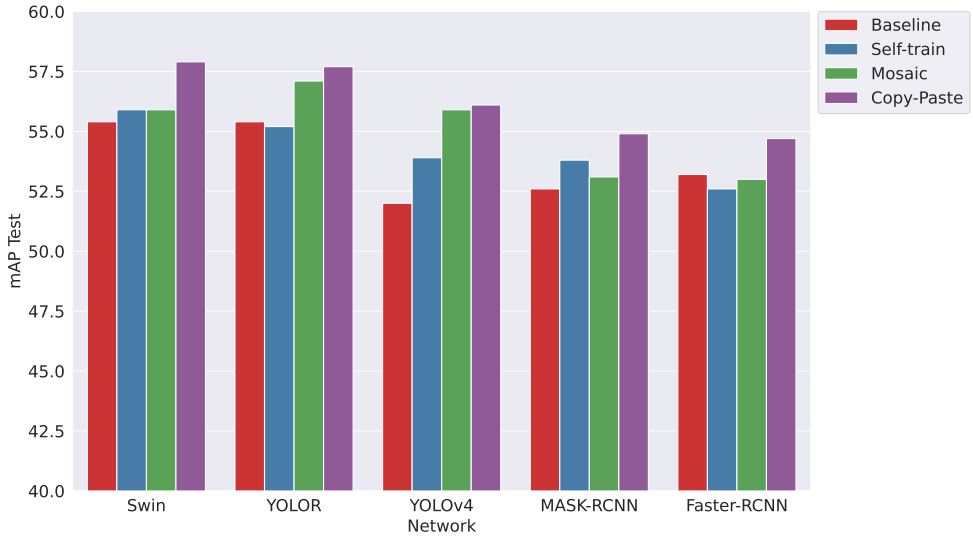


Figura 7.4: Comparación de mAP entre métodos y redes en la conf. Madurez.

Tabla 7.5: Variaciones de Recuento Test en cada método de aumento con respecto al caso base. Celdas verdes y rojas reflejan aumentos y disminuciones, respectivamente.

Configuration	Augmentation	Swin	YOLOR	YOLOv4	MASK-RCNN	Faster-RCNN
Maduraz	Self-train	3.1	4.4	5.4	2.7	-0.6
Maduraz	Mosaic	3.0	3.3	7.1	0.5	-1.2
Maduraz	Copy-Paste	4.0	1.2	1.6	0.6	-1.6
Madurez	Self-train	-0.3	-1.2	0.7	0.5	-3.4
Madurez	Mosaic	-0.4	0.5	0.4	1.2	0.4
Madurez	Copy-Paste	1.2	0.8	0.6	2.8	1.8
Combinada	Self-train	0.4	-2.1	-4.5	0.1	-3.9
Combinada	Mosaic	0.3	-0.4	-1.3	0.0	-2.0
Combinada	Copy-Paste	1.1	0.1	-1.8	-4.5	-0.8

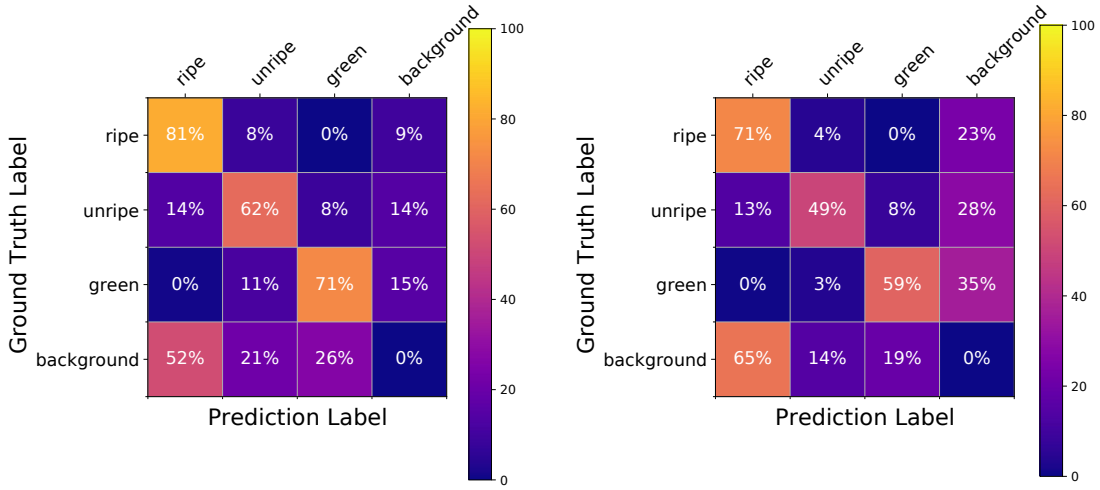
7.2.1.2. Discusión de Cherry CO

Los resultados expuestos en 7.2.1.1 muestran una comparación exhaustiva entre distintas redes y métodos de entrenamiento. A partir de esto se puede apreciar que las redes más precisas en términos de AP@0.5 son Swin, Mask-RCNN y Faster-RCNN en la mayoría de casos, superando a YOLOR y YOLOv4. Cuando se evalúa mAP (ya sea en el conjunto de prueba o validación), en cambio, se observa que las mejores redes son Swin, YOLOR y YOLOv4. Adicionalmente, las redes YOLO muestran el más alto Recuento en todas las configuraciones. Esto indica que las redes convencionales generan varias predicciones imprecisas que les permiten detectar la mayoría de las cerezas, mientras que las redes modernas producen menos predicciones, pero más precisas, logrando así un mayor mAP y Recuento. Dado que tanto los conjuntos de test como de validación exhiben el mismo comportamiento, esto indica que AP@0.5 es una métrica imprecisa por sí sola en este tipo de ambientes, donde existen varios objetos pequeños, altamente ocluidos y superpuestos.

El método Copy-Paste demostro mejorar el mAP de la mayoria de la mayoria de las redes, salvo por Faster-RCNN y YOLOv4 en la configuración Combinada. También demostró ser el método mas consistente a la hora de mejorar el Recall (ver Tabla 7.5). Por otro lado, los demás métodos presentaron mejoras en la mayoría de los experimentos donde se evaluaron las redes modernas. Sin embargo, la red Faster-RCNN disminuye su mAP cuando utiliza métodos de aumento en múltiples experimentos. Por ejemplo, en la conf. Combinada y Maduraz el mejor resultado mAP Test se observa en el caso base, y en la conf. Madurez la red Faster-RCNN presenta peores resultados usando los métodos Mosaic y Self-train que sin usarlos. Por otro lado, si se evalúan las mejoras observadas en Mask-RCNN por usar Mosaic se observa una mejora de 0.07 %, y una de 0.20 % al usar Self-Train, debido a que no en todos los experimentos se observan mejoras usando estos métodos, y de observarse son modestas. Esto indica que las redes modernas tienen más capacidad de inferencia para aprovechar los métodos de aumento de datos modernos. A partir de estos resultados se concluye que estas metodologías son deseables en todas las configuraciones y en general de gran utilidad dado que facilita que las redes aprendan a distinguir frutas y a ubicarlas, aunque son mas efectivas cuando se aplican en redes modernas.

La configuración Combinada muestra que las mejores redes en términos de mAP son YOLOR y YOLOv4, seguidas por Swin. Además estas redes tienen el mejor Recuento en las distintas metodologías de aumento. YOLOR y YOLOv4 difieren del resto en que son arquitecturas One-Stage, están basadas en operaciones eficientes (por sus Módulos CSP), las cuales les permiten tener un mayor batch-size que las demas redes manteniendo la resolucion. También utilizan la funciones de pérdida Focal Loss [115] (ver Sección 4.5.1), la cual facilita el entrenamiento de redes One-Stage. Por otro lado, Swin también presente buenos resultados en términos de mAP y logra superar a YOLOv4 cuando se evaluó usando Copy-Paste. La diferencia más significativa entre las redes Mask-RCNN y Swin (en este estudio) es el uso de módulos transformers [95], indicando su potencial para la detección de objetos pequeños.

