UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IDENTIFICACIÓN Y CONTEO DE FRUTOS EN ARÁNDANOS Y UVAS POR MEDIO DE IMÁGENES MULTIESPECTRALES Y DEEP LEARNING

AUTORES: Diego Francisco Anabalon Anabalon
Luis Guillermo Osses Gutiérrez.

PROYECTO DE TÍTULO PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN AUTOMATIZACIÓN

CONCEPCIÓN 2021

UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IDENTIFICACIÓN Y CONTEO DE FRUTOS EN ARÁNDANOS Y UVAS POR MEDIO DE IMÁGENES MULTIESPECTRALES Y DEEP LEARNING

PROYECTO DE TÍTULO PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN AUTOMATIZACIÓN

AUTORES: Diego Francisco Anabalon Anabalon
Luis Guillermo Osses Gutiérrez.

Profesor guía: Dr. Cristhian Aguilera Carrasco
Profesor Co-guía: Dr.Pedro Campos Soto

Resumen

En la actualidad, Chile se ha posicionado y consolidado en la exportación de arándanos, logrando ser el mayor exportador del hemisferio sur desde 2015 a 2018 y los años posteriores junto con Perú [1]. Esto implica un monitoreo y análisis constante para lograr cumplir con los estándares de calidad. Los arándanos poseen características particulares que lo definen, incluyendo color, diámetro y defectos en su composición física.
Aunque estadísticamente las exportaciones han ido en aumento, los productores de baja y media escala realizan el proceso de selección de manera manual, no pudiendo cumplir con los estándares de calidad requeridos en países extranjeros. Es por esto, que surge la necesidad de un análisis exhaustivo, preciso y eficiente del fruto. Para lograr una clasificación y detección precisa del punto de maduración, buscando las condiciones ideales para la cosecha y la toma de decisiones en la postcosecha.

De esta forma, en el presente trabajo de título exploramos el potencial de usar métodos de Aprendizaje Automático junto con 400 imágenes que luego aumentamos a 1130 imágenes VIS-NIR capturadas en un ambiente natural al aire libre, para la clasificación multiclase y conteo de arándanos de la especie Highbush Legacy, en diferentes etapas de crecimiento.

Implementamos tres algoritmos de aprendizaje profundo, YOLOv5, YOLOR y Mask R-CNN, siendo YOLOv5 el con mejor resultado con un mAP (precisión media) del 85 %, seguido de YOLOR con un mAP 85 %, pero detectando menos arándanos que YOLOv5. Mask R-CNN que a pesar de entrenarlo con pocas etapas alcanzó un mAP del 70 %.

Si bien utilizamos algoritmos de última generación los resultados indican que tienen una precisión media cercana al 85 %, creemos que funcionaria mejor en aplicaciones donde las imágenes son capturadas con un campo de visión acotado, donde el objeto se aprecia claramente. Y aunque partimos con la hipótesis de que YOLOv5 tiene dificultades para detectar objetos pequeños que estén de fondo, YOLOv5 muestra la capacidad de detectar arándanos en imágenes con un campo de visión más amplio. Mask R-CNN genera buenos resultados, pero se ve limitado por los recursos computacionales que necesita.
Para proyectos futuros se puede usar esta misma metodología para otro tipo de cultivos donde el fruto sea de mayor tamaño como manzana, naranjas, paltas o limones, donde estos algoritmos tendrán un mejor desempeño. Por ahora se deben seguir mejorando los algoritmos y el numero de muestras por cada clase, para poder enfrentar de manera más robusta a las oclusiones y frutos pequeños.

Con este estudio, creamos una nueva base de datos con anotaciones para la detección y segmentación de instancias de arándanos, y el repositorio de Github con el código utilizado. Además, se utilizó un conjunto de datos de test idéntico con cada algoritmo, permitiéndonos comparar de forma más estandarizada los resultados. [Conjunto de test].

Abstract

Currently, Chile is positioned and consolidated in the export of blueberries, becoming the largest exporter in the southern hemisphere from 2015 to 2018 and the following years along with Peru [1]. This involves constant monitoring and analysis to meet quality standards; blueberries have particular defining characteristics, including color, diameter, defects and multispectral analysis.

Although statistically exports have been increasing, low and medium scale producers carry out the selection process manually, not being able to meet the quality standards required in foreign countries. For this reason, there is a need for an exhaustive, precise and efficient analysis of the fruit. To achieve an accurate classification and detection of the ripening point of fruit, looking for the ideal conditions for harvesting and making decisions in the post-harvest.

In this way, in the present title work we explore the potential of Machine Learning methods together with 400 images that we later increase to 1130 VIS-NIR images captured in a natural outdoor environment, for the multiclass classification and counting of blueberries of the High-bush Legacy species, at different stages of growth.

We implemented three deep learning algorithms, YOLOv5, YOLOR and Mask R-CNN, with YOLOv5 being the best performer with a mAP (mean precision) of 85 %, followed by YOLOR with a mAP 85 %, but detecting less blueberries than YOLOv5. Mask R-CNN that despite training him with few stages reached a mAP of 70 %.

Although we use state-of-the-art algorithms, the results indicate that they have an average precision close to 85 %, we believe that it would work better in applications where the images are captured with a limited field of view, where the object is clearly seen. And while we start with the hypothesis that YOLOv5 has difficulty detecting small objects in the background, YOLOv5 shows the ability to detect blueberries in images with a wider field of view. Mask R-CNN generates good results, but is limited by the computational resources it needs.
For future projects, this same methodology can be used for other types of crops where the fruit is larger, such as apples, oranges, avocados or lemons, where these algorithms will have a better performance. For now, the algorithms and the number of samples per class should be further improved in order to be able to deal more robustly with occlusions and small fruits.

With this study, we created a new database with annotations for detection and segmentation of blueberry instances, and the Github repository with the code used. In addition, an identical test dataset was used with each algorithm, allowing us to compare in a more standardized way the results.. [Test dataset].

Dedicatoria

Dedicamos el presente trabajo a nuestros padres, que con todo el amor y dedicación nos criaron y enseñaron. Gracias por cada palabra de aliento y por el apoyo incondicional brindado en este largo camino. Dedicado a todos los que estuvieron presentes en este proceso.

Agradecimientos

En primera instancia agradecemos a Dios por permitirnos culminar esta etapa con una experiencia enriquecedora, a nuestros profesores, en especial al profesor guía que nos facilito todo el equipo y siempre estuvo dispuesto a ayudarnos.
El mas sincero agradecimiento a nuestros compañeros, a nuestra familia que siempre estuvieron apoyándonos y a toda la comunidad de Github e investigadores que siguen aportando en el desarrollo y mejora de tantas aplicaciones.

Tabla de Contenidos

Índice de figuras

Índice de tablas

CAPÍTULO 1. INTRODUCCIÓN

1.1. Motivación.

Chile y Perú son los máximos exportadores de arándanos en el mundo, y la mayor parte de su producción se exporta a mercados de consumo lejanos, como lo son Estados Unidos, China, Europa y Australia. Para poder ser aceptados en estos mercados se deben cumplir altos estándares de calidad, donde la fruta debe ser completamente azul y firme. Por lo tanto, la logística hace que la gestión de la recolección sea crítica. Esto es particularmente difícil para los productores, ya que, los grupos de arándanos no suelen ser cosechados por completo al mismo tiempo, debido a que la fruta se encuentra en diferentes etapas de crecimiento simultáneamente.

En la actualidad, la valoración y apreciación de la fruta, previa a la cosecha se hace por medio de agricultores, que hacen una inspección visual de la fruta. Este proceso tiene notables limitaciones, ya que la visión del agricultor tiene una alta subjetividad, por este motivo, en los últimos años las técnicas de Visión por Computador y Aprendizaje Automático están adquiriendo un fuerte rol en esta aplicación y en la agricultura en general.

El sistema automatizado de Visión por Computador es uno de los métodos más utilizados para la estimación de rendimiento de varios cultivos, como Manzanas y Cítricos, y para la optimización y destino de recursos.

1.2. Industria frutícola en Chile

La industria frutícola en Chile tiene un reconocimiento a nivel mundial y es el principal productor y exportador de frutas del hemisferio sur, con un gran énfasis en uvas y arándanos. Esto se ha facilitado gracias a las condiciones climáticas favorables de Chile (edafoclimaticas) [2].

La clasificación climática de Köppen, indica que Chile posee al menos 7 subtipos climáticos en su territorio, desde el clima desértico del Norte hasta el clima polar en el territorio Antártico Chileno. Y las cuatro estaciones están presentes en todo el país.

Es por esto que en los últimos 20 años, la superficie fruticola en Chile ha crecido exponencialmente. Desde 1999 a 2019 creció un 87,6 %, como se ve en la figura 1. Esta evolución muestra la importancia e influencia que tiene la industria en el país, la que se traduce en exportaciones e ingresos. Por esto mismo, la industria despierta un gran interés.

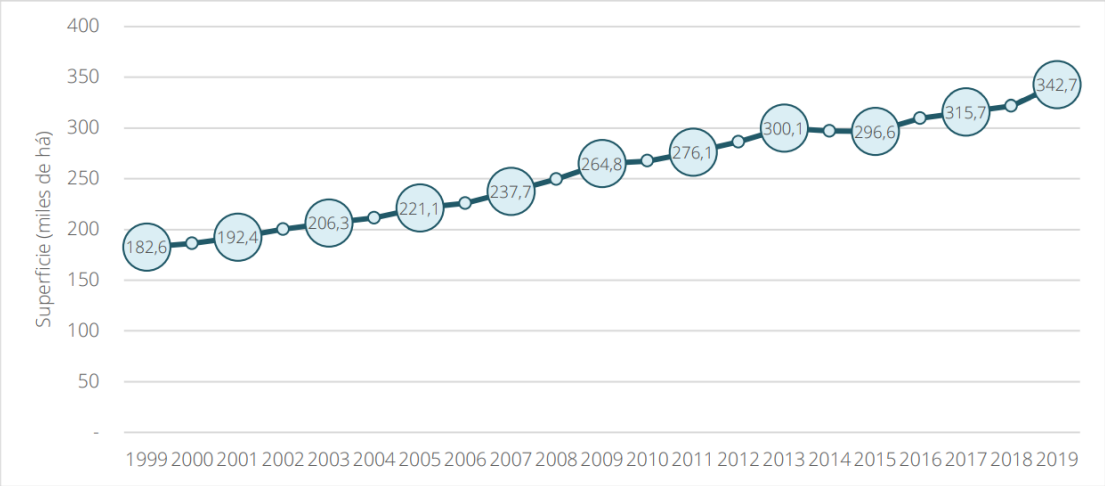


Figura 1: Evolución de la superficie frutícola en Chile.
Fuente: Catastro Frutícola Odepa Ciren (2020, actualización anual desde 2002)[2].

1.2.1. Arándanos en Chile

Chile ha tenido gran influencia en el mercado del arándano, desde el 2015 hasta 2018 fue el mayor exportador de arándano fresco en el Hemisferio Sur llegando a generar 111.109 toneladas en el 2018.

Junto con Perú en los últimos años tienen una tendencia positiva en exportaciones de arándanos, lo que se traduce en una alza y aumento significativo de ingresos al país. Perú desplazó a Chile en el Hemisferio Sur, convirtiéndose en el principal exportador, con el mayor índice de participación en el ultimo año.

Como se muestra en la tabla 1, existe solo una variación negativa en tres variedades, lo que demuestra la vigencia y relevancia del arándano y sus variedades en Chile.

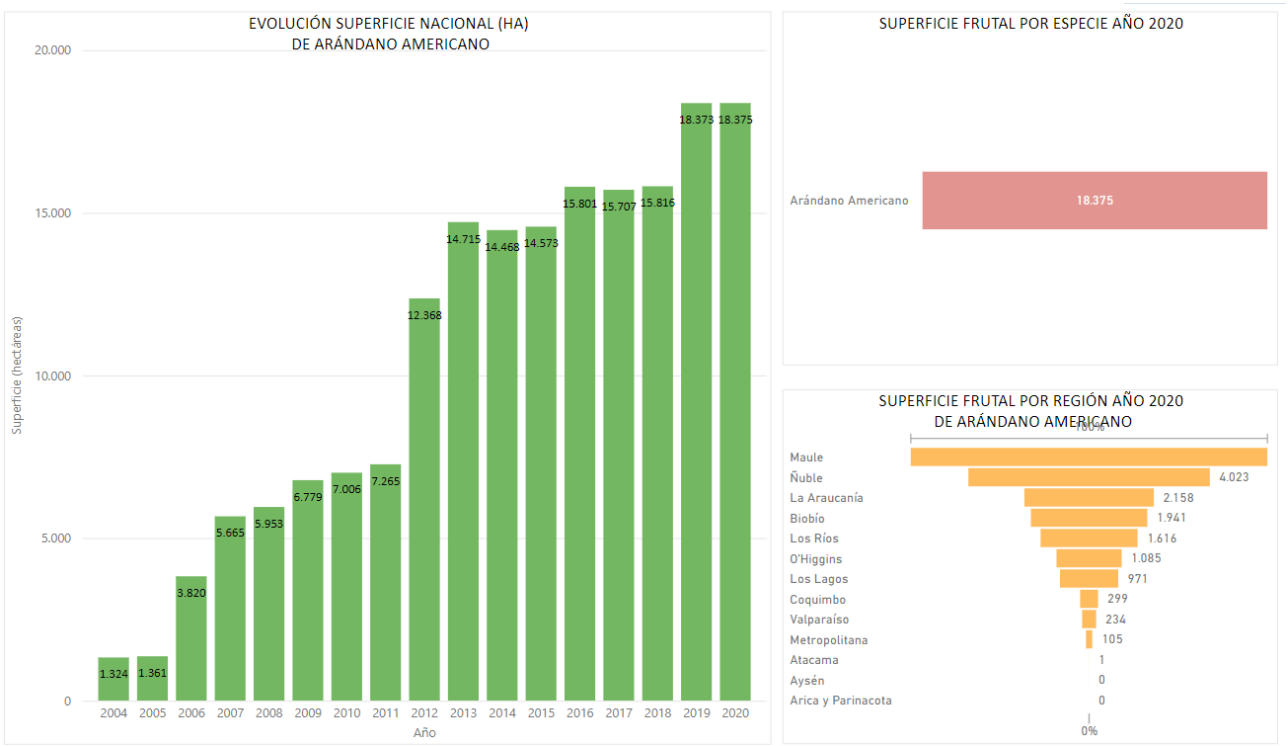


Figura 2: Evolución de la superficie de arándano americano [HA].
Fuente: Odepa, 2021.

Variedades	2016/17	2017/18	2018/19	2019/20	2020/21	$\Delta 19/20 - 20/21$
Legacy	19,291	22,777	25,760	24,819	28,579	16 %
Duke	17.466	16.191	18.667	19.114	20.401	7 %
Brigitta	18.266	14.036	15.953	13.737	12.685	-8 %
Brightwell	9.357	11.596	8.754	9.937	10.771	8 %
O'Neill	8.065	7.473	7.171	4.182	3.826	-9 %
Star	3.101	5.000	4.179	3.524	3.733	6 %
Ochlockonee	3.634	4.866	3.721	3.051	2.700	-12 %
Blue Ribbon	83	295	630	1.316	2.452	86 %
Top Shelf	123	401	997	1.666	2.412	45 %
Suzie Blue	119	335	747	1.212	2.302	90 %
Total	103.693	110.198	111.109	109.421	117.898	8 %

Tabla 1: Variedades de arándanos exportados [Toneladas].
Fuente: iQonsulting con información de SAG - ASOEX, 2020.

Las variedades de arándanos exportados en Chile descritos en la tabla 1, poseen diferentes características, se aprecia que la variedad predominante es Highbush Legacy y la variación anual sugiere un crecimiento exponencial a lo largo de los años dado el mismo crecimiento descrito anteriormente con la evolución de superficie de arándano en hectáreas.

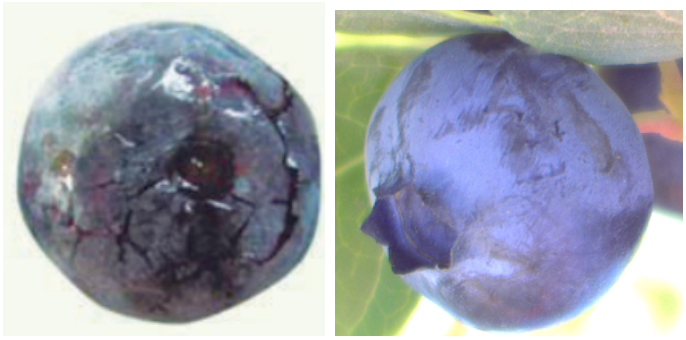
Cada una de estas variedades requiere distintos cuidados, almacenamientos a distintas temperaturas y horas de frío definidas para cada una de estas.

En este trabajo utilizamos la variedad predominante en Chile, por lo que las imágenes capturadas son de la variedad **Highbush Legacy**, fue escogida por la relevancia que tiene en la industria del arándano, dado que actualmente es la variedad con el mayor volumen de toneladas exportadas.

1.2.2. Criterios de selección de arándanos.

Para definir la variedad requerida en la postcosecha, ya sea para exportación o mercado nacional, el manual de manejo del arándano creado por el instituto de investigaciones agropecuarias (INIA) y el instituto de desarrollo agropecuario (INDAP) se consideran:

- Firmeza: Estado físico relacionado con la maduración. En las distintas etapas de crecimiento y maduración, la consistencia esta dada por componentes internos.



((a)) Piel suelta ((b)) Piel firme

Figura 3: Estados de piel en arándanos.
Fuente: Manual de manejo del arándano (INIA-INDAP, 2017) [3].

- Color: Dada la diferencia de color entre cada etapa de madurez, éste define que tan maduro está el arándano, como se ve en la figura 4. Se aprecia en 1 y 2 es la etapa de maduración inicial, en 3 se encuentra la maduración ideal y 4 muestra una sobre maduración donde se ha perdido consistencia y firmeza.

Figura 4: Etapas de maduración.
Fuente: Manual de manejo del arándano (INIA-INDAP, 2017) [3].

Observación:
Se relaciona la madurez del fruto con el color azul/morado, además, la firmeza se asocia a este color y a su forma esférica particular. La que puede ser apreciada en las imágenes generadas en este trabajo, ya que se asocia un fruto maduro y sano con el color azul/morado, morado claro se considera medio maduro y verde o rojizo se considera inmaduroo.

- Solidos solubles: Medición de azúcar, carbohidratos, ácidos, proteínas, grasas y minerales del fruto. Este se mide en Brix y se considera un rango de 10 a 14.
- Otros: También se usan mediciones de acidez titulable y desordenes.

Problema en el almacenaje:

- Pudriciones: Al almacenar los arándanos en frío, dependiendo de la cadena de frío existen distintos tipos de pudriciones del fruto, siendo irreversibles figura (5).

Figura 5: Pudriciones a lo largo de almacenamiento en frío.
Fuente: Manual de manejo del arándano (INIA-INDAP, 2017) [3].

1.3. Planteamiento del problema.

Una estimación del rendimiento del campo de arándanos antes de la cosecha ayuda a los productores a encontrar posibles problemas con anticipación. Además, es útil para que se tomen decisiones adicionales, como el riego, control de maleza e incluso de plagas. La asignación de trabajo para una cosecha eficiente en un campo de arándanos de grandes dimensiones puede reducir gran parte del costo de recolección

Durante la temporada de cosecha, los arándanos individuales en un grupo de frutas maduran en diferentes momentos, estos grupos pueden contener todas las etapas de crecimiento, como son la fruta joven (color verde), la fruta intermedia (color rojo) y la fruta madura (azul oscuro / purpura) al mismo tiempo. Se realiza una inspección visual por parte de los recolectores antes de realizar la cosecha, esta inspección es subjetiva dada las propias limitaciones del ser humano, por lo que la cosecha es realizada en múltiples etapas.

Debido a este problema se propone un método que explora la viabilidad de utilizar imágenes multiespectrales y del espectro visible en la clasificación de diferentes etapas de crecimiento de los arándanos. Con el objetivo de combinar estos dos tipos de imágenes, para identificar y contar el número y madurez asociada a cada baya contenida en las imágenes, permitiendo estimar el tiempo óptimo de la cosecha.