Cuando se evalúa las configuraciones Maduraz y Madurez, se puede observar que la red más precisa es Swin, independiente del método de aumento que se utilice. Estas configuraciones evalúan la capacidad de la red de identificar cerezas, así como poder distinguir el grado de madurez de estas, a diferencia de la configuración Combinada que busca identificar frutas, sin importar su color. La red YOLOR y YOLOv4 presentaron un mAP más bajo que Swin, aunque manteniendo un superior rendimiento que las redes convencionales. Al comparar las matrices de confusión en los mejores casos de ambas redes (usando Copy-Paste), se observa que YOLOR tiene porcentajes más altos de no detecciones en todas las clases (ver Fig. 7.5 b). Esto indica que YOLOR tiende a no considerar ciertas detecciones, principalmente aquellas difíciles de identificar en el background, con el fin de minimizar Falsos Positivos. Este comportamiento difiere del mostrado en la conf. Combinada donde YOLOR detecta con más precisión que Swin, lo que indica que la distinción de madurez obliga a YOLOR a optar por una estrategia más segura. Este comportamiento le permite obtener un alto Recuento por encima que el resto de las redes.



(a) (b)

Figura 7.5: Matriz de Confusión en la configuración Madurez. a) Swin Copy-Paste. b) YOLOR Copy-Paste.

En resumen, se observó que las redes modernas presentaron consistentemente mejor mAP y Recuento que las redes convencionales. De igual modo se evidencio la utilidad de usar métodos de aumento de datos para mejorar el rendimiento de las redes y se observó que las redes modernas se benefician más de estos. Las redes YOLOR y YOLOv4 presentan el más alto Recuento en la mayoría de los experimentos, lo que a su vez también le permite a estas redes tener un alto F1-score. En cambio, la red Swin presento los mejores resultados de mAP en las configuraciones Maduraz y Madurez, indicando que es la red más apta para la inferencia de madurez.

7.2.2. MinneApple

7.2.2.1. Resultados de experimentos

En la Tabla 7.6 están los resultados de los experimentos realizados en MinneApple. Similarmente a lo realizado en los experimentos de Cherry CO se entrenaron las redes tanto en el caso base y como en versiones aumentadas. Luego fueron evaluadas en los conjuntos de validación y test, y se obtuvieron métricas de AP@0.5, mAP y Recuento tanto en el conjunto de validación como en el de Test. Cada uno de estos es el resultado del entrenamiento de la red correspondiente, de acuerdo a las especificaciones descritas en 7.1.2.

En primer lugar, se observa que las redes más precisas en términos de mAP y AP@0.5 son las redes Swin, YOLOv4 y YOLOR, siendo esta última la más precisa. De igual modo se puede observar que estas redes superan a las redes convencionales en términos de Recuento (ver Fig. 7.6). Cabe mencionar que el resultado del caso Base de YOLOR es el segundo mejor resultado en la competencia MinneApple en el momento de escribir esta Tesis y que el resto de las redes modernas se desempeñan mejor que otras redes en la competencia como lo es el tercer lugar logrado por Callum Clark [15] con una puntuación mAP de 44.1 %, usando una red Mask-RCNN con un backbone Resnext [92]. Con respecto al rendimiento de Recuento, los experimentos muestran que YOLOR, YOLOv4 y Swin tienen puntajes mucho más altos en promedio en todos los aumentos, con puntajes de 56.4 %, 55.5 y 53.9 % respectivamente.

Por otro lado, la mayoría de las redes se vieron beneficiadas por usar estrategias de aumentó

Tabla 7.6: Comparación de métodos y redes en MinneApple.

Network	Batch size	Aumento	AP@0.5 VAL	AP@0.5 Test	mAP VAL	mAP Test
Faster-RCNN	12	Base	0.8221	0.7600	0.4191	0.3993
Faster-RCNN	12	Mosaic	0.8342	0.7543	0.4199	0.4044
Faster-RCNN	12	Copy-Paste	0.8520	0.7580	0.4210	0.4053
Mask-RCNN	12	Base	0.8570	0.7500	0.4250	0.4004
Mask-RCNN	12	Mosaic	0.8588	0.7501	0.4299	0.4090
Mask-RCNN	12	Copy-Paste	0.8590	0.7560	0.4340	0.4119
YOLOV4	12	Base	0.8745	0.7550	0.4258	0.4215
YOLOV4	12	Mosaic	0.8882	0.7530	0.4536	0.4468
YOLOV4	12	Copy-Paste	0.8999	0.7970	0.4810	0.4778
YOLOR	12	Base	0.8965	0.7860	0.4548	0.4580
YOLOR	12	Mosaic	0.8901	0.7500	0.4597	0.4413
YOLOR	12	Copy-Paste	0.9014	0.8060	0.4729	0.4768
Swin Mask-RCNN	6	Base	0.8650	0.7920	0.4330	0.4222
Swin Mask-RCNN	6	Copy-paste	0.8700	0.7800	0.4420	0.4322
Swin Mask-RCNN	6	Copy-paste	0.8680	0.7793	0.4360	0.4336

de datos, especialmente Copy-Paste. La red YOLOv4 mostró el aumento más significativo con un salto de 5.7 % en mAP entre el caso entrenado con Copy-Paste y el base. El caso de base de YOLOR demostró tener la Precisión y Recuento más altos no solo en relación a las demás redes, sino que también superó a las versiones de YOLOR entrenadas con Aumento de datos.

7.2.2.2. Análisis de MinneApple

Las redes modernas se desempeñaron mejor que las redes convencionales, por sobre todo YOLOR y YOLOv4 en todas las métricas. De igual modo los métodos de aumento demostraron mejorar el rendimiento de la mayoría de las redes. Una importante excepción es que la red YOLOR obtuvo su máximo puntaje en el caso base. En este sentido es importante aclarar que el caso base de YOLOR no obtiene consistentemente los mismos resultados, aun manteniendo los mismos parámetros. Otros entrenamientos en el caso base mostraron peores resultados que la mayoría de los casos de los entrenamientos con Copy-Paste. La Fig. 7.7 muestra el resultado promedio de múltiples entrenamientos de la red YOLOR con los métodos de aumento Base y Copy-Paste, y en esta se evidencia que la red YOLOR obtiene en promedio mejores resultados usando Copy-Paste que usando Base lo que indica que ofrece resultados más consistentes, sin embargo el mejor resultado sigue siendo con el entrenamiento base.

Los resultados en esta base de datos se asemejan a los observados en Cherry CO, teniendo a YOLOR y YOLOv4 como las redes más precisas. Ambas bases de datos están compuestos por una sola clase y por ende no se distingue el color de la fruta en las etiquetas, indicando que este tipo de redes sobresale en la detección de fruta cuando no hay que distinguir su madurez. Una de las principales dificultades en esta base de datos es que no considera manzanas en el suelo o en el segundo plano como objetos válidos, aun si son visibles y por ende las redes deben detectar solo frutas que estén aun en un manzano y en primer plano. Las redes YOLO demostraron tener una tasa de Falsos Positivos mas baja que las demás redes, en gran medida por ser capaces de identificar esta diferencia. La figura 7.8 ilustra un ejemplo de este comportamiento, donde las predicciones de Swin, tienen mas falsos negativos por este motivo.

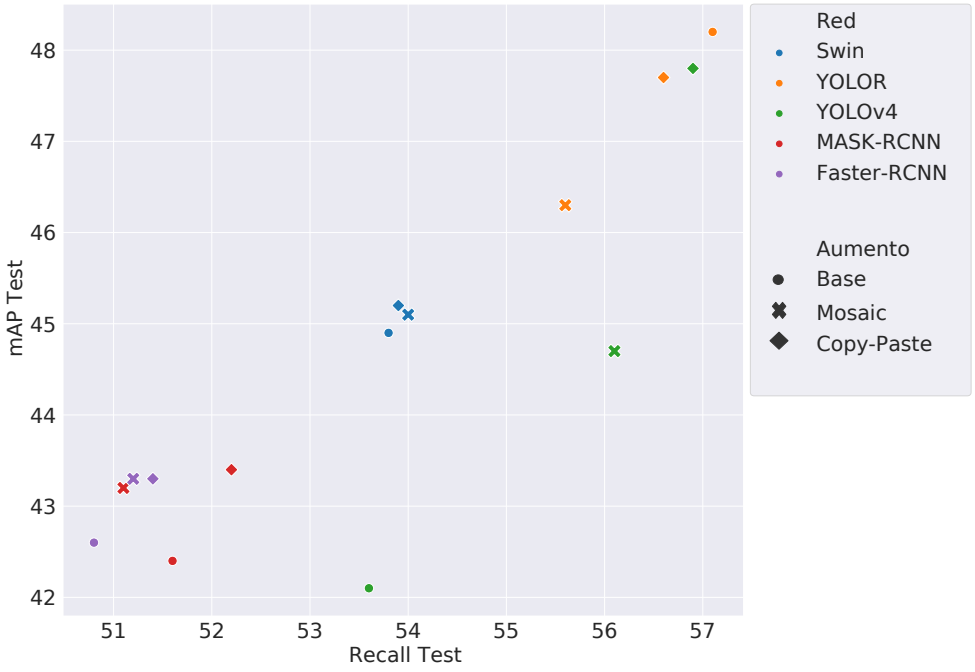


Figura 7.6: Comparación de métodos modernos de aumento de datos en MinneApple.

(a) (b)

Figura 7.8: Comparación de predicciones en MinneApple. Predicciones incorrectas en el suelo o en segundo plano están de color rojo. a) Predicciones Swin. b) Predicciones YOLOR.

El hecho de que la red Swin se desempeñe peor en esta base se puede deber a que este tipo de red requiere de un mayor número de datos para aprovechar al máximo sus módulos

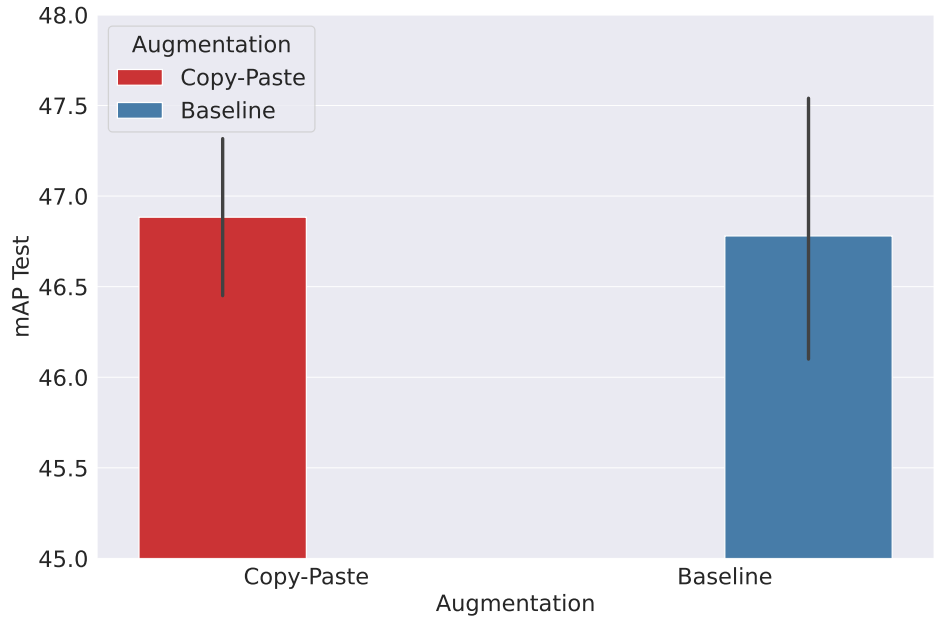


Figura 7.7: Gráficos de valor promedio de mAP test de diferentes experimentos usando la red YOLOR en MinneApple. Se realizaron 5 entrenamientos con el aumento Base y 5 entrenamientos con el aumento Copy-Paste. Las líneas representan los valores máximos y mínimos de cada método.

transformers, en relación a YOLOR que logra identificar la mayoría de las frutas, a pesar de tener una base de datos limitada. También es posible que una exploración mas completa de hiper-parámetros pueda entregar mejores resultados enfocado en las dimensiones y resoluciones de las redes pueda entregar mejores resultados. Cabe mencionar que se realizaron múltiples ajustes para optimizar los hiper-parámetros de la red en aspectos como el peso de las funciones de pérdida, entre otros.

7.2.3. Velocidad de procesamiento

Para poder comparar la velocidad de inferencia de las distintas redes se evaluó la velocidad de procesamiento de cada red en una sola imagen de 1024×1024 utilizando una GPU NVIDIA GeForce RTX 3070. Los resultados se encuentran en la Tabla 7.7, los cuales están expresados en Frames por segundo (FPS) que pueden procesar las distintas redes.

Tabla 7.7: Comparación de velocidad en FPS's de las distintas redes en la base de datos Cherry CO - Madurez (batch size=1, resolución 1024×1024).

	Swin	YOLOR	YOLOv4	MASK-RCNN	Faster-RCNN
Velocidad [FPS]	7.45	25.7	25.6	10.6	11.8

Como se observa las redes YOLOv4 y YOLOR son las rápidas, seguidas por las redes Faster-RCNN y Mask-RCNN, estas últimas tienen menos de un 50 % de la velocidad de las redes YOLO. Finalmente la red Swin demostró ser la más lenta, con menos de un tercio de la velocidad de YOLOR.

7.3. Desempeño y utilidades de Redes

A partir de los experimentos realizados en este estudio, se pudo comprobar que las redes modernas tienen una serie de ventajas frente a las redes convencionales. En primer lugar, son más precisos que los CNN convencionales, lo que les permite identificar la mayoría de las frutas, a pesar del alto grado de oclusión de las hojas y superposición entre las frutas. Además de ser más precisas, tienen un Recuento mucho mayor que las redes convencionales, lo que las hace menos propensas a sufrir efectos de sobre conteo en las aplicaciones que las utilizan. En segundo lugar, se puede apreciar una mejora entre los entrenamientos base y los entrenamientos con métodos modernos de aumento de datos en la mayoría de los experimentos, lo que indica que su uso es deseable con independencia de la red a usar.

Se puede apreciar que la red más precisa para identificar grados de madurez es Swin, por lo que se recomienda su uso para aplicaciones basadas en clasificación de estado de frutos. Por otro lado el bajo Recuento observado en las redes YOLO indica que estas son deseable para aplicaciones basadas en conteo de frutas para así evitar problemas de sobre-conteo.

Al evaluar la velocidad de las redes se observó que la red Swin es la más lenta (ver Tabla 7.7), con solo 7.54 [FPS], que puede ser una velocidad insuficiente para ciertas aplicaciones. En cambio las redes YOLOv4 y YOLOR mostraron más de tres veces esa velocidad, lo que las hace opciones preferibles para tareas que requieran generar predicciones en tiempo real. Aunque la velocidad de las redes Faster-RCNN y Mask-RCNN es más alta que Swin, estas son más lentas y menos precisas que las redes YOLO, por lo que se concluye que son peores alternativas.