1.4. Revisión Bibliográfica.

Las siguientes investigaciones se utilizaron como revisión bibliográfica:
Para este estudio se tomó, como referencia múltiples estudios anteriores, analizaremos los más relevantes, como (Ce Yang, 2012) en su trabajo "Blueberry fruit detection by Bayesian classifier and support vector machine based on visible to near-infrared multispectral imaging", [4]. Comenta que la estimación temprana del rendimiento basada en la visión por computadora permite una mejor gestión de la mano de obra y un menor gasto de cosecha en un campo de arándanos a gran escala. En su estudio uso 90 imágenes multiespectrales con bandas en el espectro NIR y RG (red, green) de la variedad de arándanos Highbush del sur 'Sweetcrisp' de un campo de arándanos en Florida. Se recopilaron cinco mil píxeles de fruta pura y 5000 píxeles de fondo de las imágenes. El 66 % de ellos estaban en un conjunto de calibración y el otro 34 % se utilizó como conjunto de validación. Se aplicó un clasificador bayesiano y SVM para la clasificación de la fruta y las clases de fondo. Los resultados muestran que SVM superó al clasificador bayesiano con una tasa de verdaderos positivos más alta (84 % para la clase de fruta y 73 % para la clase de fondo) y una tasa más baja de falsos positivos (27 % para la clase de fruta y 16 % para la clase de fondo). (Walter A. 2019) en su trabajo "Espectroscopia infrarrojo y técnicas de Machine learning y Deep learning para la detección y clasificación de arándanos" [5], se toma como caso de estudio, la clasificación de arándanos, para elaborar una herramienta que permita hacer la clasificación, reducir el costo, el tiempo y optimizar el proceso de detección y clasificación de frutas, este trabajo se realizó con una muestra de 1000 arándanos. Utilizando un equipo de espectroscopia del infrarrojo cercano (NIR), se entrenó una red neuronal, utilizando Python, con ayuda de Keras y Tensor Flow. En la etapa de testing utilizo espectros NIR de nuevas muestras de arándanos, demostrando que se pueden clasificar los arándanos con una exactitud del 92 %, (Walter A. 2019) obtiene mejores resultados que los nuestros debido a que sus imágenes son de arándanos individuales y capturadas en un laboratorio, en cambio nuestras imágenes están capturadas en un ambiente natural, al aire libre, con diversos obstáculos como ramas y hojas que hace más complejo tener buenos resultados. (Sebastian G. 2019) en su trabajo "Deep Blueberry: Quantification of Blueberries in the wild Using Instance Segmentation" [6]. Propone una CNN basada en Mask R-CNN para la detección y segmentación. Se usan redes convolucionales como ResNet (50 y 101) y MobileNetV1, las cuales permiten clasificar imágenes en 1000 categorías de objetos. Se realizó una prueba con la red pre entrenada COCO y esta no es capaz de detectar frutas tan pequeñas, por lo que los entrenamientos se realizan desde cero, por lo que no se necesitan redes tan complejas y profundas. Para medir el rendimiento se usa error de intersección sobre la unión (IoU) y la precisión media por imágenes (mAP). Los mejores resultados en la detección se obtuvieron con ResNet50 alcanzando 0.595 y 0.759 (IoU y mAP). El trabajo se compara con clasificadores como KNN, SVM y AdaBoost. En la etapa de segmentación el score es de 0.909 y 0.759 en la detección de objetos. El mejor resultado se obtiene con la red con menos capas dado el tamaño de datos de entrada. Al usar redes con más capas se necesita un mayor número de datos para no caer en sobre entrenamiento, también se genera la opción de la red convolucional

MobileNet 1 que mejora el tiempo de cálculo por imagen. En nuestro caso si utilizamos la red pre entrenada COCO para entrenar a Mask R-CNN, alcanzando una precisión media del 70 % confirmando que la red COCO tiene dificultades para detectar objetos pequeños, además podemos destacar que YOLOv5 obtuvo un 79,16 % de precisión media y ResNet50 solo alcanzo el 75,9 %. (Arnold W. Schumann, 2019) en su trabajo "Detection of Three Fruit Maturity Stages in WildBlueberry Fields Using Deep Learning Artificial Neural Networks" [7]. Señala que en su mayoría los cultivos de arándanos son cosechados de manera mecánica al final del verano en una sola instancia, esto trae consigo imprecisiones y obtener una mezcla de bayas en distintas etapas de maduración. Por lo que en su estudio utilizó visión artificial para la detección de la madurez de los arándanos en los arbustos y posteriormente se evalúa el rendimiento de está. Se trabaja mediante el análisis de imágenes, donde se es capaz de extraer información útil sobre la que el usuario puede actuar e identificar patrones específicos. Se recolectan 211 imágenes que se subdividen en 4220 imágenes recortadas. Donde 424 imágenes seleccionadas al azar ya recortadas y etiquetadas (10 %) se usan para probar el rendimiento de DL-ANN entrenados y el 90 % restante se utilizó para entrenar cuatro variantes de DL-ANN de detección de objetos. Las imágenes recortadas se etiquetan de acuerdo con su apariencia y color (Verde (no maduro), Rojo (poco maduro) y Azul (maduro)), se usó un programa personalizado desarrollado con el compilador Lazarus. Se evalúa el rendimiento de la visión artificial de aprendizaje profundo con Yolo DL-ANN para detectar la madurez de la fruta de arándano silvestre en los arbustos antes de la cosecha. La familia YOLO usada contiene: YoloV3, YoloV3-spp, YoloV3-tiny, YoloV3-tiny-XNOR. Se utilizó la métrica AP para determinar el rendimiento de la red para cada clase de objeto en el conjunto de validación. Y mAP se utilizó para generalizar una media respecto de las tres clases. En general se logró el entrenamiento, llegando a la convergencia y logrando aprender a reconocer las distintas clases. A pesar de esto todas las redes tienen problemas al identificar el fruto no maduro dado que este es fácil de confundir con el fondo mismo y las hojas. La red YoloV3-spp tuvo la precisión promedio más alta (mAP = 85.3 %) en todas las clases de madurez de la fruta, seguida de YoloV3 (82.3 %), YoloV3-tiny (76.6 %) y YoloV3-tiny-XNOR (46.8 %). Por lo que el aprendizaje profundo con redes Yolo descrito en este artículo permite clasificar y cuantificar el desarrollo de frutos verdes, rojos inmaduros y frutos azules maduros, logrando realizar estimaciones de rendimiento al menos un mes antes de la cosecha. Este estudio podría ser llevado a una escala mayor con un reconocimiento en tiempo real mediante drones y máquinas de monitoreo. (Arnold W. Schumann, 2019), obtiene buenos resultados, un poco mejores que los nuestros debido a que contábamos con pocas imágenes de arándanos poco maduro, lo que reduce el accuracy notablemente. (Wei Zheng, 2020) en su trabajo "Self-adaptive models for predicting soluble solid content of blueberries with biological variability by using near-infrared spectroscopy and chemometrics" [8], desarrolló un modelo auto adaptativo mediante la combinación de cinco métodos de corrección para la eliminación de la variabilidad biológica, la estrategia de autoselección y la tecnología de búsqueda de modelos. Se busca que el modelo pueda adaptarse automáticamente al cambio de diversas variaciones biológicas en comparación con otros. Para el mismo conjunto de arándanos, cinco modelos de corrección mostraron diferentes rendimientos de predicción y todos lograron una precisión

satisfactoria en comparación con el modelo de variación individual y el modelo de variación híbrida. Los resultados indicaron que la variabilidad biológica tuvo impacto en la predicción y que la corrección de modelos podría mejorar la precisión de la predicción. Para las muestras de arándanos, el modelo más adecuado seleccionado de acuerdo con los resultados adaptativos fue el modelo basado en preprocesamiento. En el estudio de (Wei Zheng, 2020), observamos la ventaja de utilizar el infrarrojo cercano en combinación con arándanos.

1.5. Alcance de la investigación.

En esta investigación se desarrolla un análisis con imágenes VIS-NIR de arándanos, específicamente de la variedad Highbush Legacy. Trabajamos con el infrarrojo cercano (NIR) debido a que la vegetación absorbe la luz visible y refleja la luz infrarroja, en función del estado de salud de la planta [5], por lo tanto se usara como complemento a las imágenes RGB.

Inicialmente se genera un dataset de imágenes de arándanos obtenidas en una plantación ubicada en la ciudad de Quillón, región del Ñuble, el objetivo principal es realizar un análisis en diferentes etapas de maduración del fruto, para generar datos que nos permitan planificar, supervisar y controlar un cultivo eficaz, para poder tomar decisiones a tiempo sobre el desarrollo de la producción y una estimación de la mejor etapa de cosecha, dado que las fechas de cosecha eran estimadas por los mismos agricultores.

Finalmente con el dataset adquirido se entrenan tres algoritmos, dos versiones de YOLO, las cuales son YOLO en su quinta versión y YOLOR, si bien YOLO tiene dificultades para detectar objetos pequeños, queremos ver que tan grande son sus limitaciones y como es su desempeño comparado con MASK R-CNN, que es el otro sistema de detección que pondremos a prueba, el cual ha logrado los mejores resultados en las tres competiciones de la serie de desafíos COCO, incluida la segmentación de instancias y la detección de objetos de cuadros, todo esto con el objetivo de verificar cual tiene el mejor desempeño y cual es el más robusto para esta aplicación.

1.6. Objetivos

1.6.1. *Objetivo General*

- Identificar y contar arándanos por medio de imágenes multiespectrales y algoritmos de Deep Learning.

1.6.2. *Objetivos Específicos*

- Generar y caracterizar los arándanos desde el punto de vista de imágenes en espectro visible e infrarrojo.
- Adaptar tres algoritmos de Deep Learning para la identificación de arándanos.
- Evaluar el desempeño de los algoritmos en términos de velocidad de procesamiento y métricas de clasificación.

1.7. Metodología

Para la metodología se generó un diagrama con las etapas a desarrollar. Inicialmente, se consideró un estudio en profundidad del tratamiento y cosecha del arándano (contextualización), para luego introducir conceptos claves, selección de algoritmos, generar la data a utilizar (etiquetado y aumento de datos), entrenar los algoritmos seleccionados para finalmente generar métricas de desempeño y un análisis general.

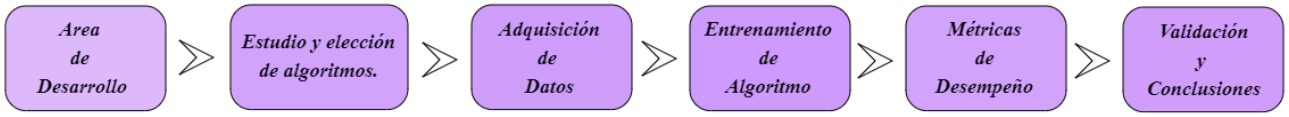


Figura 6: Etapas a desarrollar.
Fuente: Elaboración propia.

1.7.1. Área de desarrollo

Definición y comprensión del área de desarrollo, descripción del proceso desde la fisiología hasta la cosecha y postcosecha del arándano.

- Generalidades del Arándano.
- Manejo de cosecha y postcosecha.

1.7.2. Conceptos claves

Definición y análisis de conceptos claves, para interiorizar de mejor forma los tópicos base que serán de utilidad para el cumplimiento de los objetivos propuestos.

1.7.3. Adquisición de datos

Análisis y etiquetado de imágenes, tratamiento de dataset.

1.7.4. Estudio de algoritmos disponibles

Análisis de algoritmos usados en estudios anteriores y posibles a implementar.

- YOLO, YOLOR, Mask-RCNN, SegNet, Single Shot Detector, RetinaNet, Spinet y DETR.

1.7.5. Entrenamiento de algoritmo

1.7.6. Métricas de desempeño

Estudio y análisis de métricas de desempeño, para posterior uso en evaluación de algoritmos implementados.

CAPÍTULO 2. MARCO TEÓRICO

2.1. Área de desarrollo

Dada la creciente demanda de productos de calidad, seguros y nutritivos, actualmente se buscan formas de conservar de la mejor manera las características nutricionales de la fruta. Es por ello que la industria agrícola tiene como objetivo desarrollar nuevas formas de análisis y métodos de conservación para los alimentos.

2.1.1. Generalidades del Arándano

El arándano proviene de la familia Ericaceae y al genero Vaccinium. Es un arbusto de hoja caduca, mide de 0.2 a 0.4 m de altura y el fruto tiene forma de baya esférica de tamaño referencial de 7 a 15 mm, el color característico de madurez es de color azul y posee un sabor agridulce.
Resiste de gran manera los fríos de invierno, llegando a soportar temperaturas inferiores a -30 °C. Aunque cada variedad de arándanos tiene requerimientos y características diferentes.
Este arbusto requiere de vernalización la cual es la condición natural que poseen a periodos fríos para que se logre la floración o apertura de flores.
El arándano requiere de una cierta cantidad de horas de frío para inducir la floración y lograr una buena cantidad de fruto. Pero aunque el arándano resista y necesite de estas temperaturas requiere al menos de 140 a 160 días sin recibir heladas [9].
Además de los requerimientos antes mencionados, se necesita una preparación y elección del suelo especial preferentemente ácido, ligero, con gran porcentaje de materia orgánica, buen drenaje y bajo en minerales como Fósforo y Calcio.
Estos nutrientes están dados por hectárea y se asocian a la necesidad nutricional.

$$Dosis\,de\,nutriente\,[\frac{kg}{ha}] = R \cdot f[\frac{tfruta}{ha}] \cdot [\frac{kg\,nutriente}{tfruta}] \tag{1}$$

Donde R es el rendimiento esperado y f el factor de dosis.
El arbusto y fruto mostrado en la figura 7 es de la variedad Duke (Nothern Highbush) y muestra el estado de madurez de estos. Esta variedad se caracteriza por sus ramificaciones desde el suelo y se considera una variedad productiva, de floración tardía y frutos firmes. Provee de las mejores post cosechas de todas las variedades.

El arbusto de arándano en la etapa de floración se vuelve muy sensible a bajas temperaturas, lo que requiere una mayor atención. Dado que esta etapa define el crecimiento y cantidad de frutos por arbusto.

Figura 7: Arbusto y fruto de arándano.
Fuente: Boletín INIA/N° 371, 2017.

Por esto mismo, se deben considerar datos anteriores de temperaturas bajas (heladas) en la área de cultivo. Se deben considerar una cantidad de horas frío que van desde 400 a 1200 horas. La temperatura mas adecuada de crecimiento de brotes, hojas y frutos esta entre 20 °C y 26 °C [3].

Figura 8: Flores y arbusto de arándano
Fuente: Boletín INIA/N° 317, 2017.

2.1.2. Beneficios del arándano

Alto valor nutricional
La FDA (Food and Drug Administration) de Estados Unidos lo resume como entre bajo y libre de grasas y sodio, libre de colesterol y rico en fibras, refrescante, tónico, astringente, diurético y poseedor de vitamina C y vitamina K. El color de los arándanos es causado por un grupo de flavonoides llamados antocianina, que tienen un alto poder antioxidante [10].

Buena fuente de antioxidantes
Debido a las propiedades antioxidantes y antiinflamatorias de los arándanos, éstos son benéficos para la salud del corazón y lo protegen; por ejemplo ayuda a prevenir las enfermedades cardiovasculares y el padecimiento de ataques al corazón. Asimismo, han demostrado reducir el colesterol, la presión arterial, proteger contra el accidente cerebrovascular y reducir el estrés oxidativo[10].

Ayuda al cerebro
Esta fruta es rica en flavonoides, lo que hace potenciar la memoria y mejorar el aprendizaje y otras funciones cognitivas. Por otro lado, protegen al cerebro de radicales libres, los cuales son dañinos, ya que puedan dañar tejido sano y están relacionados con la pérdida de memoria. Asimismo, esta fruta ayuda a reducir el riesgo de tener ciertas enfermedades como Parkinson o Alzheimer[10].

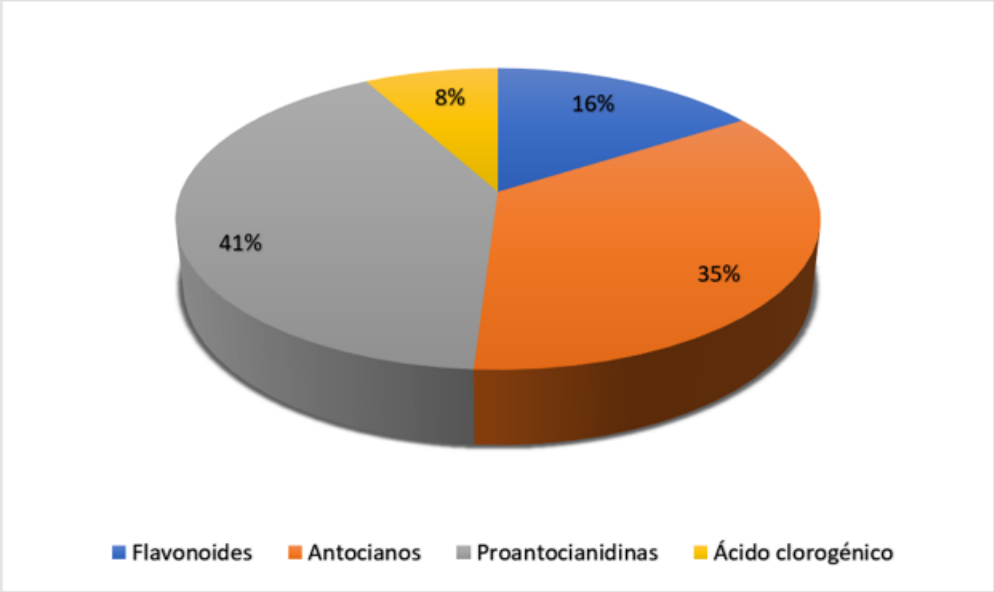


Figura 9: Composición de los distintos grupos de fenoles en el arándano.
Fuente: Elaboración en base a [11].

Información nutricional del arándano en 142 gr			
Calorías	100 Kcal	**Zinc**	0.16 mg
Proteínas	0.97 gr	**Cobre**	0.09 mg
Grasas	1 gr	**Manganeso**	0.41 mg
Carbohidratos	20.50 gr	**Vitamina C**	18.90 mg
Fibra	3 gr	**Tiamina**	0.07 mg
Calcio	9 mg	**Riboflamina**	0.07 mg
Hierro	0.24 mg	**Niacina**	0.52 mg
Magnesio	7 mg	**A. Pantoténico**	0.13 mg
Fósforo	15 mg	**Vitamina B6**	0.05 mg
Potasio	129 mg	**Folacina**	9.30 mg
Sodio	9 mg	**Vitamina A**	145 IU

Tabla 2: Información nutricional del árandano.
Fuente: Northeast Regional Agriculture Engineering S. y Produce Marketing Association's Labeling Facts, USA, 2019.

Se resume el arándano como un alimento bajo y libre de grasas y Sodio, libre de colesterol, rico en fibras y vitamina C. Considerado en el ámbito científico como poderoso antioxidante y que protege el organismo de riesgos cardiovasculares.

Si analizamos la información nutricional al consumir 142 gr de arándano, aporta a la dieta alimentaria diaria basada en 2000 calorías/día, 1 porción representa el aporte de 1 % de grasas, 9 % de carbohidratos, 14 % de fibra, 15 % de vitamina C, bajo aporte de carbohidratos y calorías, lo que lo considera un alimento hipocalórico apto para dietas hiposódicas. Con amplias virtudes y características definidas anteriormente.

2.1.3. Tipo de arándano

De las especies cultivadas, la de mayor importancia es el arándano Alto (Highbush), que representa más del 80 % de las especies cultivadas. Le sigue la especie Ojo de Conejo (Rabbiteye), con una proporción de alrededor del 14 %. El arándano Alto fue la especie que primero se introdujo a cultivo. Es una planta originaria de la costa este de América del Norte y que bajo condiciones de cultivo puede alcanzar alturas de hasta 2,5 m. Debido al largo proceso de mejoramiento al que esta especie ha sido sometida (inició en 1906), y es el tipo de arándano que muestra la fruta de mejor calidad en cuanto a tamaño y sabor.

- Southern highbush: **Legacy**

Los cultivares del tipo **Southern Highbush** tienen un menor requerimiento de frío invernal y más tolerancia al calor. En general muestran un bajo vigor y una alta mortalidad, por lo que son difíciles de cultivar. Presentan, a su vez, problemas graves cuando las plantas se cultivan en suelos con limitaciones texturales y profundidad efectiva de arraigamiento. Son de calibre más alto y, por lo tanto, su orientación es al mercado fresco con cosecha manual. Estos cultivares han sido desarrollados a partir de hibridación interespecífica entre arándano Alto (V. corymbosum) y dos especies nativas del sudeste de Norteamérica: un arándano siempre verde (V. darrowi) y el arándano Ojo de Conejo o Rabbiteye (V. ashei). Estas variedades híbridas requieren entre 200 y 600 horas bajo 7 ºC [3].

Figura 10: Planta adulta arándano Southern Highbush, variedad Legacy
Fuente: Elaboración propia.

La variedad de arándano **Legacy** que tiene requerimientos de frío entre 500 a 600 horas aproximadamente. Los frutos son de medianos a grandes, firmes y de buen sabor, con una marcada cicatriz del pedúnculo. Es una variedad bien catalogada, producto de su alta producción. El arbusto mantiene sus hojas en invierno. Existen huertos que llegan a los 18 a 20 t/ha al $4^o - 5^o$ año. Presenta una floración temprana y larga, lo que la hace propensa a hongos de flor. La fruta puede presentar partiduras con precipitaciones abundantes y se adapta a la mayoría de las zonas productivas.
Su fecha de cosecha es intermedia y se exporta a todos los mercados, lo que mejora sus expectativas comerciales. Entre otras características, se adapta a la cosecha mecánica [3].

Figura 11: Fruto en distintas etapas de maduración.
Fuente: Elaboración propia.

2.1.4. Manejo de cosecha

Dado el tamaño de la fruta, los arándanos tienden a sufrir mayor pérdida de agua que otras frutas y mantener la cutícula ayuda a reducir está pérdida.

- Calidad: Está definida por distintas variables y se puede agrupar en calidad visible que se refiere a las características físicas del fruto, su color, daño mecánico y pudriciones presentes, forma, tamaño y firmeza.

  La calidad organoléptica se define por índices internos de la fruta que definen el contenido de azucares, ácidos y compuestos relacionados al aroma del arándano.

  Y calidad nutritiva que provee la información nutricional, es decir vitaminas presentes en el fruto. Por lo que se espera obtener las características ideales para cumplir con los parámetros antes mencionados. Y la preclásica y postcosecha están orientadas a maximizar la cantidad de frutos de calidad.

- Madurez: Es la característica de mayor influencia en la cosecha. Esta define cuando se realiza cada etapa de cosecha. La fecha mas adecuada se define por este factor y esta asociada al color del fruto.
  Un color azul uniforme indica una buena calidad. Si por el contrario, la cosecha se llevara a cabo cuando el color sea aún rojo, es decir, la maduración esté en una etapa media-baja, se obtendrá un fruto con mayor firmeza pero tendrá una calidad organoléptica inferior [3].

2.1.5. Manejo de post cosecha

- Manejo de temperatura y humedad relativa: De los puntos más importantes para la prolongación de la vida del fruto postcosecha es la temperatura, que debe ser controlada de diferentes formas. Una de estas es cubriendo las bandejas de recolección con materiales reflejantes (ver figura 12). La temperatura incide en el metabolismo de la fruta y la duración de ésta. Y es ideal luego de la cosecha llegar a una temperatura entre 0 y 1 ^{o}C, que es la recomendada para su posterior almacenamiento y transporte. Es muy importante mantener la cadena de frío.
  Como se mencionó en la sección anterior los arándanos son muy susceptibles a la pérdida de agua, lo que se describe físicamente con el arrugamiento de la fruta. Por esto mismo mantener la temperatura y humedad es crucial para evitar el déficit de presión y deshidratación. La humedad relativa indicada a 0 ^{o}C es de 95 %. Manteniendo la humedad relativa entre el 90 y 95 % a 0 ^{o}C los arándanos tienen una duración mínima de 14 días[12], [13].

Figura 12: A) Etapa de recolección, B-C) Traslado de fruto y protección del sol.
Fuente: Manual de manejo agronómico de los arándanos [3].

- Uso de atmósferas controladas y modificadas: Manteniendo la temperatura y humedad mencionadas anteriormente. Se utilizan variadas formas para controlar el lugar y ambiente donde se almacenan. Estas tecnologías son: atmósfera modificada (AM) y atmósfera controlada (AC), las que consideran la modificación de la composición de gases en el almacenamiento y transporte. Estas tecnologías reducen potencialmente la deshidratación y disminuyen la probabilidad de desarrollar algún tipo de pudrición [12].
- Incidencia de pudriciones: Múltiples patógenos afectan a los arándanos. Aunque estos puedan ser reducidos o manejados no pueden ser eliminados en su totalidad en la etapa de postcosecha. Por lo que se deben aplicar fungicidas en los momentos críticos de la precosecha.

2.2. Conceptos claves

2.2.1. Python

Python es un lenguaje de programación, orientado a objetos de alto nivel y con semántica dinámica, principalmente utilizado para el desarrollo web y de aplicaciones informáticas. Su sintaxis hace el código muy legible, por lo tanto, facilita su depuración.
Python usa módulos y paquetes, lo que permite que los programas pueden ser diseñados en un estilo modular y el código se puede reutilizar en varios proyectos. Una vez desarrollado un módulo, se puede extrapolar para su uso en otros proyectos, y es fácil de importar o exportar [14].
A pesar de que Python fue creado como un lenguaje de uso general, posee librerías y entornos de desarrollo para cada una de las fases del proceso de Data Science, además su opción multiplataforma, su potencia, su carácter open source y su facilidad de aprendizaje, hacen a Python unos de los mejores lenguajes de programación de la actualidad.

2.2.2. Agricultura de precisión

La agricultura de precisión ya lleva un par de décadas en chile, en ese entonces nadie se imaginaba los avances que tendría en la producción agrícola. Cuando se habla de Agricultura de precisión (AP), se habla de optimizar la calidad y cantidad de un producto agrícola, minimizando costos a través del uso de tecnologías más eficientes [15].