Capítulo 8

Conclusiones

Esta Tesis presenta un análisis de las diferentes tendencias en estudios de detección de frutas. Para ello se hizo una revisión de la literatura sobre detección de frutas en ambientes agrícolas utilizando algoritmos de procesamiento de imágenes, con un énfasis en redes neuronales. A partir de esta revisión se expusieron las características de las bases de datos utilizadas, los principales desafíos presentes en el área de detección de frutas, y se presentaron las áreas de desarrollo de los algoritmos utilizados. Posteriormente se presenta una descripción de las aplicaciones basadas en detección de frutas, las que se categorizaron en tres tareas, que son estimación de volumen de cosecha, recolección autónoma de frutos y monitoreo de salud y madurez de frutas.

Dada la importancia de las redes neuronales en la detección de frutas también se realizó un análisis de las principales áreas de desarrollo en la detección de objetos, considerando trabajos recientes y relevantes. Se definieron 5 áreas de desarrollo en detección de objetos: Estructura de backbone, Diseño de redes, Mecanismos de atención, Modelamiento de Predicciones y Metodologías de entrenamiento.

A partir de esta revisión de las redes y metodologías usadas en detección de frutas (capítulo 3) y detección de objetos (capítulo 4) se observó ciertas diferencias en las metodologías empleadas por los estudios de detección de frutas. La falta de ciertas estrategias de entrenamiento y la ausencia de ciertos módulos son la diferencia más significativa. En este sentido se destaca la escasez de módulos transformer y la ausencia de las metodologías de entrenamiento discutidas en la sección 4.5. Dado los resultados sobresalientes de estas operaciones en la detección de objetos, es razonable asumir que su aplicación en detección de frutas resultaría en una mejora de precisión similar.

En base al análisis anterior, se propuso la realización de un análisis comparativo de redes y metodologías de entrenamiento, con el objetivo de evaluar distintas redes en bases de datos de frutas y así poder evidenciar la utilidad de estas y el impacto de ciertas metodologías de entrenamiento. Se optó por usar dos bases de datos, MinneApple y Cherry CO, al ser bases de datos desafiantes que engloban las dificultades de la detección de frutas en ambientes agrícolas. Cabe mencionar que la base de datos Cherry CO se compone de 3 configuraciones (Maduraz, Madurez y Combinada) que definen distintas tareas asociadas con la detección de cerezas. Estas bases de datos fueron usadas para evaluar las diferencias en desempeño de distintos tipos de redes, haciendo la distinción entre redes modernas y redes convencio-

nales, considerando las redes Swin [89], YOLOR [94] y YOLOv4 [33] como modernas, y las redes Faster-RCNN [27] y Mask-RCNN [28] como redes convencionales. De igual modo se evaluaron métodos modernos de aumento de datos conocidos como Copy-Paste [116], Mosaic [32] y Self-train, los cuales fueron comparados con acercamientos tradicionales basados en transformaciones geométricas y visuales.

Los experimentos demostraron que las redes modernas obtienen, consistentemente, mayor precisión mAP, recall y F1-score que las redes convencionales. Cuando se evaluó las configuraciones Maduraz y Madurez de Cherry CO, se observó que el mejor mAP (en test y validación) lo obtuvo Swin seguido por YOLOR y YOLOv4. Esto indica que la red Swin es más precisa para identificar el grado de madurez de las frutas, que el resto de las redes modernas evaluadas. En la base de datos MinneApple y en la configuración Combinada de Cherry CO la red más precisa fue YOLOR, seguida por YOLOv4 y Swin. También se demostró que las redes YOLOv4 y YOLOR tienen consistentemente mejor Recall que las demás redes incluida Swin. Con respecto a las estrategias de aumento de datos evaluadas se observó que estas aumentan la precisión y recall en la mayoría de las redes en comparación con los acercamientos convencionales de aumento de datos. Copy-Paste es particularmente consistente para mejorar el rendimiento de las redes, con respecto al entrenamiento base, especialmente si se aplica en las redes modernas.

Dada las cualidades de las redes modernas se concluye que estas son las más apropiadas para aplicaciones basadas en detección de frutas. En este sentido, se destaca la capacidad de Swin de distinguir la madurez de las cerezas, lo que la hace ideal para tareas asociadas con el monitoreo de madurez y recolección de fruta. En cambio, el alto recall de las redes YOLO, y en especial el de YOLOR, le permite realizar tareas de conteo sin incurrir en sobre-conteo. Finalmente, las redes YOLOR y YOLOv4 demostraron ser más rápidas, al ser redes ligeras, lo que les permite ser utilizadas en todo tipo de aplicaciones que requieran de una respuesta en tiempo real.

En base al análisis anterior, se concluye que las redes modernas presentan consistentemente mejores puntajes de precisión y recall que las redes convencionales en todas las bases de datos evaluadas, validando así la hipótesis sobre el mayor desempeño de las redes modernas en tareas de detección de fruta. En segundo lugar, se observaron mejoras notables en la mayoría de los experimentos empleando metodologías de entrenamiento de aumento de datos, sobre todo cuando son aplicadas en el entrenamiento de redes modernas. Copy-Paste demostró obtener los mejores resultados, seguido por Mosaic. Sin embargo, se observaron pérdidas de precisión y recall usando Self-train en algunos experimentos, principalmente aquellos asociados con la red Faster-RCNN. Por ello, se valida parcialmente la hipótesis sobre el aumento de desempeño de las redes neuronales al usar este tipo de métodos de entrenamiento.

A partir de estos resultados se puede concluir que las metodologías de aumento de datos y las redes modernas evaluadas efectivamente ofrecen un mejor desempeño en la mayoría de las tareas de detección de frutas. Cabe mencionar que se esperaba una correlación fuerte entre las posiciones relativas de las redes modernas en la competencia COCO dev-test y los resultados obtenidos en los dataset de frutas, es decir, con Swin siendo la red más precisa, seguida por YOLOR y YOLOv4. Sin embargo, este solo fue el caso en dos bases de datos. Esto se puede deber a que la red Swin requiere de un alto batch-size, lo que requiere una mayor capacidad

computacional de la que se tenía disponibilidad al momento de realizar los experimentos. También es posible que las precisiones en detección de objetos y detección de frutas no estén tan correlacionadas como se esperaba. De esta forma se valida mayoritariamente la primera hipótesis sobre la similitud del desempeño de redes neuronales en la tarea de detección de objetos y detección de frutas.

8.1. Trabajos Futuro

El trabajo desarrollado en esta tesis provee una base para futuros desarrollos en la detección de frutas. Un modo de mejorar los resultados obtenidos es realizando una optimización de hiperparametros, ya que, durante esta tesis se realizaron pocas modificaciones a las redes evaluadas con respecto a las configuraciones entregadas por los autores. Los métodos de aumento también tienen múltiples parámetros que pueden ser sometidos a optimización, por ejemplo, Copy-Paste añade a una imagen base un número de objetos aleatorio, y en una posición aleatoria pero el rango de ambas variables podría ser modificado para cada base de datos para aumentar el rendimiento de las redes. Adicionalmente, es posible combinar los distintos métodos de aumento de datos, lo que posiblemente mejoraría el rendimiento de las metodologías evaluadas. Se puede esperar mejorar el rendimiento de todas las redes al aumentar el batch-size a 32, siendo este el estándar para la mayoría de los estudios en COCO y otras tareas de procesamiento de imágenes. En este estudio no se realizaron entrenamientos de esta forma por limitaciones de hardware.

Además de lo que se evaluó en este trabajo, existen otras redes y metodologías que han recibido poco estudio en detección de frutas, tales como redes que realizan predicciones Anchorless y con predicciones en cascada, tales como Centernet [103] y HTC [29]. De igual modo redes recientes utilizan componentes que no fueron estudiados en esta tesis, como lo es la red DINO [88] que utiliza operaciones basadas en Transformers en la cabeza de predicción o la red YOLOv7 [35], la cual es la red más precisa que opera en tiempo real en la competencia COCO. Xu et al [119] presentó una metodología de entrenamiento similar a Self-training, pero de forma activa durante el entrenamiento y más sofisticada, la cual podría ser utilizada con mucho éxito en bases de datos de frutas.

Finalmente, es importante destacar la importancia del estudio y desarrollo de las aplicaciones basadas en detección de frutas, las cuales requieren de herramientas y recursos especializados en esta tarea, como lo son bases de datos de detección de frutas. Dado que esta tesis presenta una base de datos de detección y segmentación de cerezas, un siguiente paso sería el desarrollo de aplicaciones para el cosechado de cerezas basadas en detección. Por ejemplo, es posible diseñar un sistema de estimación de volumen de cosecha a partir de vídeos de hileras de árboles [18, 19, 67], utilizando una red para detectar cerezas frame a frame. Similarmente, las demás aplicaciones, descritas en 3.3, pueden ser implementadas, dado que se tiene detectores altamente precisos. Estas aplicaciones se componen de una serie de subsistemas que se pueden dividir en tareas modulares. Por ejemplo, un sistema de estimado de volumen requiere de un detector de frutas, un algoritmo que registre las frutas observadas en cada frame y un estimador que extrapole el volumen cosechable de fruta, en base al número observado de estas. Este tipo de algoritmos podría ser utilizado en otros cultivos de frutas, solo requiriendo adaptaciones menores, haciendo que separar estos sistemas en tareas modulares facilite el desarrollo de aplicaciones basadas en detección de fruta. Por ende, la

creación de recursos de acceso libre, como bases de datos, programas y equipos para abordar estas subtareas es un paso natural en esta área, lo que incentivarán el estudio y desarrollos de sistemas autónomos para cosechar frutas.

Bibliografía

[1] Bay, H., Ess, A., Tuytelaars, T., y Gool, L. V., "Speeded-up robust features (surf)," Comput. Vis. Image Underst., vol. 110, pp. 346–359, 2008.

[2] Zhang, Q., Liu, Y., Gong, C., Chen, Y., y Yu, H., "Applications of deep learning for dense scenes analysis in agriculture: A review," Sensors, vol. 20, no. 5, 2020, doi:10.3390/s20051520.

[3] Maheswari, P., Raja, P., Apolo-Apolo, O. E., y Pérez-Ruiz, M., "Intelligent fruit yield estimation for orchards using deep learning based semantic segmentation techniques-a review," Frontiers in plant science, vol. 12, p. 684328, 2021, doi:10.3389/fpls.2021.684328.

[4] Tian, H., Wang, T., Liu, Y., Qiao, X., y Li, Y., "Computer vision technology in agricultural automation —a review," Information Processing in Agriculture, vol. 7, no. 1, pp. 1–19, 2020, doi:https://doi.org/10.1016/j.inpa.2019.09.006.

[5] Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., y Zitnick, C. L., "Microsoft COCO: common objects in context," CoRR, vol. abs/1405.0312, 2014, http://arxiv.org/abs/1405.0312.

[6] Tang, Y., Dananjayan, S., Hou, C., Guo, Q., Luo, S., y He, Y., "A survey on the 5g network and its impact on agriculture: Challenges and opportunities," Computers and Electronics in Agriculture, vol. 180, p. 105895, 2021, doi:https://doi.org/10.1016/j.compag.2020.105895.

[7] Meng, H. y cheng, Y., "Research on key technologies of intelligent agriculture under 5g environment," Journal of Physics: Conference Series, vol. 1345, p. 042057, 2019, doi:10.1088/1742-6596/1345/4/042057.

[8] Saiz-Rubio, V. y Rovira-Más, F., "From smart farming towards agriculture 5.0: A review on crop data management," Agronomy, vol. 10, no. 2, 2020, doi:10.3390/agronomy10020207.

[9] Gil, G., Casagrande, D. E., Cortés, L. P., y Verschae, R., "Why the low adoption of robotics in the farms? challenges for the establishment of commercial agricultural robots," Smart Agricultural Technology, vol. 3, p. 100069, 2023, doi:https://doi.org/10.1016/j.atech.2022.100069.

[10] Arad, B., Balendonck, J., Barth, R., Ben-Shahar, O., Edan, Y., Hellström, T., Hemming, J., Kurtser, P., Ringdahl, O., Tielen, T., y Tuijl, B., "Development of a sweet pepper harvesting robot," Journal of Field Robotics, vol. 37, 2020, doi:10.1002/rob.21937.

[11] Zhang, K., Lammers, K., Chu, P., Li, Z., y Lu, R., "System design and control of an apple harvesting robot," CoRR, vol. abs/2010.11296, 2020, https://arxiv.org/abs/20

10.11296.

[12] Sane, T. U. y Sane, T. U., "Artificial intelligence and deep learning applications in crop harvesting robots -a survey," en 2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), pp. 1–6, 2021, doi:10.1109/ICECCE52056.2021.9514232.

[13] Tian, Y., Yang, G., Wang, Z., Li, E., y Liang, Z., "Detection of apple lesions in orchards based on deep learning methods of cyclegan and YOLOV3-Dense," Sensors, vol. 2019, 2019, doi:10.1155/2019/7630926.

[14] Liu, H. y Chahl, J. S., "A multispectral machine vision system for invertebrate detection on green leaves," Computers and Electronics in Agriculture, vol. 150, pp. 279–288, 2018, doi:https://doi.org/10.1016/j.compag.2018.05.002.

[15] Clark, C., "Applying Faster R-CNN and Mask R-CNN on the minneapple fruit detection challenge," BNAIC/BeneLearn 2020, p. 411, 2020.

[16] Villacrés, J. y Auat Cheein, F., "Detection and characterization of cherries: A deep learning usability case study in chile," Agronomy, vol. 10, 2020, doi:10.3390/agronomy10060835.

[17] Zheng, Y.-Y., Kong, J.-L., Jin, X.-B., Wang, X.-Y., Su, T.-L., y Zuo, M., "Cropdeep: The crop vision dataset for deep-learning-based classification and detection in precision agriculture," Sensors, vol. 19, no. 5, 2019, doi:10.3390/s19051058.

[18] Häni, N., Roy, P., y Isler, V., "Apple counting using convolutional neural networks," en 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2559–2565, 2018, doi:10.1109/IROS.2018.8594304.

[19] Liu, X., Chen, S. W., Aditya, S., Sivakumar, N., Dcunha, S., Qu, C., Taylor, C. J., Das, J., y Kumar, V. R., "Robust fruit counting: Combining deep learning, tracking, and structure from motion," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1045–1052, 2018.

[20] Pawara, P., Boshchenko, A., Schomaker, L. R., y Wiering, M. A., "Deep learning with data augmentation for fruit counting," en International Conference on Artificial Intelligence and Soft Computing, pp. 203–214, Springer, 2020.

[21] Onishi, Y., Yoshida, T., Kurita, H., Fukao, T., Arihara, H., y Iwai, A., "An automated fruit harvesting robot by using deep learning," ROBOMECH Journal, vol. 6, 2019, doi:10.1186/s40648-019-0141-2.

[22] Rosenblatt, F., "The perceptron: A probabilistic model for information storage and organization in the brain," Psychological Review, pp. 65–386, 1958.

[23] Ioffe, S. y Szegedy, C., "Batch normalization: Accelerating deep network training by reducing internal covariate shift," CoRR, vol. abs/1502.03167, 2015, http://arxiv.org/abs/1502.03167.