Esta tecnología es considerada como la práctica de utilizar satélites, redes celulares, GPS, sensores, dispositivos IoT, sistemas de control, automatización, entre otras innovaciones tecnológicas y de comunicación, en estos últimos años la agricultura de precisión ha ido en alza debido al IoT, y de los dispositivos inteligentes, si esto lo mezclamos con IA y visión artificial se pretende que el desarrollo aumente al igual que la innovación.

2.2.3. Inteligencia artificial

La **inteligencia artificial** es una subdisciplina del campo de la informática, que busca la creación de máquinas que pueden imitar comportamientos inteligentes. Esta subdisciplina contiene ramas como el Machine Learning y Deep Learning. Coloquialmente, el término *Inteligencia Artificial* se aplica cuando una máquina imita funciones "*cognitivas*" que los humanos asocian con otras mentes humanas, como el "*aprendizaje*" y la "*resolución* de *problemas*" [16]

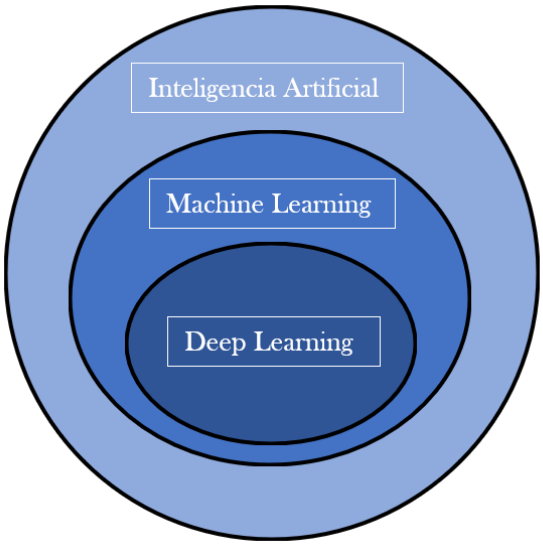


Figura 13: Diagrama global de ramas de inteligencia artificial.
Fuente: Elaboración propia en base a [16].

- Inteligencia artificial en la agricultura.

Básicamente lo que haremos en este proyecto es la detección de objetos en imágenes, que actualmente esta viviendo su mejor momento gracias a nuevas arquitecturas de redes neuronales, que están utilizando la disciplina conocida como Deep Learning.

La detección de objetos es un área de la visión por computador que a su vez pertenece al conjunto de la inteligencia artificial, esta área busca el estudio y la creación de programas o máquinas que puedan aprender en base a datos de entrada con el fin de resolver problemas, reproduciendo conductas inteligentes ligadas al ser humano [17].

La detección y clasificación de frutas mediante el procesamiento digital de imágenes es un campo muy amplio debido a la variabilidad de formas de aplicar esta tecnología, por ejemplo, la clasificación por tamaño, colores, texturas, etc.

La Visión por Computador y el Aprendizaje Automático son de las áreas mas importantes de la Inteligencia Artificial y son las que utilizaremos en este proyecto, el objetivo de la Visión por Computador es brindar a los computadores la habilidad de poder “ver” y “comprender” la información contenida dentro de fotogramas, para lograr así la automatización de distintas tareas que los humanos podríamos realizar gracias a nuestro sistema visual [18].

En la agricultura, la Visión por Computador está jugando un papel muy importante a la hora de minimizar los costos, ya que, el alto costo de la mano de obra, el aumento de los estándares de calidad y una mayor competencia global, han motivado a los agricultores a adoptar estas tecnologías con el objetivo de mejorar la producción de los cultivos para poder adaptarse a los estándares del mercado actual.

2.2.4. *Machine Learning*

Machine Learning o Aprendizaje Automático el subcampo de la Ciencia de la Computación que según Arthur Samuel en 1959 "da a las computadoras la habilidad de aprender sin ser programadas explícitamente". Por ejemplo, si se quiere diferenciar a un perro de un gato, lo primero es interpretar un conjunto de imágenes que poseen un conjunto de características especificas. Como los ojos del animal, su tamaño, el número de patas, orejas, cola, entre otros. Antes de Machine Learning se requería escribir un set de reglas con condiciones a cumplir para identificar distintas características que podrían definir el animal. Esto fue poco factible dado que se requerirían muchas reglas, lo que con Machine Learning es posible, permitiendo construir un modelo que considere y observe todos los conjuntos de características, su correspondiente tipo de animal y aprender los patrones de cada uno de estos. Lo que se logra sin la programación explicita de un modelo, mas bien es comprensión, entendimiento y diferenciación de este mismo [19].

Tipos de aprendizaje:

- Aprendizaje supervisado: Este tipo de aprendizaje se entrena utilizando ejemplos etiquetados, como una entrada en la que se conoce la salida deseada. Una técnica de este tipo recibe un conjunto de datos de entrada asociadas a salidas específicas, el algoritmo aprende de la salida que se obtiene con la deseada mediante patrones y encuentra errores, para modificar el modelo.

- Aprendizaje no supervisado: Este tipo de aprendizaje se entrena utilizando datos sin etiqueta, por lo que una técnica de este tipo debe averiguar lo que se muestra. El objetivo es explorar los datos y encontrar alguna estructura en ellos. El aprendizaje no supervisado se utiliza, por ejemplo para segmentar temas de texto, recomendar elementos e identificar datos atípicos.

- Aprendizaje semi-supervisado: Este tipo de aprendizaje utiliza las mismas técnicas que el aprendizaje supervisado, pero utiliza datos etiquetados y no etiquetados. Donde la mayoría de estos datos no son etiquetados.

- Aprendizaje por refuerzo: Este tipo de aprendizaje se basa en la prueba y error para obtener la mejor respuesta. Este tipo de aprendizaje tiene tres componentes principales: el agente (aprende y decide), el entorno (Lo que interactúa con el agente) y las acciones (lo

que el agente puede hacer). El objetivo es que el agente elija las acciones que maximicen la recompensa esperada en un tiempo determinado [19].

Algunos tareas de aprendizaje supervisado son:

- Regresión/Estimación: Predicción de valores continuos. Se logran estimaciones con datos o características conocidas, como el valor de una casa basada en características especificas o el valor de meses anteriores.
- Clasificación: Usado para predecir la clase o categoría de una muestra. Dado un set de datos que permita identificar distintas clases [16].

Algunas tareas de aprendizaje no supervisado son:

- Agrupación (Clustering): Usado para encontrar casos similares, basados en la estructura de los datos. Como en el caso de segmentación de clientes.
- Asociación: Usado para buscar elementos o eventos que suceden de manera conjunta o concurrentemente. Como la compra de comida en conjunto con una bebida particular.

2.2.5. *Red neuronal convolucional*

Una red neuronal convolucional mas conocida por su acrónimo en ingles CNN, es un algoritmo de Deep Learning, conformado por tres secciones principales (entrada, capas ocultas (aprendizaje de características) y salida (clasificación)).
Toma una imagen de entrada y pasa a las capas de aprendizaje donde se asignan pesos para optimizar el análisis e identificación de cada categoría presente en la imagen de entrada. Para lograr una diferenciación entre cada sección y finalmente lograr la clasificación en la capa de salida.
Una red neuronal se inspira en los modelos biológicos que componen el cerebro y estos se describen mediante modelos matemáticos con un elevado numero de componentes y elementos organizados en niveles y/o capas.

Una CNN esta compuesta por:

- Kernel (núcleo)
  El núcleo es una matriz que se usa como filtro para extraer las características de cada imagen, esta matriz se desplaza sobre los datos de entrada, realiza un producto de puntos con diferentes subregiones de los datos de entrada y obtiene una matriz de productos (producto escalar) de píxeles de salida. El kernel avanza a través de los datos a razón de un valor específico, generalmente este valor es 2 (se mueve 2 columnas por píxel de la matriz de entrada), permitiéndole extraer características en cada sección de la imagen y formar una nueva matriz de salida que sera una nueva capa de neuronas ocultas.

- Capa convolucional 2D
  Esta capa utiliza una matriz (kernel) convolucional , considerado como un filtro que ayuda a generar un tensor de salida con características definidas.

- Max pooling (Agrupación máxima)
  Es un proceso de discretización basado en muestras, consiste en disminuir las muestras de una entrada, reduciendo su dimensionalidad y realizar inferencias en las subregiones agrupadas.

- Funciones de activación
  Son ecuaciones matemáticas que determinan la salida de una red neuronal, decide cuando se activa cada una de las neuronas de la red. Estas funciones pude ser ReLU, Mish, lineal, sigmoide, escalón, gaussiana, entre otras. En el caso de YOLOv4 se utiliza la función de activación Mish que se muestra en la siguiente figura con otras funciones de activación, las que poseen un umbral diferente entre cada una. Y para el caso de YOLOv5 se utiliza una función de activación modificada de ReLU. YOLOv4 y YOLOv5 se detallan en la sección 2.4.

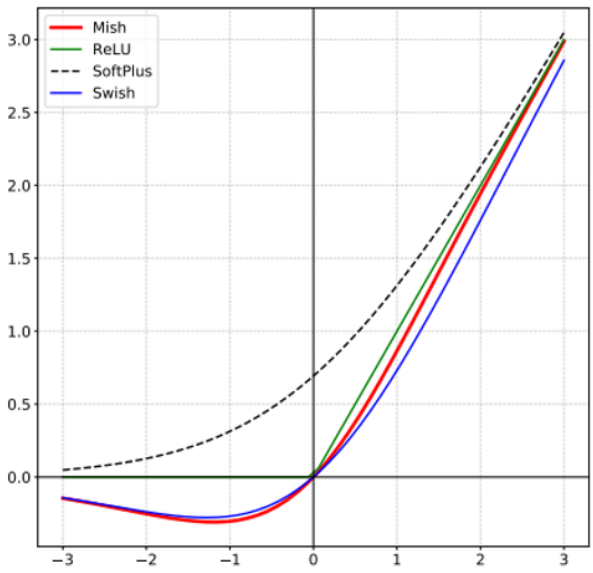


Figura 14: Gráfica de las funciones de activación.
Fuente: Mish: A Self Regularized Non-Monotonic Activation Function (2019)[20].

- Optimizador
  Algoritmo encargado de actualizar los valores de los pesos para mejorar el desempeño y funcionamiento de la red, se basa en calcular la derivada parcial por cada peso de la red. Los principales optimizadores son: SGD (Stochastic Gradient Descent), RMSprop (Root Mean Square Propagation), Adam (Adaptative moment estimation) y otros.

2.2.6. *Deep Learning*

Deep Learning o Aprendizaje Profundo es el estudio de redes neuronales artificiales y algoritmos de aprendizaje automático que contienen más de una capa oculta. Forma parte de la rama de Machine Learning basado en el aprendizaje de representaciones de datos. Surge dado que los algoritmos simples de Machine Learning no han logrado resolver los problemas centrales de la IA, como el reconocimiento del habla o el reconocimiento de objetos. Al añadir más capas y más unidades dentro de una capa, una red profunda puede representar funciones de complejidad creciente [21].
Se utiliza una cascada de muchas capas de unidades para la extracción y transformación de características. Cada capa sucesiva utiliza la salida de la capa anterior como entrada. Los algoritmos pueden ser supervisados o no supervisados y sus aplicaciones incluyen el análisis de patrones (no supervisado) y la clasificación (supervisada) [16]. Se basa en múltiples niveles donde, las características de nivel superior se derivan de las características del nivel inferior para formar una representación jerárquica.

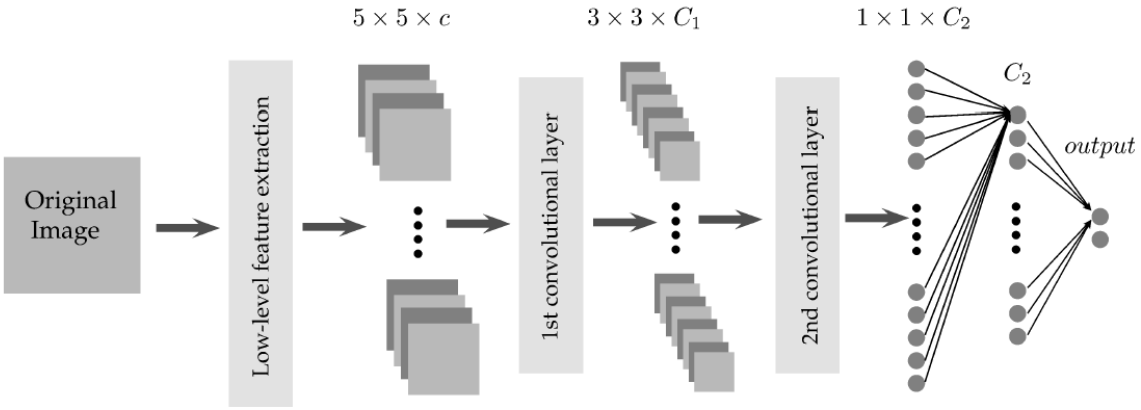
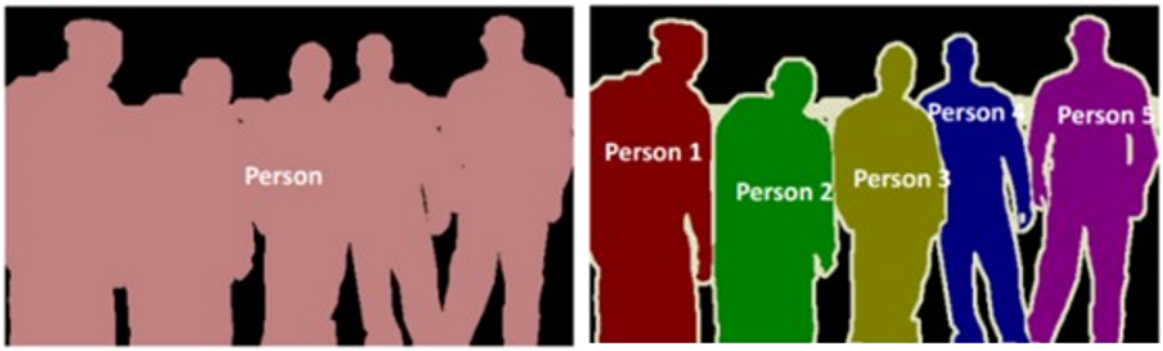


Figura 15: Modelo de arquitectura de red convolucional de aprendizaje profundo.
Fuente: Estudio de inspección [22].

2.2.7. *Detección de objetos*

En la segmentación de imágenes podemos dividir la imagen en varias partes llamadas segmentos, podemos hacer uso de estos segmentos para procesar la imagen.
La diferencia entre la detección de objetos y la segmentación de imágenes es que la detección de objetos crea un cuadro que delimita a cada clase de la imagen, pero no se sabe la forma del objeto, solo las coordenadas del cuadro delimitador, mientras que la segmentación de imágenes crea una máscara de píxeles para cada objeto de la imagen.



((a)) Segmentación semántica ((b)) Segmentación de instancias

Figura 16: Segmentación: a) Semántica, b) De instancias.
Fuente: Estudio de inspección [22].

En la figura 16 imagen (a), tenemos un caso de segmentación semántica, cada píxel es parte de una clase en particular, podemos observar que todos los píxeles son parte de una sola clase y están representados por el mismo color, en la figura 16 imagen (b), tenemos segmentación de instancias, también se asigna una clase particular a cada píxel de la imagen, pero los diferentes objetos de una misma clase tienen colores diferentes.
Un buen escenario para utilizar este modelo es cuando el objeto y el fondo tienen un alto contraste, en estas situaciones el modelo funciona muy bien, pero cuando hay una diferencia significativa en la escala de grises, es complicado obtener segmentos precisos.

2.2.8. *Imágenes Multiespectrales*

En los últimos años, la adquisición y el análisis de datos multiespectrales están adquiriendo un interés e importancia cada vez mayores en la agricultura.
Las imágenes multiespectrales son aquellas que además de la información contenida en los colores visibles del espectro, aporta información en otras bandas fuera de los umbrales visibles. Estas imágenes se obtienen con instrumentos especiales dedicados a tales tareas.
Las imágenes multiespectrales entregan información del espectro visibles más otras bandas cuyo número no supera las quince, la mayoría está en el infrarrojo.

En la agricultura de precisión la banda más utilizada es la del infrarrojo cercano, que comprende las longitudes desde 750-900 nm, por el hecho de que la vegetación absorbe la luz visible y refleja la luz infrarroja en función del estado de salud de la planta [23].

Figura 17: Ejemplo de imagen en el infrarrojo cercano e imagen RGB.
Fuente: Elaboración propia.

Ventajas de las imágenes multiespectrales [24][25][26]:

- Planificar, supervisar y controlar los cultivos eficazmente.
- Tomar decisiones a tiempo sobre el desarrollo de los sistemas de producción.
- Identificar obstáculos y facilitar el diagnostico en áreas para obtener posibles puntos de conflicto en el desarrollo del cultivo y que, aparentemente, parecían favorables.
- Detectar el buen o el mal estado en ciertas cosechas y la presencia de plagas o enfermedades en los diferentes cultivos.
- Extraer información de los cultivos que el ojo humano no puede capturar.

En este proyecto utilizaremos las imágenes multiespectrales para poder obtener mayor precisión al momento de clasificar y detectar arándanos y aplicable a otro tipo de cultivos, ya que, tendremos información adicional que no tendríamos con el espectro visible.

2.2.9. *Librerías utilizadas*

Keras

Keras es una biblioteca de código abierto para crear redes neuronales.

La implementación de las redes neuronales se simplifica notablemente con la biblioteca Keras. Keras es una biblioteca de código abierto (con licencia MIT) escrita en Python, el objetivo de esta biblioteca es acelerar la creación de redes neuronales, esto se logra porque Keras no funciona como un framework independiente, más bien funciona como una interfaz de uso intuitivo (API) que permite acceder a varios frameworks de aprendizaje automático. Entre los frameworks compatibles con Keras están TensorFlow, Theano, Microsoft Cognitive y Toolkit.

Keras proporciona bloques modulares en los cuales se desarrollan complejos modelos de aprendizaje profundo [27]. Por lo tanto, es una buena herramienta para crear redes neuronales, ya que simplifica este proceso. Otra ventaja es que los modelos desarrollados con Keras son fáciles de utilizar en diferentes plataformas, por ejemplo, es compatible con Android (Keras TensorFlow Android Runtime), iOS (Apple CoreML), Google Cloud y Raspberry Pi por defecto. Es un buen soporte para múltiples GPU, con Keras los procesos de aprendizaje profundo se pueden distribuir en varios chips o tarjetas gráficas. Además, cuenta con el mantenimiento y desarrollo de grandes empresas como Google, Nvidia, Apple, Microsoft y Amazon entre otras.

Netflix, Uber, como también grandes organizaciones como la NASA o la CERN, confían en Keras para llevar a cabo sus proyectos, pero esto no es sorpresa, ya que, gracias a su licencia gratuita y al concepto multiplataforma la biblioteca registraba mas de 250.000 usuarios individuales a mediados del 2018, y hoy en día son muchos mas, gracias a la incorporación al software TensorFlow de Keras.

Se puede acceder al proyecto completo de Keras a través del directorio oficial de GitHub de Keras.

TensorFlow

TensorFlow es una biblioteca de código abierto para la computación numérica y Machine Learning a gran escala, creado por el equipo de Google Brain, esta biblioteca reúne distintos modelos y algoritmos de Machine Learning y Deep Learning y los hace trabajar en forma conjunta.

Esta biblioteca permite a los desarrolladores crear grafos de flujo de datos, es decir, estructuras que describen como los datos se mueven a través de un grafo, o una serie de nodos de procesamiento. Y cada nodo del grafo representa una operación matemática, y cada conexión o aristas entre nodos es una matriz de datos multidimensionales o tensor [28], es decir, es capaz

de relacionar varios datos en red de forma simultánea, imitando lo que hace el cerebro humano. Todos esto TensorFlow lo proporciona a través del lenguaje de Python. Sin embargo, las operaciones matemáticas reales no se realizan en Python, si no que están disponibles a través de TensorFlow escritas como binarios C++ de alto rendimiento. Python básicamente solo dirige el tráfico entre las piezas, y proporciona abstracciones de alto rendimiento para conectarlas entre si.

TensorFlow se puede usar en diversas aplicaciones como entrenamiento y ejecución de redes neuronales profundas para el reconocimiento de imágenes, clasificación de dígitos escritos a mano, las redes neuronales recurrentes, el procesamiento del lenguaje natural y simulaciones basadas en ecuaciones diferenciales parciales.

Hay alternativas a TensorFlow como puede ser PyTorch, que tiene muchas similitudes con TensorFlow, además también está construido con Python, pero elegimos TensorFlow porque PyTorch es una mejor opción para el desarrollo de proyectos rápidos, en los que el tiempo es un factor, mientras que TensorFlow es para proyectos más grandes y flujos de trabajo más complejos.

Google, Intel, Bloomberg, DeepMind, GE Health Care, eBay, entre otros utilizan el software TensorFlow en sus investigaciones, puede ser debido a que es un software libre, podemos editarlo en función de nuestras necesidades específicas, además su perfeccionamiento a través de la licencia OpenSource ha permitido su uso masivo, otra ventaja es que constantemente se esta actualizando y se pronostica un rápido crecimiento en los próximos años.

Pandas

Pandas es una librería de código abierto desarrollada en Python, que permite manipular de manera similar a los dataframes de R. Es especializada en el manejo y análisis de datos. Pandas es una extensión de Numpy [29].

Pandas nace como la necesidad de juntar en una única librería todo lo necesario para que un analista de datos pueda tener en una sola herramienta todas las funcionalidades que necesita como; cargar datos, analizar, modelar y manipular.

Pandas proporciona herramientas que permiten:

- Transformar datos aplicando funciones tanto sobre el conjunto de datos completo como en porciones o ventanas de éste.
- Manipulación de series temporales.
- Hacer gráficas.

- Leer y escribir datos en diferentes formatos: CSV, Microsoft Excel, bases SQL y formato HDF5.
- Seleccionar y filtrar de manera sencilla tablas de datos en función de posición, valor o etiquetas.
- Fusionar y unir datos.

En pandas existen tres tipos básicos de objetos todos ellos basados a su vez en Numpy:

- Series (listas, 1D).
- DataFrame (tablas, 2D).
- Panels (tablas 3D).

Como podemos ver, se trata de una herramienta realmente eficaz con multiplicidad de usos, lo que la convierte en excelente para el tratamiento de datos.

Scikit-learn

Scikit-Learn es una librería de código abierto de Python para hacer análisis predictivo. Cuenta con algoritmos de clasificación, regresión, clustering y reducción de dimensionalidad. Además, es compatible con otras librerías como NumPy, SciPy y matplotlib [30].

La mayoría de los algoritmos de Machine Learning de aprendizaje supervisado forman parte de scikit-learn, como regresión lineal, máquina de vectores de soporte (SVM), arboles de decisión y hasta los métodos bayesianos.

2.3. *Dataset*

El presente dataset se compone de 370 imágenes capturadas en un campo de arándanos en la ciudad de Quillón (-36.744644,-72.493421), en el mes de Diciembre del 2021 con tres cámaras:

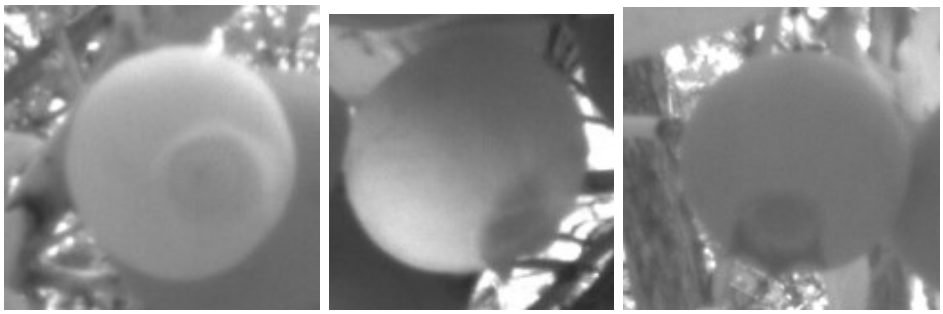
- Nikon Coolpix B700: 250 Imágenes RGB.
- Basler acA2440-20gc: 60 Imágenes RGB.
- Basler acA2000-50gmNIR: 60 Imágenes NIR (Infrarrojo cercano).

Cámara	**Tipo de imagen**	**Formato**	**Cantidad**
Nikon Coolpix B700	RGB	JPG	250 imágenes
Basler acA2440-20gc	RGB	JPG	60 imágenes
Basler acA2000-50gmNIR	NIR	JPG	60 imágenes
		Total	370 imágenes

Tabla 3: Imágenes generadas por cada cámara.

El presente dataset fue creado en condiciones climáticas favorables (día soleado) y cada imagen fue capturada en un entorno semi-controlado, ya que se generó una región de oscuridad suficiente para evitar la mayor cantidad de oclusiones debido a la sombra generada por otra baya. Evitando también el solapamiento entre múltiples bayas. El arándano se presenta en múltiples estados de maduración descritos en la figura 18, las cuales se presentan en todo el dataset, pero con diferentes tamaños, ya que se varía el campo de visión.

((a)) Inmaduro ((b)) Mediana Madurez ((c)) Maduro



((d)) Inmaduro ((e)) Mediana Madurez ((f)) Maduro

Figura 18: Estados de maduración presentes en las imágenes del dataset.
Fuente: Elaboración propia.

Para el dataset se utilizan tres criterios principales:

- Número de instancias por clase: Para cada clase se necesitan al menos 2000 muestras, este número se define de acuerdo a las recomendaciones descritas en la documentación de YOLOv5 (descrito en la sección 2.4).
- Consistencia y precisión de las etiquetas: Cada etiqueta esta definida por cuadro delimitador, por lo tanto, debe ser preciso y de ser posible no debe contener otras clases.
- Imágenes del fondo: El dataset final tendrá entre un 1 y 10 % de imágenes sin ningún objeto, para reducir los falsos positivos con las clases.

Definido el dataset inicial y los criterios, se etiqueta cada imagen con las tres clases definidas (Maduro, Inmaduro y MMaduro), para el caso de las imagenes NIR se utiliza como guía la imagen VIS correspondiente, esto ayuda a reducir las imprecisiones en el etiquetado. Estas etiquetas generan un número de instancias por clase como se muestra en la figura 19 (obtenida de la plataforma Roboflow y descrita en la sección 3.4.1).

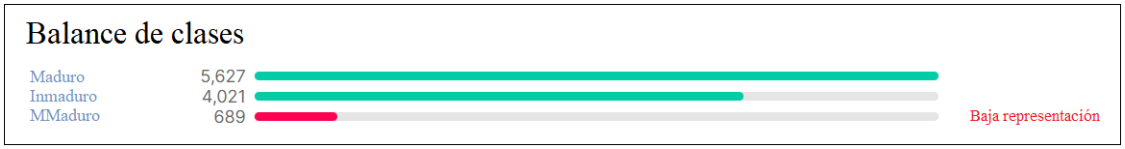
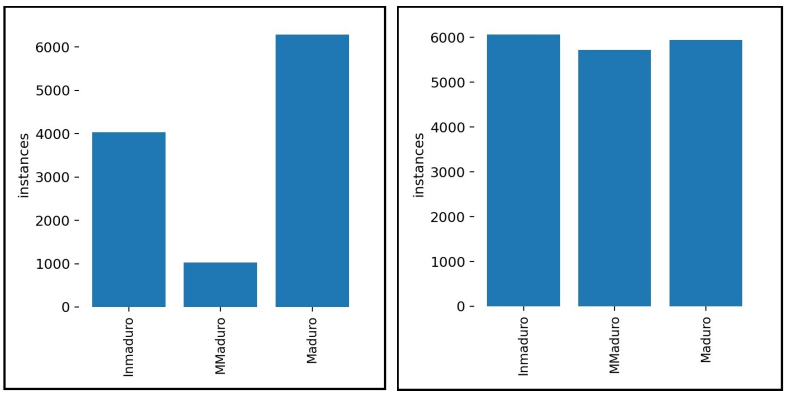


Figura 19: Balance de etiquetas de data original.

El balance de clases muestra como la clase medio maduro tiene un número de etiquetas muy bajo comparado con la clase más cercana (Inmaduro), lo que lo sitúa con una muy baja representación en el dataset.
El primer criterio no se cumple, ya que tenemos menos de 2000 muestras para la clase medio maduro y existe una brecha en el número de instancias por clase. Se generan múltiples imágenes a partir de una sola imagen, con técnicas de aumento de la data. Con la finalidad de cumplir el primer criterio.
El dataset se divide aproximadamente en 80 % de la data (290 imágenes) para la etapa de entrenamiento y el 20 % (80 imágenes) de la data se utiliza para la etapa de validación.

2.3.1. Data augmentation

Data augmentation es una técnica de aumento de datos, utilizada para aumentar la diversidad de su conjunto de entrenamiento mediante la aplicación de transformaciones aleatorias (pero realistas), como la rotación de imágenes, recorte, cambio de brillos y contraste (descrita en la documentación de tensorflow).
Inicialmente la data original capturada en la naturaleza era de 370 imágenes, las que luego de ser etiquetadas, generan un número de instancias específicos por cada clase (4021 para la clase inmaduro, 669 para la clase medio maduro y 5627 para la clase maduro), estas instancias son equivalentes al número de etiquetas realizadas por clase en todo el dataset.
Si comparamos la clase con mayor número de instancias (maduro) con la clase con menor número de instancias (medio maduro) existe una diferencia de 4958 instancias, esta diferencia muestra un desbalance en la data que permitirá tener un entrenamiento más exhaustivo para la clase con mayor número de instancias. Y la clase con menor número de instancias generara más errores y tendrá una precisión inferior.



((a)) Instancias con data original ((b)) Instancias con data generada

Figura 20: Etiquetas generadas con: a) Data original, b) Data aumentada.

Se trabaja cada imagen para lograr un balance entre las tres clases, de tal forma que generamos hasta 7 imágenes por cada una de las imágenes del dataset original, para esto se voltea horizontal y verticalmente, se aplica un zoom de un 10 a 20 % y se recorta en rangos del 10 a 30 %. Además, se generan imágenes auxiliares que solo tienen instancias medio maduras.
Luego se seleccionaron las imágenes con mayor número de instancias por clases específicas, esto ya que algunas imágenes solo tenían etiquetas de una clase específica. Estas imágenes fueron usadas para igualar el número de instancias de la clase inmaduro y medio maduro, como también disminuir la brecha entre el número de instancias por clase.
Se considera un balanceo de la data guiado con ayuda de la plataforma de etiquetado (descrito en la sección 3.4.1), que implementa las técnicas de la librería de Python imgaug que permite recortar, rotar, realizar un zoom, recortes, mosaicos, cambios de exposición y otros.

El dataset final es de 869, con 789 imágenes para la etapa de entrenamiento (considera 96 imágenes de fondo, sin etiqueta) y 80 imágenes para la etapa de validación, llegando a 6000 instancias para la clase inmaduro, 5800 instancias para la clase medio maduro, y 5900 instancias para la clase maduro.
Solo se considera aumento de la data en las imágenes utilizadas en la etapa de entrenamiento.

- Asignación de pesos por clase

Cuando existe un desbalance en el número de instancias por clase, se aborda de diferentes maneras, en nuestro caso generamos un mayor número de instancias con el aumento de la data y agregamos una asignación de pesos por clase en el entrenamiento, que muestrea las imágenes de entrenamiento de acuerdo al mAP obtenido por cada clase, lo que hará que la clase con menor número de instancias tenga mayor relevancia en el entrenamiento.
Como se muestra en el código, primero se calculan los pesos por clase del modelo y luego se le asigna un peso a las imágenes de acuerdo al número de etiquetas asociadas. Permitiendo que el entrenamiento utilice mayor número de imágenes con gran cantidad de instancias medio maduro.

```
# Update image weights
if opt.image_weights:
    cw = model.class_weights.cpu().numpy() * (1 - maps) ** 2 / nc # class weights
    iw = labels_to_image_weights(dataset.labels, nc=nc, class_weights=cw) # image weights
    dataset.indices = random.choices(range(dataset.n), weights=iw, k=dataset.n)
```

2.4. Estudio de algoritmos disponibles

Al momento de elegir un algoritmo es importante tener en cuenta el tipo de aprendizaje, es decir, si será supervisado o no supervisado. De acuerdo con el tipo de aprendizaje se utilizan diferentes algoritmos.

En este trabajo se utilizaron aprendizajes supervisados, estos consisten en entrenar un algoritmo otorgándole preguntas que se denominan características, y las respuestas conocidas como etiquetas, con el fin que las combine y se puedan hacer predicciones.

El aprendizaje supervisado es utilizado generalmente para deducir una función a partir de los datos de entrenamiento, que consisten en partes de objetos dentro de los cuales uno de los componentes son los datos de entrada y el componente restante son los resultados deseados, el objetivo principal es la creación de funciones con la posibilidad de predecir valores correspondientes a objetos de entrada luego de haberse familiarizado con una serie de ejemplos que son los datos de entrenamiento [31].

En este proyecto se implementan tres algoritmos de detección para concluir cuál es el mejor para la clasificación y detección de frutas.

- YOLO: Conocida por su acrónimo Y.O.L.O (You Only Look Once) es sistema de detección que puede identificar rápidamente objetos en imágenes, pero tiene dificultades para localizar con precisión objetos pequeños [32], en este proyecto veremos que tan grande son sus limitaciones y como es su desempeño frente a otros sistemas de detección, para la implementación de YOLO usaremos Tensorflow y Pytorch.

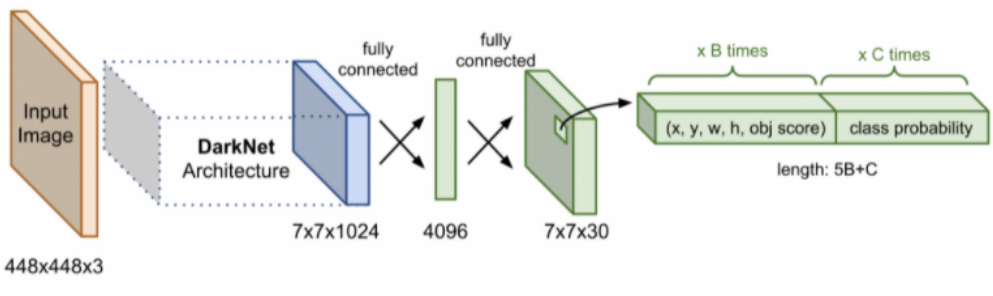


Figura 21: Arquitectura YOLO.
Fuente: Estudio [32].

- YOLOv1

La primera versión de YOLO logro una precisión del 88 % en el conjunto de validación de ImageNet 2012, con el uso del framework Darknet, a partir de esta versión se desarrollaron múltiples versiones (YOLO9000, YOLOv3, YOLOv4, YOLOv5) y derivados de ellas, con el objetivo de lograr una mejora en la precisión para la detección de objetos dentro de una imagen y/o vídeo.

- YOLO9000

Es un algoritmo capaz de detectar 9000 categorías de objetos, conocida como YOLOv2 y por superar a otros métodos como Faster RCNN con Resnet y SSD [33].
YOLO9000 fue preparado para detectar y clasificar objetos, gracias al entrenamiento simultaneo con el conjunto de datos de detección COCO y el conjunto de datos de clasificación ImageNet, permitiéndole detectar en tiempo real múltiples categorías de objetos.
YOLO9000 demuestra la capacidad que tienen técnicas como YOLO, capaces de detectar e identificar una gran cantidad de objetos desde imágenes y/o en tiempo real.

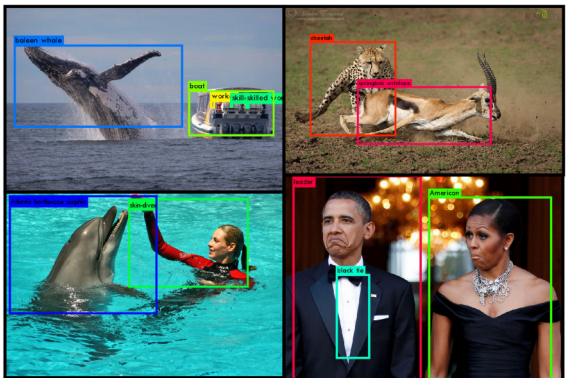


Figura 22: Detección de YOLO 9000 en tiempo real de múltiples clases.
Fuente: YOLOv2 [33].

- YOLOv3

Toma la versión anterior de YOLO y realiza múltiples cambios, como el uso de una nueva red para realizar la etapa de extracción de características, con un enfoque con matices de la utilizada en YOLOv2, Darknet-19 y una nueva red residual, utiliza 53 capas convolucionales denominada Darknet-53 [34].
YOLOv3 predice cajas delimitadoras en 3 escalas diferentes y con el nuevo extractor de características. Siendo más precisa que la versión anterior y manteniendo la rapidez característica de este tipo de técnicas, obteniendo 28.2 mAP (precisión media, descrita en la sección 2.5) en 22 ms, con un tiempo casi tres veces menor que SSD321 (ver figura 23)(SSD se describe en la sección 2.4).

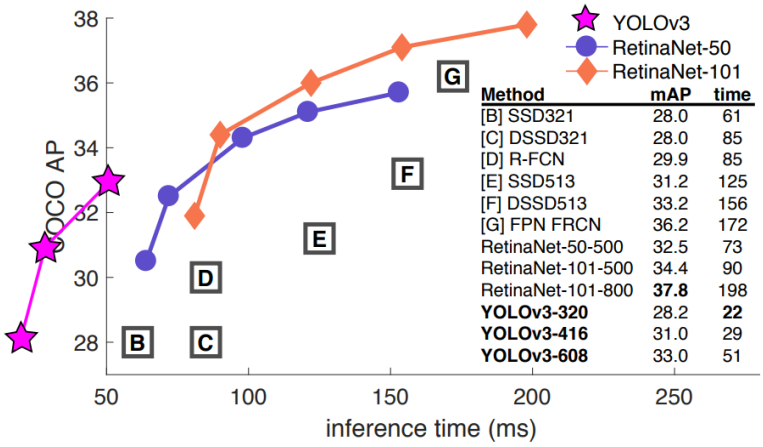


Figura 23: Comparación entre diferentes métodos.
Fuente: YOLOv3 [34].

- YOLOv4

Para lograr una mejora evidente, se crean múltiples etapas que se ejecutan a pequeña escala para problemas específicos y otras características como conexiones parciales en etapas cruzadas (WRC), la normalización por lotes, activación Mish y otras, se combinan para conseguir resultados de ultima generación como: 43.5 % AP (65,7 % AP50) para el conjunto de datos MS COCO (Microsoft Common Objects in Context) a una velocidad en tiempo real de aproximadamente 65 fps, mejorando en un 10 % su versión anterior [35]. MS COCO es un conjunto de datos de detección, segmentación, detección de puntos clave y subtítulos de objetos a gran escala. Este conjunto de datos consta de 328K imágenes.
Con estos resultados se cumplen los objetivos principales de la investigación y trabajo, el cual se centra principalmente en generar un detector de objetos eficiente y con una rápida detección. En este trabajo se menciona un concepto denominado **Bag of freebies**, el cual se asocia a un objetivo fundamental que es el aumento de datos, por lo que, para hacer un sistema más robusto, se recomienda utilizar técnicas como las distorsiones fotométricas como, brillo, contraste, tono, saturación y geométricas (recorte, giro, rotaciones, escala), permitiendo generar múltiples muestras a partir de una sola muestra. Este detector demostró ser fiable, rápido y preciso pudiendo ser ejecutado en ordenadores con GPU convencionales.

- YOLOv5:

YOLOv5 es un algoritmo que se basa en las anteriores versiones de YOLO. Desarrollado por Glenn Jocher y el equipo de Ultralytics [36]. YOLOv5 se encuentra abierto a la comunidad a través de Github, está en constante desarrollo con el objetivo de mejorar su rendimiento y también sometido a diversos cambios para lograr ser adaptado en paralelo con las librerías y métodos vigentes . Se considera la ultima versión disponible y gracias al desarrollo constante existen múltiples modelos pre-entrenados que permiten trabajar un modelo con un rango de precisión y parámetros definidos, los cuales se deben escoger de acuerdo a la disponibilidad del sistema o entorno en el que se trabaje.

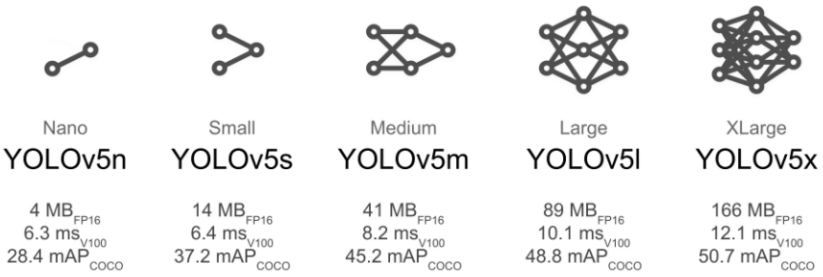


Figura 24: Modelos pre-entrenados disponibles.
Fuente: YOLOv5.

Existen 5 modelos pre-entrenados disponibles, los cuales pueden ser cargados al momento de entrenar YOLOv5 con algún conjunto de datos disponible o un conjunto de datos propio. Cada uno de estos modelos tiene un numero de parámetros definido, donde la versión más liviana YOLOv5n (Nano) que pesa 4MB posee 1.9 millones de parámetros y un desempeño estimado de 28.4 mAP (precisión media), la que sera útil cuando se use un conjunto de datos de tamaño pequeño (también será útil la versión Small YOLOv5s con 7.2 millones de parámetros), pero con al menos 1000 instancias por clases, que es lo recomendado por los desarrolladores.

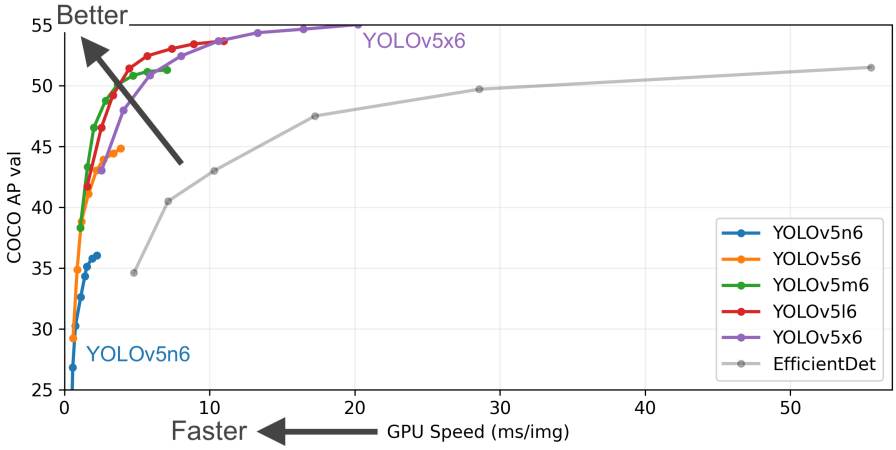


Figura 25: Gráfica de relación AP entre versiones de YOLOv5.
Fuente: Documentación de YOLOv5.

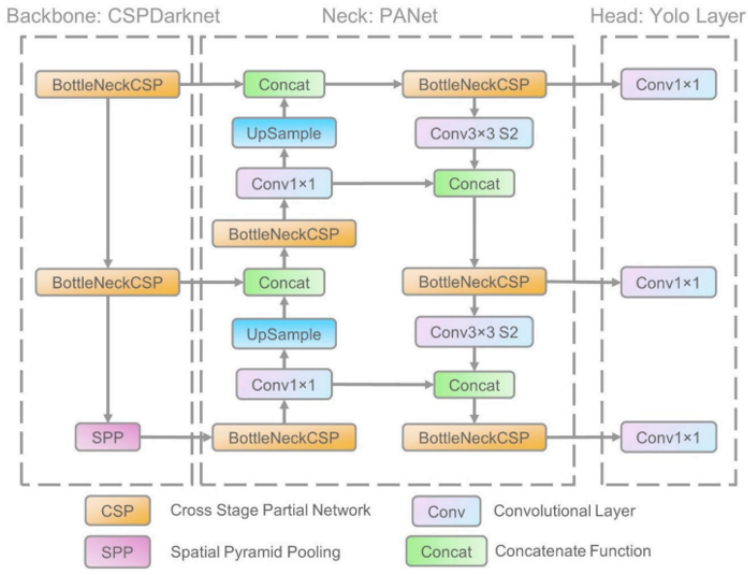


Figura 26: Arquitectura de YOLOv5.
Fuente: A Forest Fire Detection System Based on Ensemble Learning, 2021.

YOLOv5 fue desarrollado en el framework Pytorch, el cual es una biblioteca de Machine Learning de código abierto creada por Facebook, con una arquitectura de red definida como se muestra en la figura 26.

YOLOR: You Only Learn One Representation mas conocido por su acrónimo YOLOR, inicia con la premisa de que para entender el mundo se necesita unir múltiples aristas, en el caso de las personas, la unión de los sentidos y la experiencia pasada. Donde utilizamos el conocimiento como un punto de partida, el que puede ser obtenido de manera consciente o subconsciente (explícitamente e implícitamente), las que se almacenan y codifican en el cerebro [37].

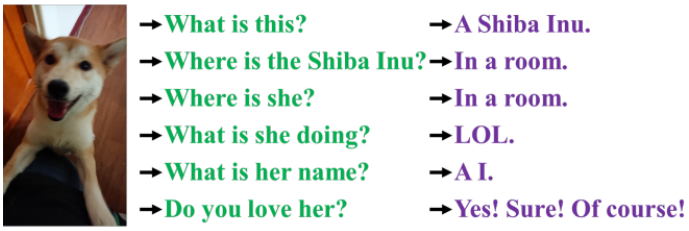


Figura 27: Análisis de un humano al recibir esta imagen como entrada.
Fuente: YOLOR [37].

Dada la necesidad de codificar y unificar estos dos tipos de conocimiento surge YOLOR, que mediante una red unificada permitirá realizar distintas tareas de manera eficiente y simultanea, con diferentes niveles de abstracción e inferencias asociadas a cada tarea. La figura 27 muestra como una persona puede realizar un análisis desde diferentes puntos de vista, respondiendo preguntas sencillas como: ¿Qué es? ¿Dónde esta? ¿Qué hace? o ¿Cuál es su nombre?, preguntas sencillas que están carentes en una red neuronal tradicional que fue entrenada para cumplir un objetivo especifico. Una red neuronal es capaz de generar conocimiento consciente en las primeras capas, para luego generar conocimiento inconsciente o implícito en las capas intermedias, que suelen adaptarse para poder obtener un desempeño o salida especifica acotada a ciertos valores o criterios.