[24] Krizhevsky, A., Sutskever, I., y Hinton, G. E., "Imagenet classification with deep convolutional neural networks," en Advances in Neural Information Processing Systems (Pereira, F., Burges, C. J. C., Bottou, L., y Weinberger, K. Q., eds.), vol. 25, Curran Associates, Inc., 2012, https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[25] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., y Fei-Fei, L., "Imagenet: A large-scale hierarchical image database," en 2009 IEEE conference on computer vision and pattern recognition, pp. 248–255, IEEE, 2009.

[26] Kingma, D. P. y Ba, J., "Adam: A method for stochastic optimization," CoRR, vol. abs/1412.6980, 2015.

[27] Ren, S., He, K., Girshick, R., y Sun, J., "Faster R-CNN: Towards real-time object detection with region proposal networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137–1149, 2017, doi:10.1109/TPAMI.2016.2577031.

[28] He, K., Gkioxari, G., Dollár, P., y Girshick, R., "Mask R-CNN," en 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2980–2988, 2017, doi:10.1109/ICCV.2017.322.

[29] Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., Loy, C. C., y Lin, D., "Hybrid task cascade for instance segmentation," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4969–4978, 2019.

[30] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., y Berg, A. C., "SSD: Single shot multibox detector," en Computer Vision – ECCV 2016 (Leibe, B., Matas, J., Sebe, N., y Welling, M., eds.), (Cham), pp. 21–37, Springer International Publishing, 2016.

[31] Redmon, J. y Farhadi, A., "YOLOv3: An incremental improvement," CoRR, vol. abs/1804.02767, 2018, http://arxiv.org/abs/1804.02767.

[32] Bochkovskiy, A., Wang, C., y Liao, H. M., "Yolov4: Optimal speed and accuracy of object detection," CoRR, vol. abs/2004.10934, 2020, https://arxiv.org/abs/2004.10934.

[33] Wang, C.-Y., Bochkovskiy, A., y Liao, H.-Y. M., "Scaled-YOLOv4: Scaling cross stage partial network," en 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 13024–13033, 2021, doi:10.1109/CVPR46437.2021.01283.

[34] Yan, B., Fan, P., Lei, X., Liu, Z., y Yang, F., "A real-time apple targets detection method for picking robot based on improved YOLOv5," Remote Sensing, vol. 13, no. 9, 2021, doi:10.3390/rs13091619.

[35] Wang, C.-Y., Bochkovskiy, A., y Liao, H.-Y. M., "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022, doi:10.48550/ARXIV.2207.02696.

[36] Wu, Z., Chen, Y., Zhao, B., Kang, X., y Ding, Y., "Review of weed detection methods based on computer vision," Sensors, vol. 21, no. 11, 2021, https://www.mdpi.com/1424-8220/21/11/3647.

[37] jun tao, X., Liu, Z., Lin, R., Bu, R., He, Z., Yang, Z., y Liang, C., "Green grape detection and picking-point calculation in a night-time natural environment using a charge-coupled device (ccd) vision sensor with artificial illumination," Sensors, vol. 18, 2018, doi:10.3390/s18040969.

[38] Kang, H., Zhou, H., Wang, X., y Chen, C., "Real-time fruit recognition and grasping estimation for robotic apple harvesting," Sensors, vol. 20, 2020, doi:10.3390/s20195670.

[39] Puttemans, S., Vanbrabant, Y., Tits, L., y Goedemé, T., "Automated visual fruit detection for harvest estimation and robotic harvesting," en 2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA), pp. 1–6, 2016, doi:10.1109/IPTA.2016.7820996.

[40] Viola, P. y Jones, M., "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, vol. 1, pp. I–I, 2001, doi:10.1109/CVPR.2001.990517.

[41] Bhusal, S., Karkee, M., y Zhang, Q., "Apple dataset benchmark from orchard environment in modern fruiting wall," 2019, doi:https://doi.org/10.7273/000001752.

[42] Häni, N., Roy, P., y Isler, V., "Minneapple: A benchmark dataset for apple detection and segmentation," IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 852–858, 2020, doi:10.1109/LRA.2020.2965061.

[43] Bargoti, S. y Underwood, J., "Deep fruit detection in orchards," en 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 3626–3633, 2017, doi:10.1109/ICRA.2017.7989417.

[44] Kang, H. y Chen, C., "Fruit detection, segmentation and 3d visualisation of environments in apple orchards," Comput. Electron. Agric., vol. 171, p. 105302, 2020.

[45] Jia, W., Wang, Z., Zhang, Z., Yang, X., Hou, S., y Zheng, Y., "A fast and efficient green apple object detection model based on foveabox," Journal of King Saud University - Computer and Information Sciences, 2022, doi:https://doi.org/10.1016/j.jksuci.2022.01.005.

[46] Yuan, T., Lv, L., Zhang, F., Fu, J., Gao, J., Zhang, J., Li, W., Zhang, C., y Zhang, W., "Robust cherry tomatoes detection algorithm in greenhouse scene based on ssd," Agriculture, vol. 10, p. 160, 2020, doi:10.3390/agriculture10050160.

[47] Wu, L., Ma, J., Zhao, Y., y Liu, H., "Apple detection in complex scene using the improved YOLOv4 model," Agronomy, vol. 11, no. 3, 2021, doi:10.3390/agronomy11030476.

[48] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., y Belongie, S., "Feature pyramid networks for object detection," en 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 936–944, 2017, doi:10.1109/CVPR.2017.106.

[49] Kang, H. y Chen, C., "Fruit detection and segmentation for apple harvesting using visual sensor in orchards," Sensors, vol. 19, p. 4599, 2019, doi:10.3390/s19204599.

[50] Yao, J., Yu, Z., Yu, J., y Tao, D., "SPRNet: Single-pixel reconstruction for one-stage instance segmentation," IEEE Transactions on Cybernetics, vol. 51, pp. 1731–1742, 2021.

[51] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., y Yuille, A., "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PP, 2016, doi:10.1109/TPAMI.2017.2699184.

[52] Tu, S., Pang, J., Liu, H., Zhuang, N., Chen, Y., Zheng, C., Wan, H., y Xue, Y., "Passion fruit detection and counting based on multiple scale faster R-CNN using rgb-d images," Precision Agriculture, pp. 1–20, 2020.

[53] Zhang, W., Liu, Y., Chen, K., Li, H., Duan, Y., Wenbin, W., Shi, Y., y Guo, W., "Lightweight fruit-detection algorithm for edge computing applications," Frontiers in Plant Science, vol. 12, p. 740936, 2021, doi:10.3389/fpls.2021.740936.

[54] Louëdec, J. L. y Cielniak, G., "3d shape sensing and deep learning-based segmentation of strawberries," Computers and Electronics in Agriculture, vol. 190, p. 106374, 2021, doi:10.1016/j.compag.2021.106374.

[55] Fan, Y., Zhang, S., Feng, K., Qian, K., Wang, Y., y Qin, S., "Strawberry maturity recognition algorithm combining dark channel enhancement and YOLOv5," Sensors, vol. 22, no. 2, 2022, doi:10.3390/s22020419.

[56] Zu, L., Zhao, Y., Liu, J., Su, F., Zhang, Y., y Liu, P., "Detection and segmentation of mature green tomatoes based on Mask R-CNN with automatic image acquisition approach," Sensors, vol. 21, p. 7842, 2021, doi:10.3390/s21237842.

[57] Wang, C.-Y., Mark Liao, H.-Y., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., y Yeh, I.-H., "CSPNet: A new backbone that can enhance learning capability of CNN," en 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1571–1580, 2020, doi:10.1109/CVPRW50498.2020.00203.