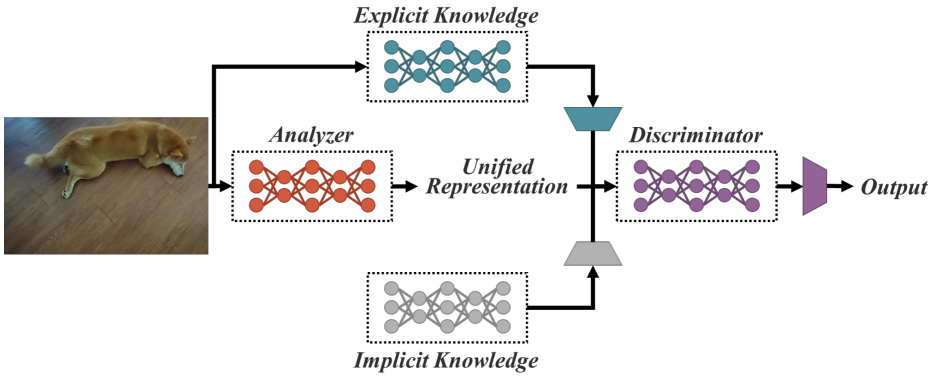


Figura 28: Red unificada: Representación con conocimiento explicito e implícito.
Fuente: YOLOR [37].

YOLOR considera una red unificada que combina el conocimiento explicito e implícito, permitiendo generar un modelo de representación general y que pueda adaptarse como otras representaciones para ser usadas en diversas tareas.

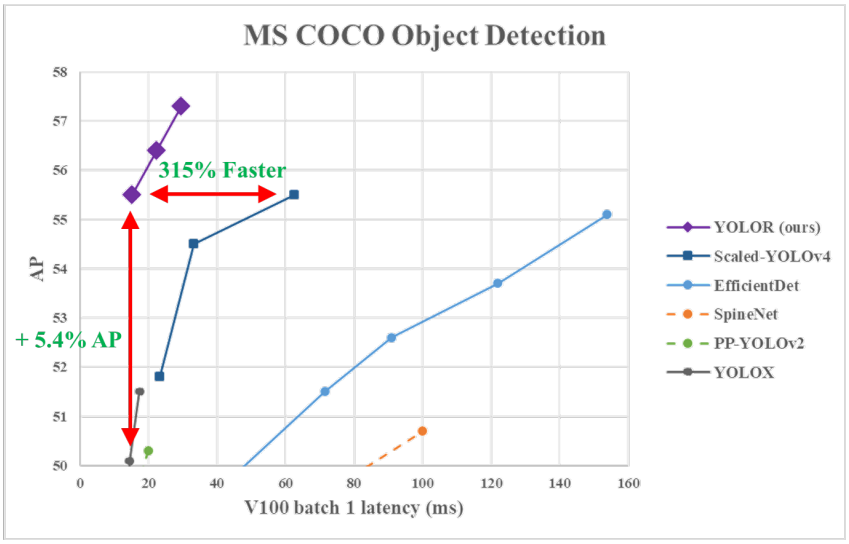


Figura 29: Comparación de desempeño con otras arquitecturas.
Fuente: Github de YOLOR.

Como se muestra en la figura 29, YOLOR obtiene 57.5 % de AP, 2 % por sobre YOLOv4 Scaled el cual alcanza un 55.5 %, pero YOLOR es un 315 % mas rápido.
Al igual que en el caso de YOLO, está orientado a la detección de objetos con una rapidez mucho mayor que sus competidores, permitiéndole ser fiable y rápido, es por esto que, surgen distintas variaciones en la arquitectura del modelo, las que se describen en la tabla 4. Estos modelos tienen un precisión media similar en la etapa de test, pero las versiones D6 son las con mejor desempeño y las que más tiempo tardan con 21.8 ms.

Modelo	ID	AP	AP50	batch1 rendimiento	batch32 inference
YOLOR-P6	1280	54.1 %	71.8 %	76 fps	8.3 ms
YOLOR-W6	1280	55.5 %	73.2 %	66 fps	10.7 ms
YOLOR-E6	1280	56.4 %	74.1 %	45 fps	17.1 ms
YOLOR-D6	1280	57.3 %	75 %	34 fps	21.8 ms
YOLOR-D6*	1280	57.8 %	75.5 %	34 fps	21.8 ms

Tabla 4: Precisión promedio de test para modelos de YOLOR.
Fuente: Github de YOLOR.

- Etapas para la detección de YOLO y YOLOR

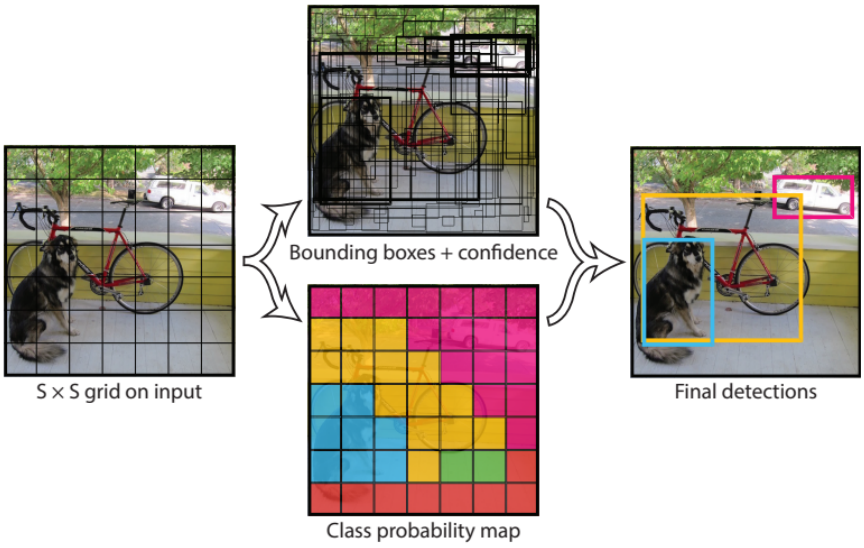


Figura 30: Generación de bounding boxes.
Fuente: YOLOv1 [38].

La figura 30 muestra las etapas que pasa cada imagen de entrada, inicialmente se genera una cuadricula definida por SxS números de cuadrados, para luego, definir el mapa de probabilidades de cada clase, esto consiste en asignar una probabilidad a cada cuadricula, como se aprecia en el class probability map se pinta de un color diferente cada cuadricula con cierto grado de pertenencia a alguna clase en específico y se generan múltiples cajas delimitadoras (bounding boxes) a las cuales se les asocia un grado de confianza (confidence)(mientras mayor sea este mejor), el que permite decidir cual caja delimitadora sirve para la etapa final.
En la ultima etapa se asocian las etapas anteriores y a las cajas delimitadoras con mayor confidence se les asocia una clase, es decir, la clase se asigna de acuerdo a la mayor cantidad de cuadriculas de una misma clase que contenga la caja delimitadora.

MASK R-CNN: Mask R-CNN, extiende Faster R-CNN, que es una red convolucional profunda utilizada para la detección de objetos, MASK R-CNN agrega una rama para predecir una máscara de objetos en paralelo con la rama existente para el reconocimiento de los cuadros delimitadores. Mask R-CNN es fácil de entrenar y agrega solo una pequeña sobrecarga a Faster R-CNN, que se ejecuta a 5 fps. Además, este algoritmo es fácil de generalizar a otras tareas [39].

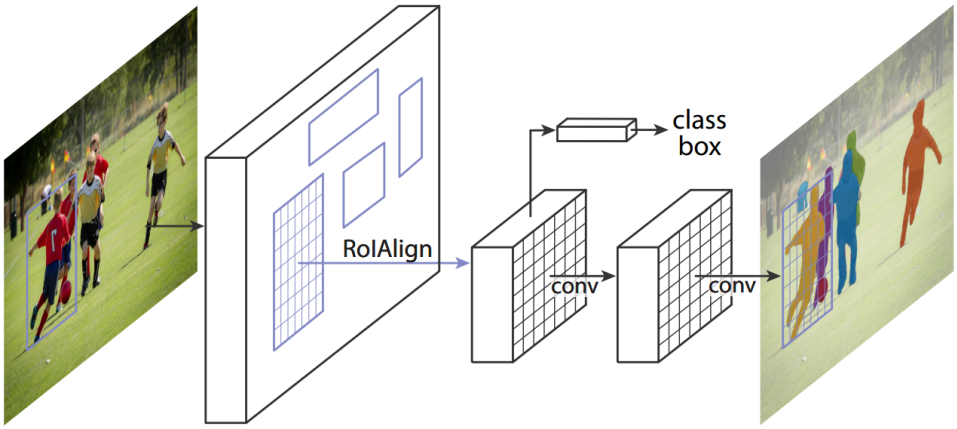


Figura 31: Arquitectura Mask-RCNN.
Fuente: Estudio [39].

SegNet: esta arquitectura de codificador-decodificar de aprendizaje profundo realiza clasificación de píxeles en imágenes RGB [40], esta arquitectura ya demostró ser exitosa en segmentación semántica sobre imágenes en otros campos, como por ejemplo conducción automática [41], en este estudio veremos como se comporta en la clasificación de arándanos y uvas.

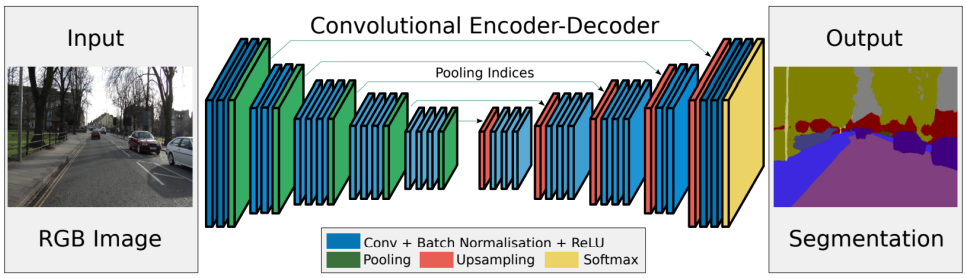


Figura 32: Arquitectura segnet.
Fuente: Estudio [41].

SSD (Single Shot Detector) : es de estructura piramidal en su CNN en las que las capa van disminuyendo gradualmente, esto le permite detectar objeto grandes y pequeños, a diferencia de YOLO no utiliza una grilla predefinida, pero posee “anclas” de distintas proporciones que se van escalando a medida que descendemos por la pirámide [42].

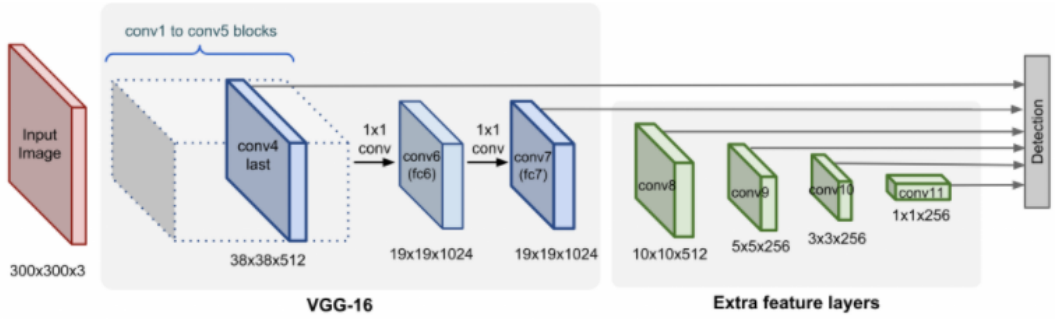


Figura 33: Arquitectura SSD.
Fuente: Estudio [42].

RetinaNet (2018): se basa en una estructura de CNN piramidal mejorada para poder reconocer objetos de diversos tamaños de una sola vez. Esta red innova con una nueva función de perdida llamada “Focal Loss” [43].

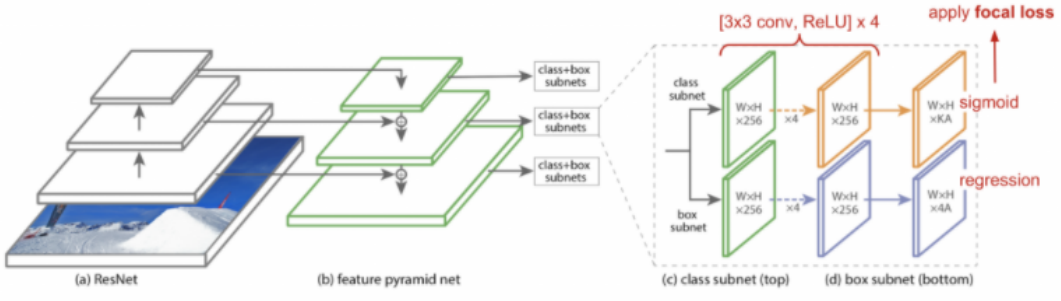


Figura 34: Arquitectura Retinanet.
Fuente: Estudio [43].

Google: Spinet (diciembre 2019): esta red rompe con la clásica estructura piramidal y propone una arquitectura llamada "scale-permuted" en la que se alternan diversos tamaños en las convoluciones [44].

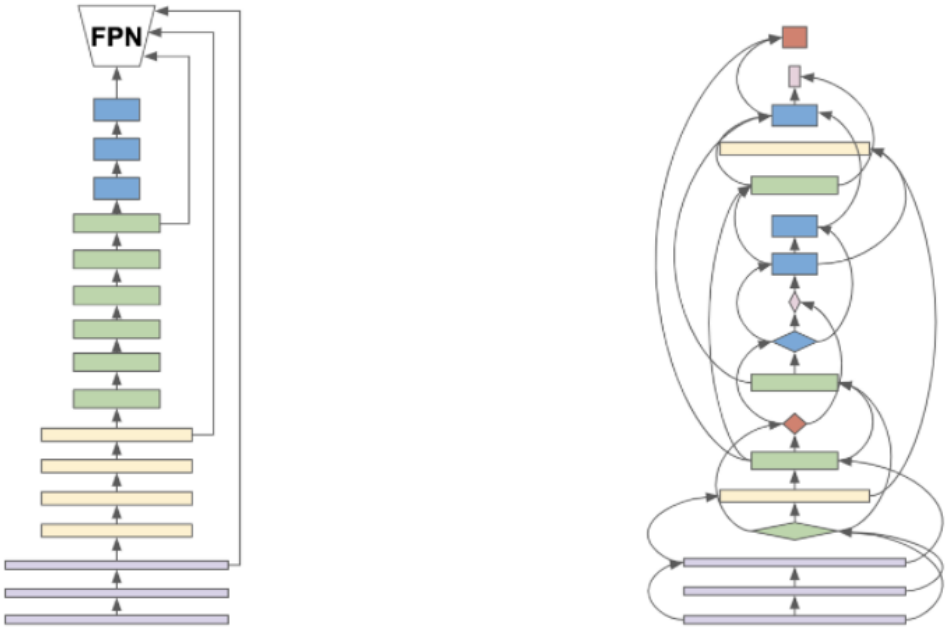


Figura 35: Arquitectura spinet.
Fuente: Estudio [44].

Facebook: DETR (junio 2020): Facebook propone una " End to End object detection with Transformers", es decir, utilizar una técnica efectiva de redes neuronales usadas en NLP, pero aplicarla a la detección de imágenes [45].

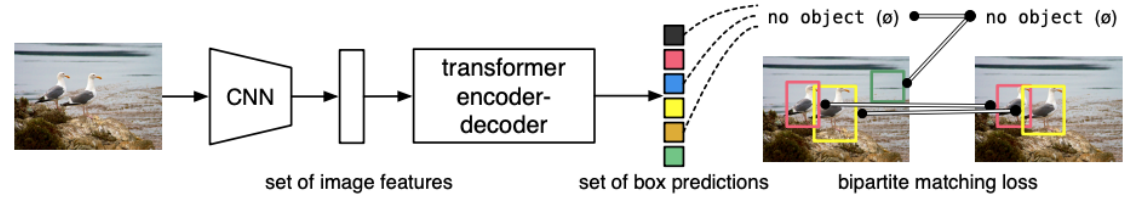


Figura 36: Arquitectura DETR.
Fuente: Estudio [45].

2.5. Métricas de desempeño

A medida que entrenamos los modelos de clasificación evaluaremos que tan buenos son, hay diversas maneras de evaluar el desempeño, en este proyecto se usara Python con la librería Scikit Learn.

- Matriz de confusión.

Figura 37: Matriz de confusión.
Fuente: Estudio [42].

Corresponde a una tabla que describe el desempeño de un modelo de clasificación en un conjunto de datos de prueba cuyos valores de prueba son desconocidos. Se llama "matriz de confusión" porque hace que sea fácil detectar donde el sistema está confundiendo dos clases.

o Verdaderos positivos (TP): Se considera cuando la clase real del punto de datos era 1 (Verdadero) y la predicha es también 1 (Verdadero).
o Verdaderos negativos (TN): Se considera cuando la clase real del punto de datos fue 0 (Falso) y el pronosticado también es 0 (Falso).
o Falso positivo (FP): Se considera cuando la clase real del puto de datos era 0 (Falso) y el pronosticado es 1 (Verdadero).
o Falso negativo (FN): Se considera cuando la clase real del punto de datos era 1 (verdadero)

y le valor predicho es 0 (Falso).
La matriz de confusión en si mismo no es una medida de desempeño, pero casi todas las métricas de desempeño se basan en la matriz de confusión.
Lo que buscamos en el proyecto es tratar de lograr que el modelo de 0 falsos positivos y 0 falsos negativos.

- Precisión

La precisión es la relación entre las predicciones correctas y el numero total de predicciones. O de manera más simple, con que frecuencia es correcto el clasificador.

$$Precisión = \frac{TP}{TP + FP} \tag{2}$$

- Acurracy

Exactitud es la relación entre las predicciones correctas y el número total de predicciones correctas previstas. Esto mide la precisión del clasificador a la hora de predecir casos positivos. Es la medida más directa de la calidad de los clasificadores. Es un valor entre 0 y 1. Cuanto mas alto mejor.

$$Exactitud = \frac{TP + TN}{TP + FP + FN + TN} \tag{3}$$

- Precisión promedio (AP)

La documentación de scikit learn precisa que es la media ponderada de la precisión alcanzada en cada umbral, considerando el resultado del umbral anterior como se muestra en la ecuación.

$$AP = \sum_{n} (R_n - R_{n-1}) \cdot P_n \tag{4}$$

Donde, se considera el enésimo valor de recall y precisión, como también el valor anterior de recall.

- Sensibilidad o recall

Es la relación entre las predicciones positivas correctas y el numero total de predicciones positivas, otra manera de explicarlo seria, cuan sensible es el clasificador para detectar instancias positivas. Esto se conoce como la tasa verdadera positiva.

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

CAPÍTULO 3. MATERIALES Y MÉTODOS

3.1. Ubicación

El sector frutícola para la región de Ñuble sin duda es una fuente de trabajo para un gran número de personas de la zona, hace ya bastantes años que se nota un progreso constante, con la ayuda de inversiones e iniciativas exitosas.

En esta zona el cambio climático parece ser una ventaja, aunque también presenta desafíos, básicamente relacionados con los recursos hídricos, en cuanto a la superficie frutícola, ésta ha aumentado considerablemente en los últimos años y según las cifras oficiales, los arándanos son la segunda fruta que más superficie tiene con 3.428 ha; después de las viñas con 12.858 ha; le siguen, frambuesas, 1.379 ha; cerezos, 1300 ha; nogal, 1,135 ha; manzanos, 924 ha; avellano, 760 ha; castaño, 690 ha; y kiwis, 464 ha.

Hoy en día las comunas con mayor presencia frutícola, incluyendo las viñas, son San Nicolas, con 5.552 ha; Coihueco, con 3.084 ha; Quillón, 1.923 ha y Chillan con 1.819 ha.

Este estudio se realizó en un campo de arándanos en el pueblo de Quillón, en el sector El Peumo, Quillón que por su ubicación geográfica posee un microclima de carácter templado, debido a que está en la línea de contacto entre el valle central y la vertiente oriental de la cordillera de la costa, cuenta con una temperatura media anual de 14.9°C. las temperaturas medias de enero varían de 27 a 30°C, la precipitación media anual oscila entre 700 a 1000 mm, principalmente entre abril y septiembre.

Un dato curioso del clima de Quillón; el jueves 26 de enero de 2017, se registro la temperatura mas alta a nivel nacional desde que existen registros, 44.9° a la sombra. Pero no es reconocida por la Dirección Meteorológica de Chile, debido o los incendios que circundaban la zona.

El campo de arándanos en el sector El Peumo en Quillón se visitó tres veces, en tres etapas distintas de maduración de los arándanos, entre finales de octubre y principios de diciembre del año 2021, en las tres visitas, nos encontramos con un abundante sol, con temperaturas sobre los 20°C.

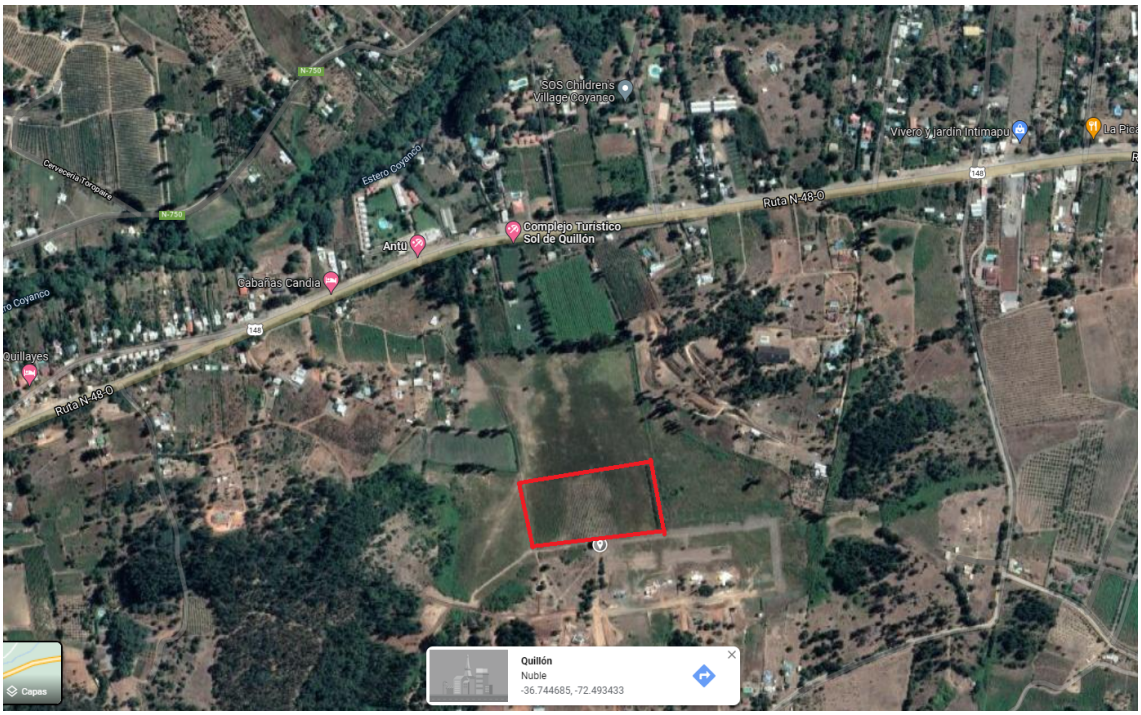


Figura 38: Ubicación exacta campo de arándanos.
Fuente: Google Maps.