[58] Tan, M. y Le, Q. V., "Efficientnet: Rethinking model scaling for convolutional neural networks," CoRR, vol. abs/1905.11946, 2019, http://arxiv.org/abs/1905.11946.

[59] Ishimwe, R., Abutaleb, K., y Ahmed, F., "Applications of thermal imaging in agriculture - a review," Advances in Remote Sensing, vol. 3, pp. 128–140, 2014, doi:10.4236/ars.2014.33011.

[60] Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., y Mccool, C., "Deepfruits: A fruit detection system using deep neural networks," Sensors, vol. 16, p. 1222, 2016, doi:10.3390/s16081222.

[61] Gené-Mola, J., Vilaplana, V., Rosell-Polo, J. R., Morros, J.-R., Ruiz-Hidalgo, J., y Gregorio, E., "Kfuji rgb-ds database: Fuji apple multi-modal images for fruit detection with color, depth and range-corrected ir data," Data in Brief, vol. 25, p. 104289, 2019, doi:https://doi.org/10.1016/j.dib.2019.104289.

[62] Qi, C. R., Liu, W., Wu, C., Su, H., y Guibas, L. J., "Frustum pointnets for 3d object detection from rgb-d data," en 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 918–927, 2018, doi:10.1109/CVPR.2018.00102.

[63] Li, T., Feng, Q., Qiu, Q., Xie, F., y Zhao, C., "Occluded apple fruit detection and localization with a frustum-based point-cloud-processing approach for robotic harvesting," Remote Sensing, vol. 14, no. 3, 2022, doi:10.3390/rs14030482.

[64] Gené-Mola, J., Vilaplana, V., Rosell-Polo, J. R., Morros, J.-R., Ruiz-Hidalgo, J., y Gregorio, E., "Multi-modal deep learning for fuji apple detection using rgb-d cameras and their radiometric capabilities," Computers and Electronics in Agriculture, vol. 162, pp. 689–698, 2019, doi:https://doi.org/10.1016/j.compag.2019.05.016.

[65] Behera, S. K., Rath, A. K., y Sethy, P. K., "Fruits yield estimation using Faster R-CNN with miou," Multimedia Tools Appl., vol. 80, p. 19043–19056, 2021, doi:10.1007/s11042-021-10704-7.

[66] Zhu, J.-Y., Park, T., Isola, P., y Efros, A. A., "Unpaired image-to-image translation using cycle-consistent adversarial networks," en 2017 IEEE International Conference

on Computer Vision (ICCV), pp. 2242–2251, 2017, doi:10.1109/ICCV.2017.244.

[67] Halstead, M., McCool, C., Denman, S., Perez, T., y Fookes, C., "Fruit quantity and ripeness estimation using a robotic vision system," IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 2995–3002, 2018, doi:10.1109/LRA.2018.2849514.

[68] Wang, Z., Hu, M., y Zhai, G., "Application of deep learning architectures for accurate and rapid detection of internal mechanical damage of blueberry using hyperspectral transmittance data," Sensors, vol. 18, no. 4, 2018, doi:10.3390/s18041126.

[69] Firouzjaei, R., Minaee, S., y Beheshti, B., "Sweet lemon mechanical damage detection using image processing technique and uv radiation," Journal of Food Measurement and Characterization, vol. 12, 2018, doi:10.1007/s11694-018-9766-8.

[70] V G, N. y Pinto, A., Defects Detection in Fruits and Vegetables Using Image Processing and Soft Computing Techniques, pp. 325–337. Springer Singapore, 2021, doi:10.1007/978-981-15-8603-3_29.

[71] Jia, W., Zhang, Y., Lian, J., Zheng, Y., Zhao, D., y Li, C., "Apple harvesting robot under information technology: A review," International Journal of Advanced Robotic Systems, vol. 17, no. 3, p. 1729881420925310, 2020, doi:10.1177/1729881420925310.

[72] Kuhn, H. W., "The hungarian method for the assignment problem," Naval Research Logistics (NRL), vol. 52, 1955.

[73] Liu, G., Hou, Z., Liu, H., Liu, J., Zhao, W., y Li, K., "Tomatodet: Anchor-free detector for tomato detection," Frontiers in Plant Science, vol. 13, p. 942875, 2022, doi:10.3389/fpls.2022.942875.

[74] Zhou, Y., Tang, Y., Zou, X., Wu, M., Tang, W., Meng, F., Zhang, Y., y Kang, H., "Adaptive active positioning of camellia oleifera fruit picking points: Classical image processing and YOLOv7 fusion algorithm," Applied Sciences, vol. 12, no. 24, 2022, doi:10.3390/app122412959.

[75] Wilms, C., Johanson, R., y Frintrop, S., "Localizing small apples in complex apple orchard environments," ArXiv, vol. abs/2202.11372, 2022, doi:10.48550/ARXIV.2202.11372.

[76] Liang, C., Xiong, J., Zheng, Z., Zhong, Z., Li, Z., Chen, S., y Yang, Z., "A visual detection method for nighttime litchi fruits and fruiting stems," Computers and Electronics in Agriculture, vol. 169, p. 105192, 2020, doi:https://doi.org/10.1016/j.compag.2019.105192.

[77] Afonso, M., Fonteijn, H., Fiorentin, F. S., Lensink, D., Mooij, M., Faber, N., Polder, G., y Wehrens, R., "Tomato fruit detection and counting in greenhouses using deep learning," Frontiers in Plant Science, vol. 11, 2020, doi:10.3389/fpls.2020.571299.

[78] Woods, N. y Wuzor, G., "On tree guava fruit detection and yield estimation," International Journal of Scientific and Engineering Research, vol. 11, pp. 723 – 731, 2020.

[79] Santos, T. T., de Souza, L. L., dos Santos, A. A., y Avila, S., "Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association," Computers and Electronics in Agriculture, vol. 170, p. 105247, 2020, doi:https://doi.org/10.1016/j.compag.2020.105247.

[80] Guo, Q., Chen, Y., Tang, Y., Zhuang, J., He, Y., Hou, C., Chu, X., Zhong, Z., y Luo, S.,

"Lychee fruit detection based on monocular machine vision in orchard environment," Sensors, vol. 19, no. 19, 2019, doi:10.3390/s19194091.

[81] Grimm, J., Herzog, K., Rist, F., Kicherer, A., Töpfer, R., y Steinhage, V., "An adaptable approach to automated visual detection of plant organs with applications in grapevine breeding," Biosystems Engineering, vol. 183, pp. 170–183, 2019, doi:https://doi.org/10.1016/j.biosystemseng.2019.04.018.

[82] Jaco Fourie, Jeffrey Hsiao, A. W., "Crop yield estimation using deep learning," 7th Asian-Australasian Conference on Precision Agriculture, 2017.

[83] Cheng, H., Damerow, L., Sun, Y., y Blanke, M., "Early yield prediction using image analysis of apple fruit and tree canopy features with neural networks," Journal of Imaging, vol. 3, p. 6, 2017, doi:10.3390/jimaging3010006.

[84] Chen, S., Skandan, S., Dcunha, S., Das, J., Okon, E., Qu, C., Taylor, C., y Kumar, V., "Counting apples and oranges with deep learning: A data-driven approach," IEEE Robotics and Automation Letters, vol. PP, pp. 1–1, 2017, doi:10.1109/LRA.2017.2651944.

[85] Song, Y., Glasbey, C., Horgan, G., Polder, G., Dieleman, J., y van der Heijden, G., "Automatic fruit recognition and counting from multiple images," Biosystems Engineering, vol. 118, pp. 203–215, 2014, doi:https://doi.org/10.1016/j.biosystemseng.2013.12.008.

[86] Wang, W., Dai, J., Chen, Z., Huang, Z., Li, Z., Zhu, X., Hu, X., Lu, T., Lu, L., Li, H., Wang, X., y Qiao, Y., "Internimage: Exploring large-scale vision foundation models with deformable convolutions," 2022, doi:10.48550/ARXIV.2211.05778.

[87] Zhu, X., Su, W., Lu, L., Li, B., Wang, X., y Dai, J., "Deformable DETR: deformable transformers for end-to-end object detection," CoRR, vol. abs/2010.04159, 2020, https://arxiv.org/abs/2010.04159.

[88] Zhang, H., Li, F., Liu, S., Zhang, L., Su, H., Zhu, J., Ni, L. M., y Shum, H.-Y., "Dino: Detr with improved denoising anchor boxes for end-to-end object detection," 2022, doi:10.48550/ARXIV.2203.03605.

[89] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., y Guo, B., "Swin transformer: Hierarchical vision transformer using shifted windows," 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 9992–10002, 2021.

[90] Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., Wei, F., y Guo, B., "Swin transformer v2: Scaling up capacity and resolution," en 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11999–12009, 2022, doi:10.1109/CVPR52688.2022.01170.

[91] Qiao, S., Chen, L., y Yuille, A., "Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution," en 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (Los Alamitos, CA, USA), pp. 10208–10219, IEEE Computer Society, 2021, doi:10.1109/CVPR46437.2021.01008.

[92] Xie, S., Girshick, R., Dollár, P., Tu, Z., y He, K., "Aggregated residual transformations for deep neural networks," en 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5987–5995, 2017, doi:10.1109/CVPR.2017.634.

[93] Tan, M., Pang, R., y Le, Q. V., "Efficientdet: Scalable and efficient object detection," en 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),

pp. 10778–10787, 2020, doi:10.1109/CVPR42600.2020.01079.

[94] Wang, C., Yeh, I., y Liao, H. M., "You only learn one representation: Unified network for multiple tasks," CoRR, vol. abs/2105.04206, 2021, https://arxiv.org/abs/2105.04206.

[95] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., y Polosukhin, I., "Attention is all you need," en Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, (Red Hook, NY, USA), p. 6000–6010, Curran Associates Inc., 2017.

[96] He, K., Zhang, X., Ren, S., y Sun, J., "Deep residual learning for image recognition," en 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, 2016, doi:10.1109/CVPR.2016.90.

[97] Dollár, P., Singh, M., y Girshick, R., "Fast and accurate model scaling," en 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 924–932, 2021, doi:10.1109/CVPR46437.2021.00098.

[98] Liang, T., Chu, X., Liu, Y., Wang, Y., Tang, Z., Chu, W., Chen, J., y Ling, H., "Cbnet: A composite backbone network architecture for object detection," IEEE Transactions on Image Processing, vol. 31, pp. 6893–6906, 2022.

[99] Hu, J., Shen, L., y Sun, G., "Squeeze-and-excitation networks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7132–7141, 2018.

[100] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., y Zagoruyko, S., "End-to-end object detection with transformers," en Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I, (Berlin, Heidelberg), p. 213–229, Springer-Verlag, 2020, doi:10.1007/978-3-030-58452-8_13.

[101] Dai, X., Chen, Y., Xiao, B., Chen, D., Liu, M., Yuan, L., y Zhang, L., "Dynamic head: Unifying object detection heads with attentions," 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7369–7378, 2021.

[102] Cai, Z. y Vasconcelos, N., "Cascade r-cnn: Delving into high quality object detection," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6154–6162, 2018.

[103] Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., y Tian, Q., "Centernet: Keypoint triplets for object detection," en 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 6568–6577, 2019, doi:10.1109/ICCV.2019.00667.

[104] Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., y Tian, Q., "Centernet++ for object detection," 2022, doi:10.48550/ARXIV.2204.08394.

[105] Li, L. H., Zhang, P., Zhang, H., Yang, J., Li, C., Zhong, Y., Wang, L., Yuan, L., Zhang, L., Hwang, J.-N., Chang, K.-W., y Gao, J., "Grounded language-image pre-training," en 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10955–10965, 2022, doi:10.1109/CVPR52688.2022.01069.

[106] Bao, H., Dong, L., y Wei, F., "Beit: Bert pre-training of image transformers," ArXiv, vol. abs/2106.08254, 2021.

[107] Wang, W., Bao, H., Dong, L., Bjorck, J., Peng, Z., Liu, Q., Aggarwal, K., Mohammed, O. K., Singhal, S., Som, S., y Wei, F., "Image as a foreign language: Beit pretraining for all vision and vision-language tasks," 2022, doi:10.48550/ARXIV.2208.10442.

[108] Zhang, H., Zhang, P., Hu, X., Chen, Y.-C., Li, L. H., Dai, X., Wang, L., Yuan, L., Hwang, J.-N., y Gao, J., "Glipv2: Unifying localization and vision-language understanding," 2022, doi:10.48550/ARXIV.2206.05836.

[109] Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., y Zhang, L., "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," 2023, doi:10.48550/ARXIV.2303.05499.

[110] Ramachandran, P., Zoph, B., y Le, Q. V., "Searching for activation functions," CoRR, vol. abs/1710.05941, 2017, http://arxiv.org/abs/1710.05941.

[111] Misra, D., "Mish: A self regularized non-monotonic neural activation function," CoRR, vol. abs/1908.08681, 2019, http://arxiv.org/abs/1908.08681.

[112] Hendrycks, D. y Gimpel, K., "Gaussian error linear units (gelus)," 2016, doi:10.48550/ARXIV.1606.08415.

[113] Bodla, N., Singh, B., Chellappa, R., y Davis, L. S., "Soft-nms — improving object detection with one line of code," en 2017 IEEE International Conference on Computer Vision (ICCV), pp. 5562–5570, 2017, doi:10.1109/ICCV.2017.593.

[114] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., y Savarese, S., "Generalized intersection over union: A metric and a loss for bounding box regression," en 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 658–666, 2019, doi:10.1109/CVPR.2019.00075.

[115] Lin, T.-Y., Goyal, P., Girshick, R., He, K., y Dollár, P., "Focal loss for dense object detection," en 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2999–3007, 2017, doi:10.1109/ICCV.2017.324.

[116] Ghiasi, G., Cui, Y., Srinivas, A., Qian, R., Lin, T.-Y., Cubuk, E. D., Le, Q. V., y Zoph, B., "Simple copy-paste is a strong data augmentation method for instance segmentation," en 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2917–2927, 2021, doi:10.1109/CVPR46437.2021.00294.

[117] Yang, S., Luo, P., Loy, C. C., y Tang, X., "Wider face: A face detection benchmark," en IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[118] Zoph, B., Ghiasi, G., Lin, T., Cui, Y., Liu, H., Cubuk, E. D., y Le, Q. V., "Rethinking pre-training and self-training," CoRR, vol. abs/2006.06882, 2020, https://arxiv.org/abs/2006.06882.

[119] Xu, M., Zhang, Z., Hu, H., Wang, J., Wang, L., Wei, F., Bai, X., y Liu, Z., "End-to-end semi-supervised object detection with soft teacher," 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 3040–3049, 2021.

[120] Simonyan, K. y Zisserman, A., "Very deep convolutional networks for large-scale image recognition," CoRR, vol. abs/1409.1556, 2015.

[121] Wang, J., Zhang, Z., Luo, L., Zhu, W., Chen, J., y Wang, W., "Swingd: A robust grape bunch detection model based on swin transformer in complex vineyard environment," Horticulturae, vol. 7, no. 11, 2021, doi:10.3390/horticulturae7110492.

[122] Smith, L. N. y Topin, N., "Super-convergence: very fast training of neural networks using large learning rates," en Defense + Commercial Sensing, 2019.