Figura 39: Ubicación exacta campo de arándanos (zoom).
Fuente: Google Maps.

3.2. Equipos utilizados

Para el desarrollo de este trabajo se utilizan tres cámaras montadas en un trípode SOLIGOR WT-330A, permitiendo mantener la estabilidad en la toma de fotografías (ver figura 40), usadas de esa forma para capturar imágenes con posiciones similares que obtengan la mayor información de un sector específico. Además se utilizó la cámara Nikon Coolpix B700 en otras sesiones por separado para capturar múltiples imágenes para el testeo general de cada algoritmo.

Figura 40: Cámaras montadas en trípode.
Fuente: Elaboración propia.

Cámaras utilizadas:

- Nikon Coolpix B700
- Basler acA2440-20gc
- Basler acA2000-50gmNIR

3.2.1. Nikon Coolpix B700

Cámara digital compacta que posee un lente de cristal NIKKOR con un sensor CMOS y un zoom óptico de 60x, con una resolución de 20.2 MP, pantalla de 3 ” y un peso de 570 gramos. La cual se trabaja en modo manual regulando la apertura y sensibilidad.

Cámara Nikon Coolpix B700			
Modelo	Coolpix B700	**Manufactura**	Nikon
Tipo	Cámara digital compacta	**Peso (g)**	570
Sensor	CMOS	**Lente**	Lente de cristal NIKKOR con zoom óptico de 60x
Distancia focal	4.3-258 mm	**Numero f/ del lente**	f/3.3-6.5
Tamaño del sensor	1/2.3”	**Resolución (MP)**	20.2
Pixeles (H x V)	5184 x 3888	**Apertura**	Diafragma de iris de 6 laminas electromagnético.
Enfoque automatico	AF con detección de contraste	**Visor**	Visor electronico, 0.5 cm aprox. 921 mil puntos LCD.
Tamaño pantalla	3”diagonal	**Tipo de pantalla**	Ángulo Variable TFT LCD con Tratamiento antirreflejos.
Sensibilidad ISO	ISO 100 - 1600 ISO 3200	**Medición de la Exposición**	Matricial Centrada Puntual
Obturador	Obturador mecánico y electrónico CMOS	**Velocidad de obturación**	1/4000 a 1 s 1/4000 a 15 s 25 s
Alimentación	Bateria de Ion Litio EN-EL23	**Duración bateria**	420 disparos. 1h y 30 minutos.
Dimensiones (mm)	125.0 x 85 x 106.5	**Temperatura de Almacenamiento (°C)**	0 a 40 °C

Tabla 5: Especificaciones Nikon Coolpix B700.
Fuente: Nikon.

Figura 41: Cámara Nikon Coolpix B700.
Fuente: Nikon.

3.2.2. Cámaras Basler

Las cámaras Basler tienen múltiples ventajas, como la alta velocidad y tecnología. Son cámaras de tamaño reducido con grandes prestaciones, las cuales dan la posibilidad de alimentar con energía y transmitir los datos obtenidos por el mismo cable mediante la conexión PoE (Power over ethernet). Diseñadas para aplicaciones industriales, investigaciones y campos de todo tipo.

- Basler acA2440-20gc

La cámara Basler acA2440-20gc GigE con el sensor CMOS Sony IMX264 entrega 20 cuadros por segundo a una resolución de 5.0 MP.

Cámara Basler acA2440-20gc			
Video Output	GigE (PoE)	**Model number**	acA2440-20gc
Manufacturer	Basler	**Type**	Color camera
Sensor	Sony IMX264	**Type Sensor**	Progressive Scan CMOS
Type of Shutter	Global	**Camera sensor format**	2/3”
Camera Family	ace	**Resolution (MP)**	5.00
Pixels (H x V)	2,448 x 2,048	**Sensing Area H x V (mm)**	8.45 x 7.07
Frame rate (fps)	23	**Pixel Depth**	10/12 bit
Exposure time	Programmable via camera API, hardware trigger	**Connector**	GigE, RJ45 with Screw Locks
Power consumption (W)	2.7 (12VDC External Power) 3.3 (PoE)	**GPIOs**	1 opto-coupled input line, 1 opto-coupled output line, 1 general purpose I/O (GPIO) line
Dimensions (mm)	42 x 29 x 29	**Weight (g)**	90
Mount	C-Mount	**Machine Vision Standard**	GigE Vision, GenICam
Operation Temperature (°C)	0 to +50	**Storage Temperature (°C)**	-20 to +80

Tabla 6: Especificaciones de Cámara Basler acA2440-20gc.
Fuente: [46][47].

((a)) Frontal ((b)) Trasera

Figura 42: Cámara Basler acA2440-20gc.
Fuente: [46][47].

- Basler acA2000-50gmNIR

La cámara Basler acA2000-50gmNIR GigE con el sensor CMOS CMV2000 de ams (NIR mejorada) entrega 50 cuadros por segundo a una resolución de 2 MP.

Cámara Basler acA2000-50gmNIR			
Video Output	GigE (PoE)	**Model number**	acA2000-50gmNIR
Manufacturer	Basler	**Type**	NIR camera
Sensor	CMOSIS CMV2000	**Type Sensor**	Progressive Scan CMOS
Type of Shutter	Global	**Camera sensor format**	2/3”
Camera Family	ace	**Resolution (MP)**	2.20
Pixels (H x V)	2,048 x 1088	**Sensing Area H x V (mm)**	11.26 x 5.98
Frame rate (fps)	2350	**Pixel Depth**	12 bit
Exposure time	Freely Programmable	**Connector**	GigE, RJ45 with Screw Locks
Power consumption (W)	2.5 (12VDC External Power Supply) 3.0 (PoE)	**GPIOs**	1 opto-coupled input line, 1 opto-coupled output line, 1 general purpose I/O (GPIO) line
Dimensions (mm)	42 x 29 x 29	**Weight (g)**	90
Mount	C-Mount	**Machine Vision Standard**	GigE Vision
Operation Temperature (°C)	0 to +50	**Storage Temperature (°C)**	-20 to +80

Tabla 7: Especificaciones de Cámara Basler acA2000-50gmNIR.
Fuente: [47].

((a)) Frontal ((b)) Trasera

Figura 43: Cámara Basler acA2000-50gmNIR.
Fuente: [47].

Diagrama de conexionado

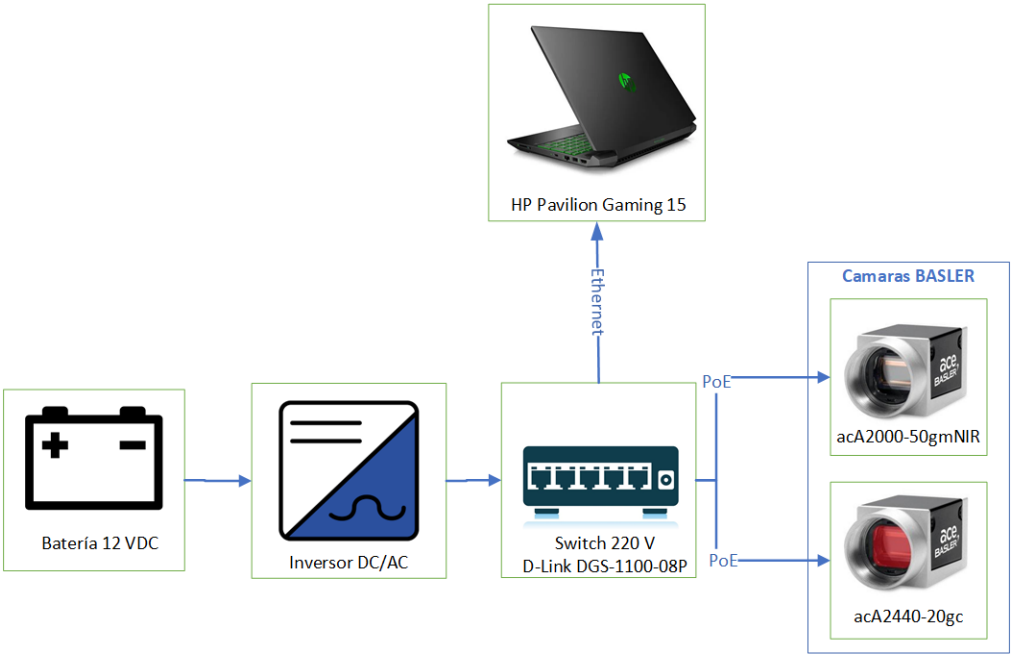


Figura 44: Diagrama de conexionado.
Fuente: Elaboración propia.

En este diagrama de conexionado no se considera la cámara Nikon Coolpix 7000 dado que esta no requiere alimentación desde el switch o la batería auxiliar, mas bien utiliza su propia batería y almacena las fotos directamente a una tarjeta de almacenamiento SDXC de 128 GB en nuestro caso, una vez finalizada la sesión se traspasan al computador utilizado.

El diagrama descrito en la Figura 44 considera una batería de 12 VDC que se conecta mediante dos cables con terminal tipo pinza a un inversor de DC a AC el cual es necesario para alimentar el switch que funciona solo con 220 VAC. El cual mediante tres cables ethernet se conecta a las cámaras (2 cables que alimentan y transmiten datos) y el tercero al computador (Transmite datos mediante ethernet), para estas dos cámaras se utiliza el software Pylon viewer para la visualización y almacenamiento de los datos.

3.3. Software utilizado

Para el uso de cámaras especializadas como las utilizadas en este trabajo, se requiere considerar el software adaptado y más utilizado para cada cámara. En este caso las cámaras Basler se utiliza un software específico denominado **Pylon viewer**, permite verificar si existe disponibilidad de cada una de ellas y además permite visualizar y capturar las imágenes obtenidas. También, incorpora **Pylon IP configurator** que permite verificar la comunicación entre el computador y las cámaras.

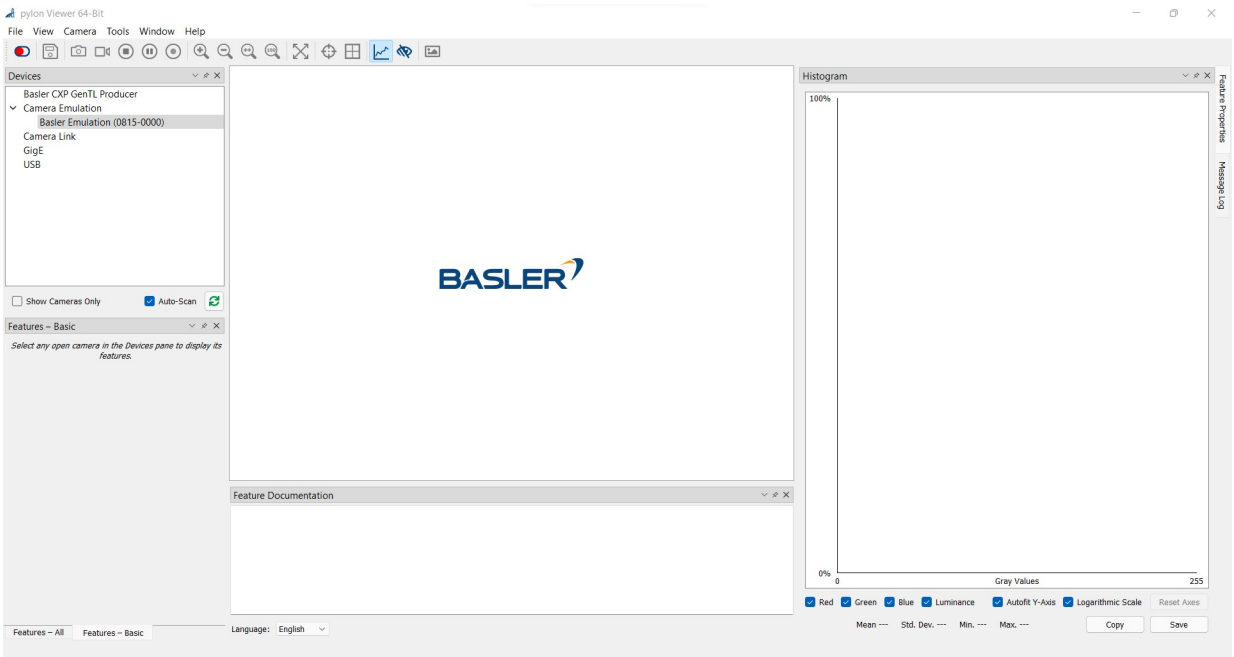
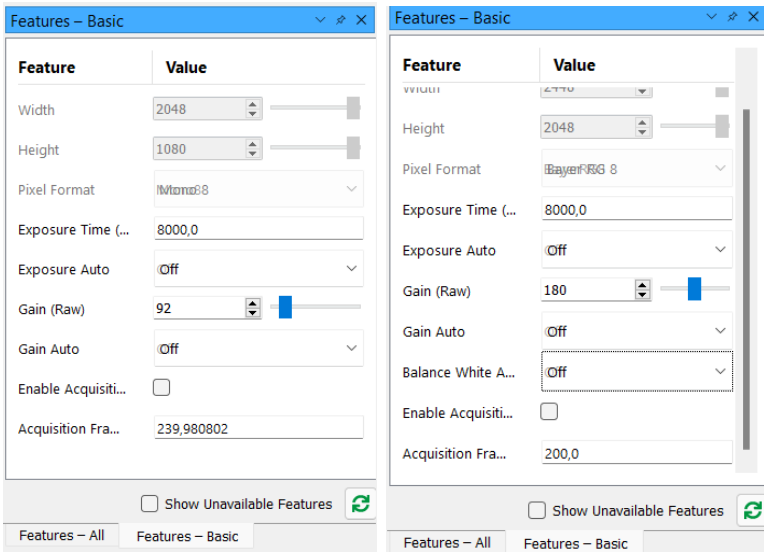


Figura 45: Interfaz de Pylon viewer.

3.3.1. Parámetros utilizados



((a)) Basler acA2000-50gmNIR ((b)) Basler acA2440-20gc

Figura 46: Parámetros utilizados en Pylon viewer en cada cámara.

Para cada caso particular se definen tres parámetros fundamentales conocidos como:

Ganancia (Gain): La función de ganancia de la cámara permite aumentar el brillo de las imágenes emitidas por la cámara.
Al aumentar la ganancia se incrementan todos los valores de los píxeles de la imagen.

Exposición (Exposure): La función de tiempo de exposición especifica cuánto tiempo se expone el sensor a la luz durante la adquisición de imágenes.

Frames (Acquisition): La función de velocidad de adquisición de cuadros le permite establecer un límite superior para la velocidad de cuadro de la cámara.
Estos parámetros pueden definirse de forma automática seleccionando el apartado correspondiente, en nuestro caso estos parámetros automáticos no eran los óptimos dadas las condiciones climáticas y por sobre todo de luz. Por lo que todos se definen de forma manual, optimizando la captura de las fotos.

3.4. Procedimiento

Existen diversos algoritmos para el análisis y detección de objetos en imágenes, trabajamos con dos versiones de YOLO que son YOLOv5 y YOLOR.
Elegimos a YOLO por que es un sistema de detección probado en múltiples estudios, especialmente bueno en vehículos autónomos, ya que permite guiar al automóvil para evitar colisiones con peatones o arboles etc. Otro ámbito en que YOLO es muy bueno es en el conteo, por ejemplo, asistentes a un restaurante, además tiene buenos resultados en controles de calidad de frutas y verduras, cualidad que nos llamó la atención para elegirlo en este proyecto. Se distingue de otros algoritmos por su rapidez en identificar objetos en imágenes, pero sacrifica precisión al localizar objetos pequeños, en este estudio analizaremos que tan grande son sus limitaciones al identificar arándanos, donde en una imagen puede haber cientos.

De todas las versiones existentes de YOLO, elegimos YOLOv5 que es su versión más actual, además se encuentra abierta a la comunidad por medio de Github, y en constante desarrollo, la otra versión elegida es YOLOR, que por medio de una red unificada permite realizar tareas de manera eficiente y simultánea, con múltiples niveles de abstracción e inferencias asociadas a cada tarea. YOLOR nace con la intención de responder preguntas sencillas, ¿Qué es? ¿Dónde está? ¿Cuál es su nombre?, preguntas que no son respondidas mediante redes neuronales tradicionales que solo se entrenan para cumplir con un objetivo específico.
Por último, elegimos al mejor algoritmo de detección de objetos actualmente, Mask R-CNN, que es una especie de sucesor de Faster R-CNN, ya que, lo extiende agregando una rama para predecir una máscara de objetos en paralelo con la rama existente para el reconocimiento de los cuadros delimitadores. Ha logrado los mejores resultados en las tres competiciones de la serie de desafíos COCO en el año 2016 (33), incluida la segmentación de instancias, la detección de objetos de cuadro y la detección de puntos clave humanos. En resumen, Mask R-CNN es una red neuronal convolucional, que además de identificar el objeto y su posición, también dibuja el contorno del objeto.
Compararemos estos tres algoritmos para determinar cuál tiene un mejor desempeño en la detección de arándanos.

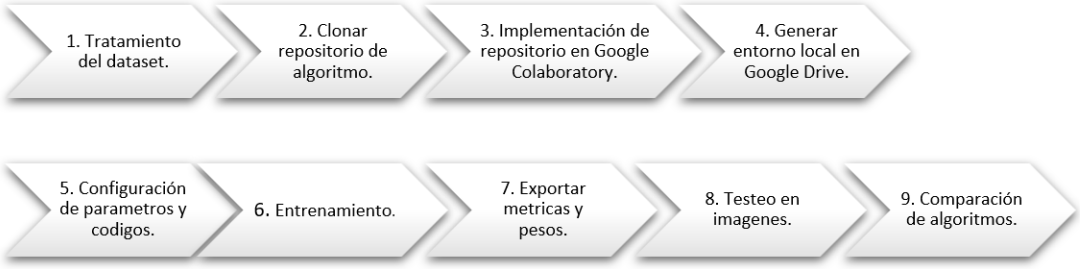


Figura 47: Etapas del procedimiento.

3.4.1. Tratamiento del dataset

La recopilación de imágenes se realizó en un campo de arándanos, en la región de Ñuble, en un periodo de un poco más de un mes, desde finales de octubre hasta principios de diciembre. En el momento que se tomaron las fotografías el clima en todo momento fue favorable, es decir sobre los 20°C y despejado. Se utilizaron diferentes cámaras, para captar un espectro electromagnético más amplio. La parte infrarroja del espectro electromagnético se divide en tres regiones, infrarrojos cercanos, medios y lejanos, en este caso utilizamos el infrarrojo cercano, medio y térmico que posee longitudes de onda entre los 700 a 1400 nm [23].

En la práctica entre más sean las imágenes y entre más diferentes sean, mejor será el modelo, es por esto que usamos una técnica que se usa habitualmente en estos casos que es la expansión del dataset, es una técnica para aumentar la diversidad del conjunto de entrenamiento mediante la aplicación de transformaciones aleatorias, como por ejemplo la rotación de imágenes. Para hacer esto, inicialmente usamos una biblioteca de Python llamada imgaug, la que permite el aumento de imágenes en experimentos de aprendizaje automático, con múltiples técnicas como rotaciones, variaciones de color, contraste, aumentos de puntos claves, cuadros delimitadores y otros.

Con la técnica de la expansión del dataset generamos imágenes con algunas modificaciones, como rotaciones, recortamos partes de imágenes, ampliamos imágenes. Por cada imagen original generamos hasta 7 imágenes, con esas pequeñas variaciones y luego de seleccionar algunas imágenes, logramos extender el dataset a 869 imágenes.

Para las anotaciones debemos definir la posición del arándano manualmente y asignarle las etiquetas correspondientes, se asignan según el criterio del color de cada baya, es decir, se asigna la clase inmaduro a un arándano de color verde y/o rojizo, se asocia la clase medio maduro a un color morado claro y/o rojo mas oscuro y se asocia la clase maduro a un color azul y/o morado oscuro.

Para este proyecto usamos Make sense que es una herramienta online open source que no requiere instalación, donde podemos importar nuestras imágenes y dibujar la posición del arándano manualmente, por medio de polígonos. Luego de que todas las imágenes tuvieran sus etiquetas correspondientes, podemos exportar la anotación con extensión .json, que luego será usada en el modelo.

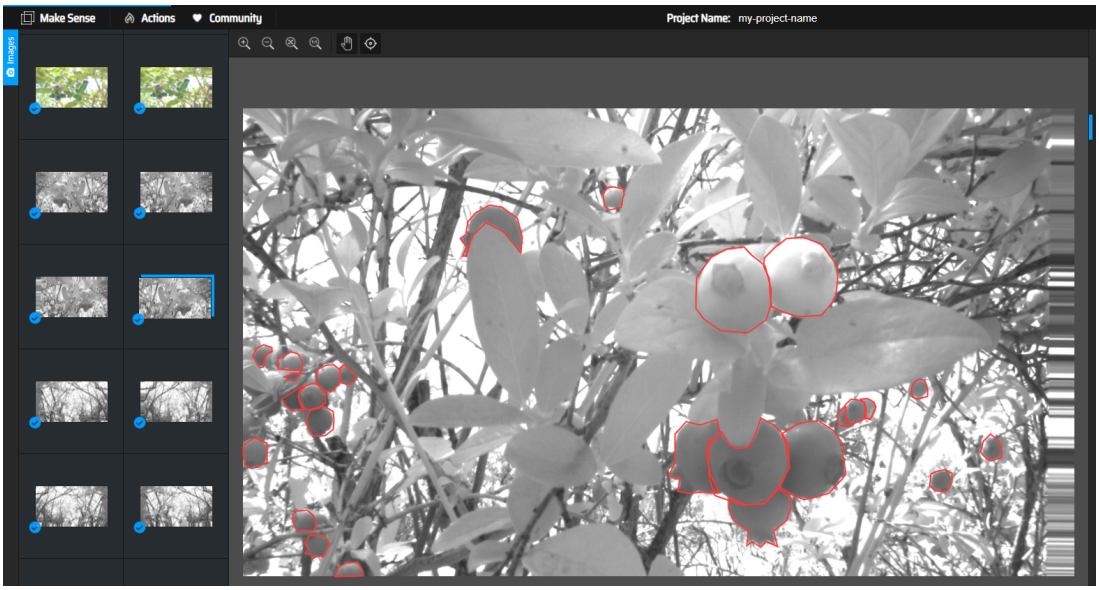


Figura 48: Anotaciones NIR en makesense.

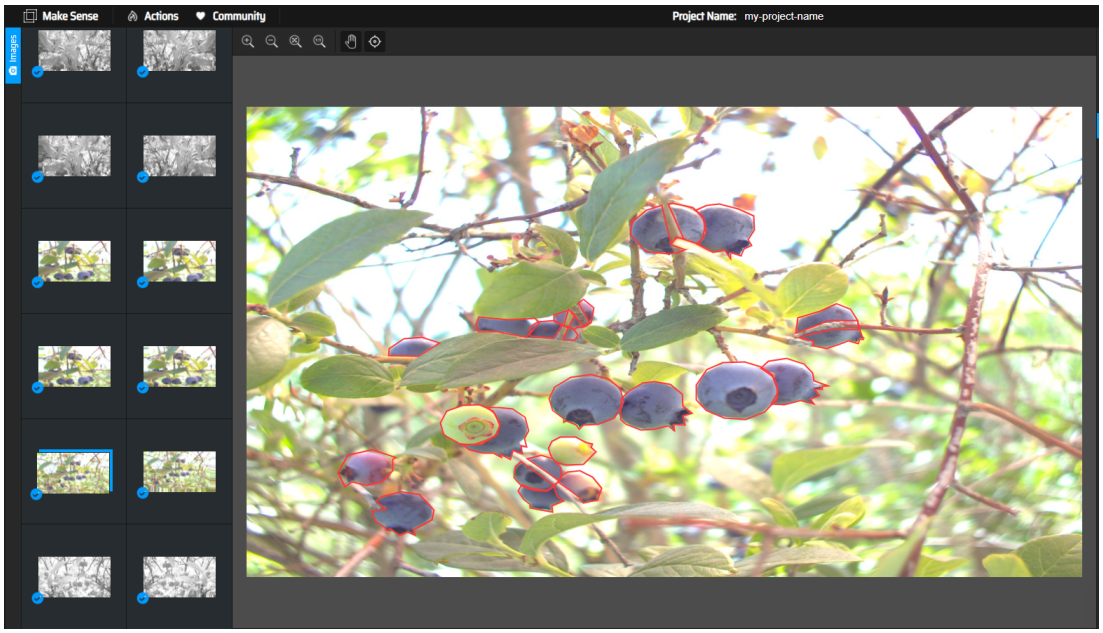


Figura 49: Anotaciones VIS en makesense.

Para el caso de la detección de objetos, utilizamos una plataforma llamada Roboflow annotate que permite sin descargar ningún software subir las imágenes que se necesite trabajar para crear un dataset con anotaciones en los formatos más utilizados como lo son json, xml, csv y txt, útiles para ser usados en múltiples algoritmos y modelos creados mediante Machine Learning, Deep Learning e Inteligencia Artificial en general.
Esta plataforma permite etiquetar de manera rápida y con múltiples clases solo utilizando el navegador. Se trabaja en dos interfaces principales, la figura 50 muestra donde se cargan las imágenes, se divide el dataset entre imágenes para entrenamiento y validación y se guardan todos los progresos. En la figura 51 se generan las tres clases y se etiqueta manualmente cada una de estas, las cuales se diferencian por el nombre y color específico que le asigna la misma plataforma, para luego generar el dataset en formato .txt.
Para exportar el dataset vamos directamente al botón **Generate new version** el cual permite dejar registro de cada una de los datasets generados para exportar, esta opción también permite realizar tratamiento de la data directamente a las imágenes etiquetadas (lo utilizamos para generar 3 imágenes: la primera se rota en un rango del 10 al 15 %, la segunda se voltea horizontalmente y a la tercera se le aplica un zoom de un 20 a 40 %). Luego de generar la versión, la plataforma permite exportar en un archivo zip las etiquetas en múltiples formatos, para nuestro caso utilizamos la versión YOLOv5 PyTorch el cual nos entrega el formato específico para nuestro algoritmo, con las imágenes debidamente etiquetadas y agrupadas en dos carpetas (train y valid), las que luego serán utilizadas para el entrenamiento de YOLOv5 y YOLOR.

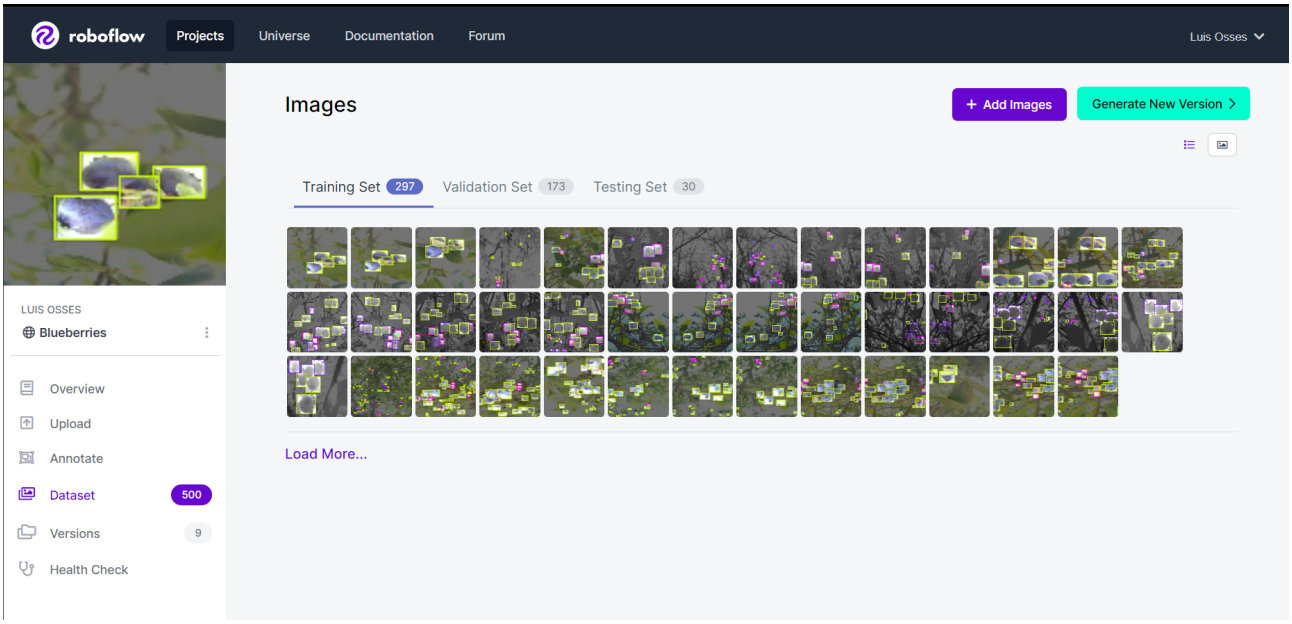


Figura 50: Inicio de plataforma de Roboflow.

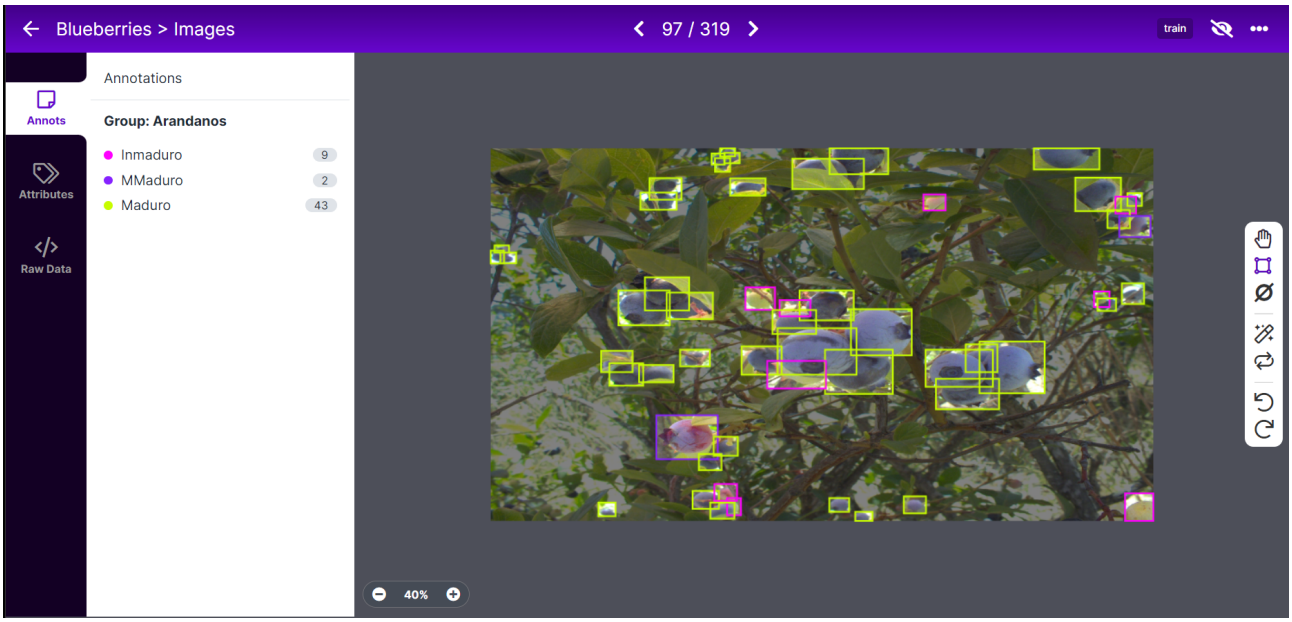


Figura 51: Anotación de imágenes de Roboflow.

3.4.2. Clonar e implementar repositorio en Google Colaboratory

Colaboratory o "Colab", es un producto que ofrece Google Research, que permite a cualquier usuario escribir y ejecutar código arbitrario de Python en el navegador, basada en los cuadernos de jupyter notebook. Esta herramienta es especialmente adecuada para tareas de aprendizaje automático y análisis de datos.

Para cada algoritmo implementado se utiliza el repositorio original desarrollado por cada investigador y grupo de investigación según corresponda.
En cada caso se utiliza la misma linea de código para clonar e implementar el algoritmo.
Antes de clonar el repositorio, generamos un entorno local en nuestro Google Drive, de la siguiente forma:

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

De esta forma instanciamos Google Drive y damos acceso a todas las carpetas, para luego cargar el dataset y cualquier código a trabajar directamente en nuestras carpetas, esto sirve para guardar los pesos, detecciones y testeo en Google Drive.

```
1. !git clone https://github.com/ultralytics/yolov5
2. !git clone https://github.com/luisosses1601/yolor
```

Existen dos formas para trabajar con este tipo de recursos de Github, podemos utilizar la opción 1, descargando directamente desde el repositorio original. Y la opción 2, es realizar un fork al repositorio original, esto se utiliza para crear una copia del repositorio en una cuenta de Github, vale decir que en este caso esta copia del repositorio es modificable y se puede trabajar sin afectar el original.

- YOLOv5: El Github oficial desarrollado por Ultralytics y Gleen Jocher su fundador, este Github esta alojado en YOLOv5 y posee múltiples carpetas que contienen todo lo necesario para su implementación.
- YOLOR: El Github oficial desarrollado por Wong Kin Yiu, contiene todo lo necesario para implementar el algoritmo con algunas modificaciones, y los pesos y métricas de la investigación base [37], alojado y documentado en YOLOR.
- Mask R-CNN: El Github oficial desarrollado por Matterport con un grupo de investigadores, este algoritmo requiere un etiquetado especial, por lo que la data se trabaja de otra manera y se explica en el Github Mask R-CNN, la que contiene todo los archivos necesarios para su implementación con diferentes modificaciones en cada caso.

Para que cada algoritmo funcione se deben cumplir algunos requerimientos y cargar librerías y versiones específicas de estas mismas. Por lo tanto, en cada caso se prepara un archivo .txt, con las librerías utilizadas y se instancian de la siguiente forma. Recordar que $\%pip$ es el instalador de paquetes de Python, por lo que se utiliza en varias ocasiones para cargar e instalar los paquetes necesarios para que la implementación funcione de manera correcta.

```
1. %pip install -qr requirements.txt
```

```
# Base ----------------------------------------
matplotlib>=3.2.2
numpy>=1.18.5
opencv-python>=4.1.2
Pillow>=7.1.2
PyYAML>=5.3.1
requests>=2.23.0
scipy>=1.4.1
torch>=1.7.0
torchvision>=0.8.1
tqdm>=4.41.0

# Logging -------------------------------------
tensorboard>=2.4.1
# wandb

# Plotting ------------------------------------
pandas>=1.1.4
seaborn>=0.11.0

# Export --------------------------------------
# coremltools>=4.1  # CoreML export
# onnx>=1.9.0  # ONNX export
# onnx-simplifier>=0.3.6  # ONNX simplifier
# scikit-learn==0.19.2  # CoreML quantization
# tensorflow>=2.4.1  # TFLite export
# tensorflowjs>=3.9.0  # TF.js export
# openvino-dev  # OpenVINO export

# Extras --------------------------------------
# albumentations>=1.0.3
# Cython  # for pycocotools https://github.com/cocodataset/cocoapi/issues/172
# pycocotools>=2.0  # COCO mAP
# roboflow
thop  # FLOPs computation
```

Figura 52: Requerimientos de YOLOv5.

Como se muestra en la figura 52, se utilizan versiones especificas de algunas librerías base como numpy, opencv y otras, como también se utilizan librerías especificas, con versiones definidas para las gráficas y visualización de métricas de desempeño (pandas, seaborn y tensorboard). Esto mismo se repite para cada algoritmo implementado en este documento.

3.4.3. Entrenamiento de algoritmo

Para la formación de cada algoritmo (YOLOv5, YOLOR y Mask R-CNN) hemos preparado un cuaderno en Google Colab.
En primer lugar, debemos habilitar la GPU porque necesitamos la tarjeta gráfica para realizar el entrenamiento.
La computación acelerada por GPU, por sus siglas, unidad de procesamiento de gráficos, junto a una CPU permiten el aceleramiento y funcionamiento de las aplicaciones de aprendizaje profundo, esta tecnología permite asignar a la GPU el trabajo de los aspectos de la aplicación donde la computación es más intensa, mientras que el resto del código se ejecuta en la CPU. Desde nuestra perspectiva las aplicaciones se ejecutan más rápido, la forma más sencilla de comprender la diferencia entre una GPU y una CPU es comparar la forma en como procesan las tareas. Una CPU tiene algunos núcleos optimizados para el procesamiento en serie secuencial, en cambio una GPU cuenta con una arquitectura en paralelo que consiste en miles de núcleos pequeños y eficaces, por lo tanto pueden resolver varias tareas al mismo tiempo.
En la primera parte del código, clonaremos el repositorio como se señala en la sección anterior. Además, instalaremos Tensor Flow y todas las librerías necesarias para el proceso (Requirements.txt). Luego, instanciamos las imágenes ya cargadas en Google Drive, para el caso de YOLOv5 y YOLOR dividimos las imágenes en dos carpetas (train y valid) y en cada una de estas carpetas se crean otras dos carpetas que almacenan las imágenes y las etiquetas (images y labels), para el caso de Mask-RCNN creamos dos carpetas llamada train y val, estas contienen todas las imágenes etiquetadas y un solo archivo en cada caso (annotations.json) que contiene la referencia a cada una de las imágenes y las etiquetas.

Una vez cargados los archivos, se señala el directorio donde están ubicadas las imágenes. Para señalarle al algoritmo donde debe buscar, para el caso de YOLOv5 y YOLOR generamos un archivo .yaml que contiene el directorio de las carpetas de entrenamiento y validación, y se especifica el numero de clases con sus respectivos nombres, y se carga en la carpeta data generada por el repositorio de Github. Para el caso de Mask basta con especificar en el cuaderno de Google Colab la dirección de la carpeta de imágenes y el nombre de las anotaciones, estas imágenes se separan con el criterio 70-30, un 70 % será para entrenamiento y un 30 % para validación.

Para cada algoritmo tenemos archivos comunes (archivos Python), que realizan las mismas tareas, finalmente se utilizan en conjunto, permitiendo generar métricas de desempeño en el entrenamiento (Utils.py), el modelo contenido en model.py, visualize.py para el muestreo y visualización de datos y pruebas, config.py que contiene las configuraciones relevantes, como el licenciamiento y carpetas de la data. Como ejemplo se detalla el proyecto Mask R-CNN que cuenta con 4 directorios:

- **mrcnn:** Este es el directorio principal que contiene el código Python del proyecto.
- **samples:** Jupyter notebooks que proporcionan algunos ejemplos para usar el proyecto.

- **images:** Una colección de imágenes de prueba
- **assets:** Algunas imágenes anotadas

El directorio más importante es mrcnn, ya que contiene el código fuente del proyecto, donde conectamos todas las utilidades provista por el repositorio (Blueberry.py). Y tiene los siguientes archivos **Python**:

- **init.py:** Marca la carpeta mrcnn como una biblioteca de Python.
- **model.py:** Tiene las funciones y clases para construir las capas y el modelo.
- **config.py:** contiene una clase denominada Config que contiene algunos parámetros de configuración sobre el modelo.
- **utils.py:** Incluye algunas funciones y clases auxiliares.
- **visualize.py:** Visualiza los resultados del modelo.
- **parallel.model.py:** Para admitir varias GPU.

Algunos de los archivos en la raíz del proyecto son:

- **setup.py:** Se utiliza para instalar el proyecto mediante pip.
- **README.md:** Un archivo Markdown que documenta el proyecto.
- **LICENCIA:** La licencia MIT.

Para cada entrenamiento se debe considerar la plataforma y recursos que dispone esta misma. En este caso se utiliza Google Colab, el cual dispone de 16GB de GPU, de los cuales se utilizan 15.9 GB como máximo para que el entrenamiento termine de forma exitosa.

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   40C    P0    27W / 250W |      2MiB / 16280MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

Figura 53: Recursos disponibles para el entrenamiento (GPU) en Google Colab pro.

Para cumplir con este requerimiento se deben definir 3 parámetros importantes para cada caso:

- Batch size: Es el número de ejemplos por cada iteración, es decir, lo usamos como una tasa de aprendizaje que define el número de muestras a usar antes de actualizar los parámetros internos del modelo, por lo que utilizamos el batch size más grande que permita nuestro sistema (mayor batch size implica un mayor uso de recursos).
- Image size: Se define de acuerdo al tamaño de las imágenes del dataset de entrada. De no poder utilizar el tamaño de las imágenes de entrada, se utiliza la mayor resolución que permita el sistema utilizado.
- Modelo: Se define como la estructura y arquitectura con la que se llevara a cabo el entrenamiento, en cada caso es diferente y especifica. Los modelos disponibles tienen características definidas (número de capas, parámetros, costo computacional, tamaño), por lo que requieren diferentes recursos.

3.4.4. Análisis de parámetros de entrenamiento

Para realizar un análisis de los parámetros a utilizar, se generan tres entrenamientos con diferentes valores de batch size, utilizando YOLOv5s con el mismo tamaño de imagen de entrada y mismo número de épocas. Con la finalidad de ver después de cuantas etapas de entrenamiento el valor de mAP se asienta.

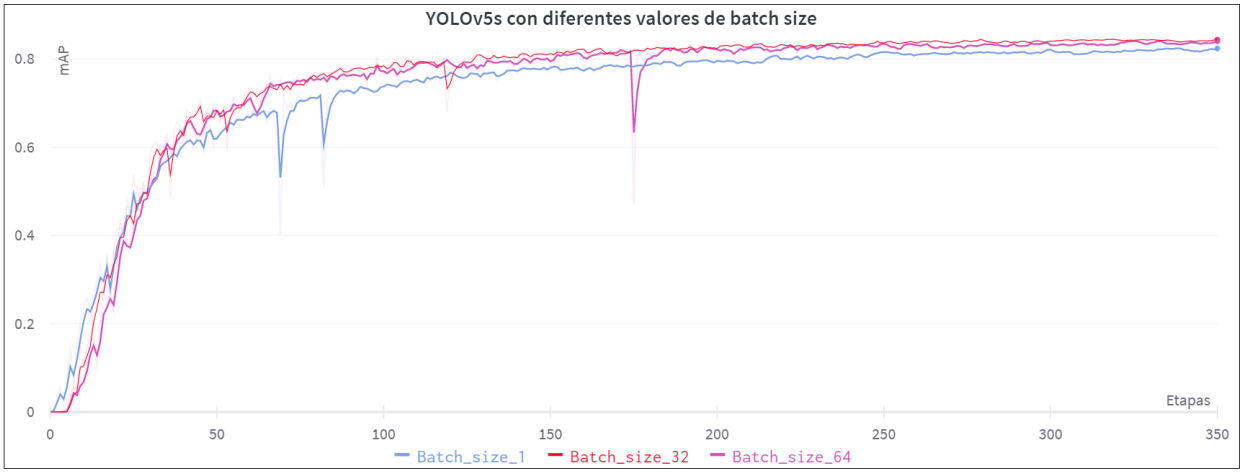


Figura 54: Gráfica de mAP con distintos valores de batch size.

Arquitectura	ID	Épocas	Batch size	Image size	Max. mAP	Tiempo empleado
YOLOv5s	Batch_size_1	350	1	720	82.5 %	10 horas
	Batch_size_32	350	32	720	84.4 %	2.4 horas
	Batch_size_64	350	64	720	84.3 %	2.5 horas

Tabla 8: Especificaciones utilizando YOLOv5s con distintos valores de batch size

La figura 54 muestra como pasadas las 300 épocas, el valor de mAP se asienta en un valor cercano al 84 %, por lo que 350 épocas es un valor suficiente para llegar al valor de asentamiento del mAP. Y la tabla 8 muestra como el tiempo empleado varia de acuerdo al batch size asignado, en este caso, el maximo mAP se obtiene con un batch size igual a 32 y en el menor tiempo. Por lo tanto, se utiliza un batch size igual a 32, con 350 épocas y una resolución de imágenes igual a 720, un modelo personalizado (intermedio) para cada caso, lo que nos permite utilizar el batch size descrito y el tamaño de la imagen adecuado. Con estos parámetros se alcanza el limite de recursos, llegando hasta 16GB de memoria disponible de la GPU.

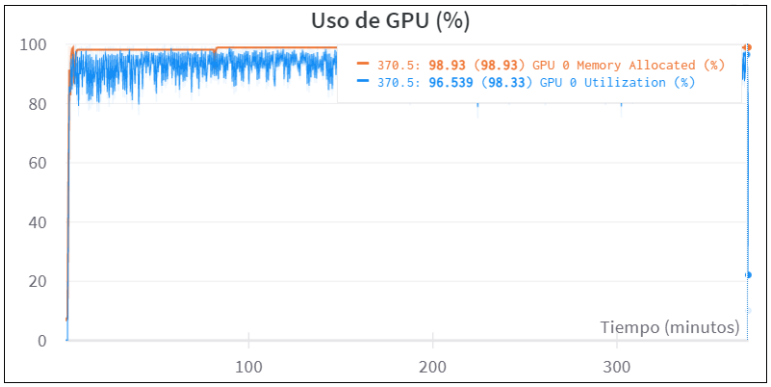


Figura 55: Recursos usados en el entrenamiento de YOLOv5 (GPU) en Google Colab pro. Fuente: Obtenida de la plataforma Wandb.ai.

La figura 55 muestra como la mayor parte del tiempo se utiliza un porcentaje cercano al 100 % de la GPU, utilizando toda la memoria disponible, llegando al 98.93 % de memoria asignada. En cada caso, instanciamos los modelos que serán detallados en la siguiente sección, además del número de épocas (se estandariza en 350), parámetros e hiperparámetros utilizados. Se definen las métricas, las que graficamos con ayuda de tensorboard que es el kit de herramientas de visualización de tensorflow (permite graficar los entrenamientos, funciones de perdida, grafo del modelo e imágenes) y wandb (permite graficar los entrenamientos, visualizar los lotes de entrenamiento, funciones de perdidas, gráfica y detección en tiempo real).

CAPÍTULO 4. RESULTADOS

En esta sección se muestran los resultados al utilizar dos técnicas para identificar y detectar tres clases de arándanos (Maduro, Mmaduro, Inmaduro). Estas técnicas son: **Segmentación semántica** mediante la creación de una mascara sobre cada uno de los arándanos identificados y **detección de objetos** mediante la generación de un cuadro delimitador sobre cada arándano identificado y su respectivo estado de maduración. Los algoritmos trabajados son: MASK RCNN (Segmentación semántica), YOLOv5 y YOLOR (Detección de objetos).

4.1. YOLOv5

You Only Look Once mas conocido por el acrónimo Y.O.L.O, el cual desde que fue presentado se utiliza para la detección de objetos de manera rápida y permite procesar imágenes en tiempo real a 45 fps [38], mediante el uso de clasificadores que se utilizan para realizar la detección y luego se crea un cuadro delimitador (bounding boxes), las cuales se separan espacialmente dependiendo de la clase detectada. La clase es detectada de acuerdo a una probabilidad asociada al cuadro delimitador, por lo tanto, cuando situamos el cuadro delimitador en cierto objeto detectado, se asocia un grado de pertenencia (de 0 a 1) a cada una de las clases posibles, para finalmente elegir la clase con mayor grado de pertenencia (1).
En este caso se utiliza la YOLO v5, la que es considerada la ultima versión de YOLO como tal, desarrollada y abierta a la comunidad por Ultralytics.

Luego del análisis de algoritmos específicos, YOLOv5 presenta múltiples arquitecturas y modelos pre-entrenados disponibles, desde modelos para pequeños datasets, hasta dataset con una mayor cantidad de datos e instancias generadas por cada imagen.
En esta sección se consideran dos versiones disponibles (YOLOv5s (37.2 mAP_{coco}) y YOLOv5x (50.7 mAP_{coco})), se utilizan estas dos versiones de YOLOv5 dado que, se realiza un análisis y comparación de desempeño usando las arquitecturas sin los pesos pre-entrenados.

En la figura 56, se describen las dos arquitecturas implementadas en YOLOv5:

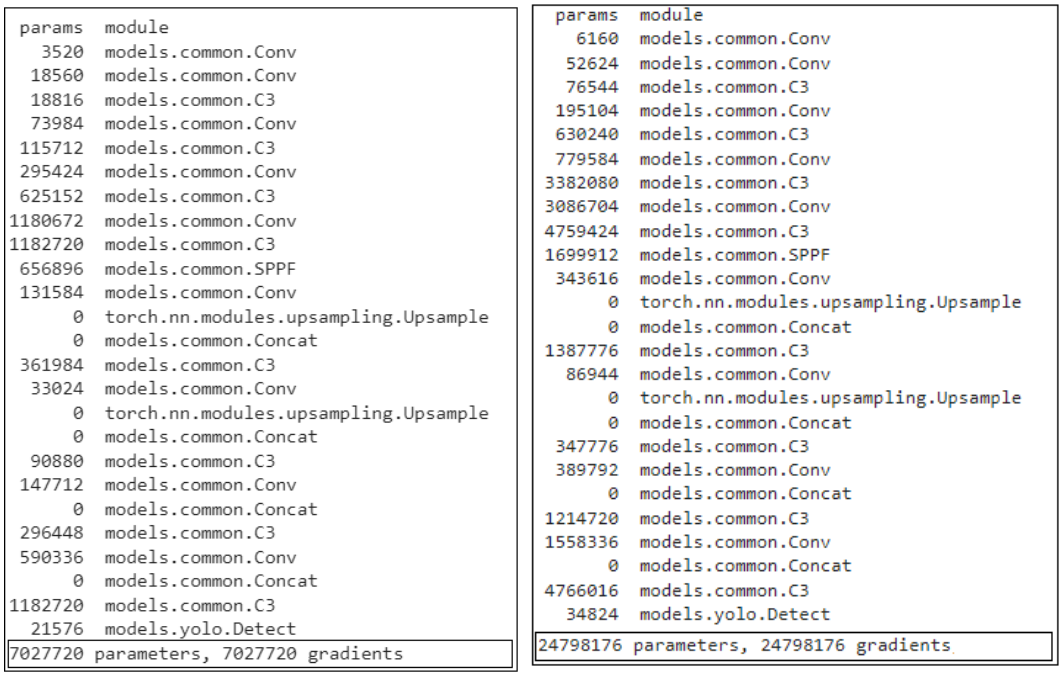
- YOLOv5s

Esta compuesta por 213 capas, descritas y conectadas como se muestra en la figura 56 a. La unión de cada capa consigue afinar y reducir errores en el modelo y aumentar el número de parámetros (7.018.216), además, se debe considerar que un mayor número de parámetros, implica un mayor costo y recursos computacionales.

- Custom YOLOv5

A diferencia de la arquitectura del modelo anterior utiliza 387 capas distribuidas como se muestra en la figura 56 b, la cual posee una cantidad de parámetros mucho mayor (24.798.176),

lo que se justifica con el aumento del número de capas y se traduce en una mejora en la precisión y rendimiento en general.



```
 params  module
   3520  models.common.Conv
  18560  models.common.Conv
  18816  models.common.C3
  73984  models.common.Conv
 115712  models.common.C3
 295424  models.common.Conv
 625152  models.common.C3
1180672  models.common.Conv
1182720  models.common.C3
 656896  models.common.SPPF
 131584  models.common.Conv
      0  torch.nn.modules.upsampling.Upsample
      0  models.common.Concat
 361984  models.common.C3
  33024  models.common.Conv
      0  torch.nn.modules.upsampling.Upsample
      0  models.common.Concat
  90880  models.common.C3
 147712  models.common.Conv
      0  models.common.Concat
 296448  models.common.C3
 590336  models.common.Conv
      0  models.common.Concat
1182720  models.common.C3
  21576  models.yolo.Detect
7027720 parameters, 7027720 gradients
```

((a)) Arquitectura YOLOv5s

```
 params  module
   6160  models.common.Conv
  52624  models.common.Conv
  76544  models.common.C3
 195104  models.common.Conv
 630240  models.common.C3
 779584  models.common.Conv
3382080  models.common.C3
3086704  models.common.Conv
4759424  models.common.C3
1699912  models.common.SPPF
 343616  models.common.Conv
      0  torch.nn.modules.upsampling.Upsample
      0  models.common.Concat
1387776  models.common.C3
  86944  models.common.Conv
      0  torch.nn.modules.upsampling.Upsample
      0  models.common.Concat
 347776  models.common.C3
 389792  models.common.Conv
      0  models.common.Concat
1214720  models.common.C3
1558336  models.common.Conv
      0  models.common.Concat
4766016  models.common.C3
  34824  models.yolo.Detect
24798176 parameters, 24798176 gradients
```

((b)) Arquitectura Personalizada de YOLOv5

Figura 56: Arquitectura de modelos YOLOv5.

La figura 57 y la tabla 9, muestran el resultado al utilizar YOLOv5s (**v5s-DataO** en gráfica) y custom YOLOv5 (**custom-Data0** en gráfica). Estos resultados fueron obtenidos utilizando un dataset de 400 imágenes (dataset original). Con el objetivo de analizar el comportamiento de ambas arquitecturas con el dataset inicial.
El entrenamiento se realiza en 350 épocas o etapas, para permitir que la respuesta se asiente con el paso de cada etapa. Donde cada modelo alcanza una media de precisión promedio (mAP) cercana al 77 % en el caso de YOLOv5s y 81 % en el caso de la versión personalizada de YOLOv5. El entrenamiento fue completado en 1 hora con YOLOv5s y custom YOLOv5 el entrenamiento se completa en 2.6 horas.

Clase	Precisión	Recall	mAP
Inmaduro	75.5 %	76.2 %	82.4 %
MMaduro	69.9 %	59 %	64.8 %
Maduro	76.3 %	81.8 %	85.9 %
	YOLOv5s		

Clase	Precisión	Recall	mAP
Inmaduro	83.2 %	79 %	84.9 %
MMaduro	71.4 %	61.7 %	70.6 %
Maduro	80.2 %	84.9 %	88.4 %
	custom YOLOv5		

Tabla 9: Métricas por clase de cada arquitectura utilizada.

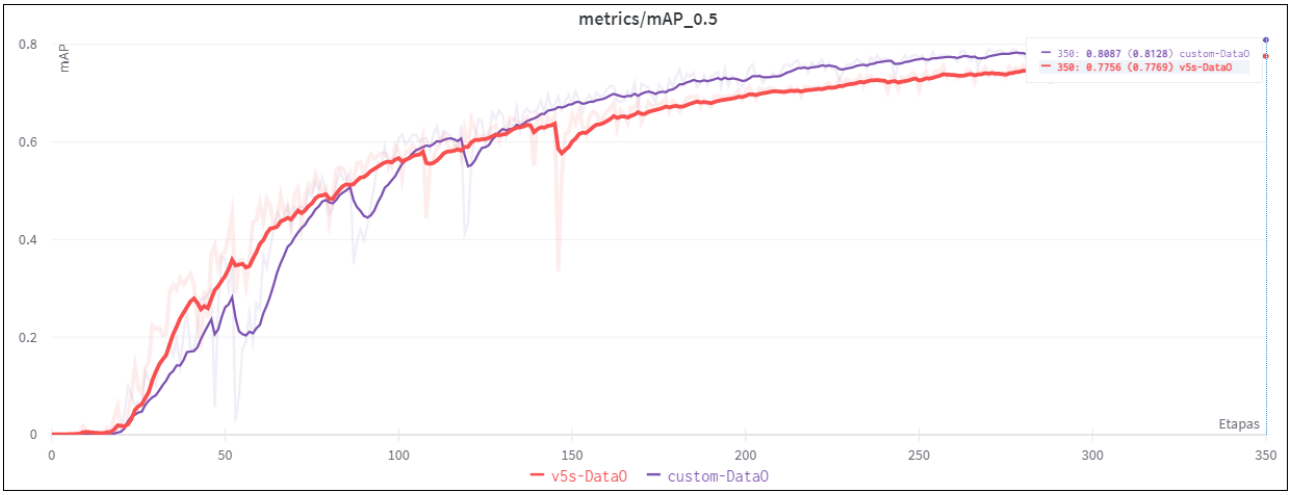


Figura 57: Gráfica comparativa entre dos modelos (mAP).

En ambos casos, el mejor rendimiento ocurre en la etapa final, donde YOLOv5s obtuvo 77.7 % mAP y custom YOLOv5 alcanza un 81.28 %, el que se muestra en el cuadro de la figura 57. El que se considera un resultado aceptable, pero es la base para analizar el tipo de modelo a utilizar.
Aunque la diferencia de la métrica mAP entre los dos modelos es cercana al 4 %, pero si revisamos la clase con mayor variación en la precisión es la clase inmaduro que obtiene 75.5 % con YOLOv5s y 83.2 % con custom YOLOv5, una variación cercana al 8 %.
Para realizar un análisis práctico se genera un testeo con un set de imágenes desconocido para cada modelo entrenado, la cual si se revisa exhaustivamente tiene **11 arándanos**. Se aprecia en la figura 58 que, en la práctica cada una de las arquitecturas usadas en el entrenamiento son capaces de detectar el arándano y el estado de maduración asociado a cada una de ellas, pero el modelo personalizado de YOLOv5 fue capaz de detectar los 11 arándanos contenidos en la imagen y cada uno con la madurez asociada correspondiente, con una precisión del 100 % con un valor de confidence igual a 0.46, por el contrario YOLOv5s detecta 12 arándanos , ya que no tiene la capacidad de diferenciar a que clase pertenece un arándano detectado (le asigna dos clases a un mismo arándano).
Si se compara el tiempo que demora cada arquitectura en detectar y generar los cuadros delimitadores en cada caso, YOLOv5s (0.011 s) es más rápida que la versión personalizada de YOLOv5 (0.017 s), lo que debe ser considerado a la hora de utilizar una arquitectura de acuerdo a las características y tiempos de un proyecto.

((a)) YOLOv5s ((b)) Custom YOLOv5

Figura 58: Detección de arándanos con confidence igual a 0.46.

Figura	ID	Clase	Detectados	Total
58	a	Inmaduro	3 arándanos	12 arándanos
		Mmaduro	2 arándanos	
		Maduro	7 arándanos	
	b	Inmaduro	3 arándanos	11 arándanos
		Mmaduro	2 arándanos	
		Maduro	6 arándanos	

Tabla 10: Detección de arándanos con: a) YOLOv5s y b) Custom YOLOv5.

4.1.1. Métricas de desempeño

La versión personalizada de YOLOv5 (Custom YOLOv5) demostró la capacidad de identificar y detectar un mayor número de arándanos, el cual obtuvo el mejor desempeño descrito en las métricas de la tabla 9. Es por esto que en la siguiente sección se utilizará esta versión, el cual se entrenara en 350 épocas con el dataset generado mediante técnicas de aumento de data y se compara con el entrenamiento con la data original. Para cada modelo se utilizo un batch size de 32, lo que se traduce en 25 iteraciones por etapa, además se utiliza la asignación de pesos para mejorar la precisión en la clase medio maduro.

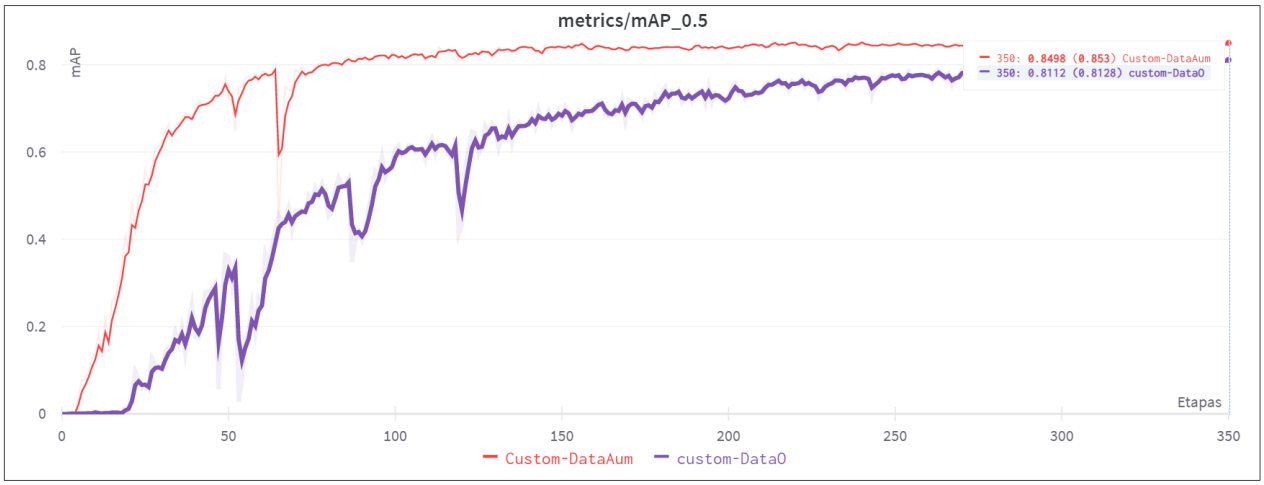


Figura 59: Gráfica de desempeño utilizando custom YOLOv5 (mAP).

Clase	Precisión	Recall	mAP
Inmaduro	83.2 %	79 %	84.9 %
MMaduro	71.4 %	61.7 %	70.6 %
Maduro	80.2 %	84.9 %	88.4 %
	Custom YOLOv5 (Data original)		

Clase	Precisión	Recall	mAP
Inmaduro	84.3 %	81.7 %	86.2 %
MMaduro	77.3 %	78.3 %	81.6 %
Maduro	84.1 %	84 %	88.1 %
	Custom YOLOv5 (Data aumentada)		

Tabla 11: Métricas de desempeño utilizando Custom YOLOv5.

En general las métricas de desempeño de Custom YOLOv5 son superiores al 70 %, inicialmente con la data original (color morado), la clase **Inmaduro** es la que tiene mayor precisión y la clase **Maduro** tiene un mAP igual al 88.4 %, superior a las otras clases.
Con la data aumentada (color rojo)(se generan instancias para la clase medio maduro e inmaduro), se aprecia la mejora en las métricas, en este caso la clase **Inmaduro** es la con mayor precisión (84.3 %) y la clase **MMaduro** aumento la precisión en un 11 %. Obteniendo un mAP superior al 81 % en todas las clases.
Recordar que el menor número de instancias generadas fue para la clase de mediana madurez (MMaduro) e inicialmente la clase con mayor número de instancias creadas es la clase de mayor madurez (Maduro), las que posteriormente son igualadas con las instancias de la clase inmaduro. Y sabemos que un arbusto de arándano presenta niveles de madurez intermedios y diversos, para nuestro caso el mayor número de arándanos en general es inmaduro y maduro.
Con el aumento de la data se obtiene un mAP más equilibrado ya que existe una brecha de 5 a 7 % entre las clases maduro e inmaduro con la clase Mmaduro, la que inicialmente era de 14 a 18 %.

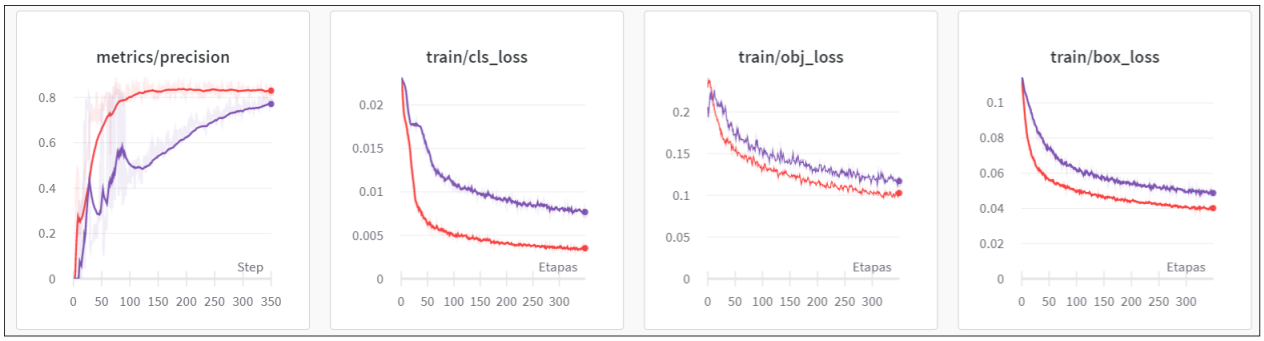


Figura 60: Métricas generales de entrenamiento.

Las métricas generales con la data aumentada se muestran en la figura 60, las ultimas tres gráficas muestran las funciones de pérdida en el entrenamiento. Estas funciones miden la diferencia entre la salida predicha y la salida deseada, es decir, que tan cerca se encuentra de una detección correcta.

- Box loss
  Asociada a medir la posición de las cajas delimitadoras predichas, es decir, analiza que tan ajustada se encuentra la caja delimitadora al objeto detectado, en este caso que tan bien se ajusta a cada arándano detectado.

- Object loss (obj loss)
  Asociada a la cantidad de objetos detectada en cada caja delimitadora, asociada a la predicción de clase.

- Class loss (cls loss)
  Asociada a medir la exactitud de la clasificación de las cajas delimitadoras asociadas a la predicción de la clase. En este caso que tan bien se ajusta a detectar el nivel de madurez de cada arándano (clase).

Estas funciones de pérdida deben ser lo mas cercanas a 0, es decir, si las funciones son cercanas a 0, el entrenamiento tendrá un mejor desempeño, con un error mínimo en las estimaciones del modelo.
Como se muestra en la figura 60, custom YOLOv5 con la data aumentada (amarillo) es superior al que fue entrenado con la data original (morado) en cada una de las graficas generadas.
Luego de superar las 200 épocas cada una de las gráficas se acerca mayormente a 0 y en especial la tercera gráfica *cls_loss* es la mas cercana a 0. Lo que implica que el mejor desempeño de custom YOLOv5 esta asociado a la identificación de clases. Lo que demuestra la fiabilidad y precisión en la detección de clases.
La primera gráfica (precisión) muestra un buen desempeño, en especial con la data aumentada la precisión alcanzada se asienta en 85 %.
Y el recall cercano al 82 %, por lo que se espera que la mayor parte de las etiquetas predichas sean correctas.

4.1.2. Testeo en imágenes

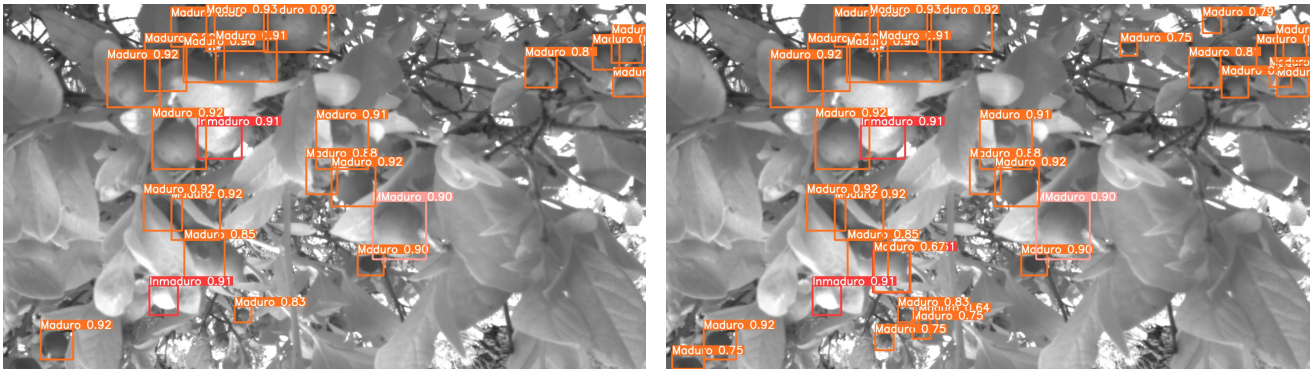
((a)) Confidence: 0.8 ((b)) Confidence:0.6

Figura 61: Detección de arándanos con distintos valores de confidence en imagen RGB.

Figura	ID	Clase	Detectados	Total
61	a	Inmaduro	9 arándanos	31 arándanos
		Mmaduro	2 arándanos	
		Maduro	20 arándanos	
	b	Inmaduro	9 arándanos	31 arándanos
		Mmaduro	2 arándanos	
		Maduro	20 arándanos	
			Observados	32 arándanos

Tabla 12: Detección de arándanos a diferentes grados de confidence, a) conf: 0.8 y b)conf: 0.6.

La figura 61 muestra que al cambiar el nivel de confidence (confianza), el algoritmo es capaz de detectar una cantidad igual de arándanos y su clase respectiva, esto se explica ya que el algoritmo esta totalmente seguro en que estado de maduración se encuentra, por lo que al variar el valor de confidence a un valor más alto, las detecciones siguen siendo las mismas.
El valor de confidence se define de acuerdo a la aplicación y al tipo de imagen que tendrá de entrada el algoritmo, en este caso se selecciona el valor de confidence 0.6 y 0.8, ya que gran parte de las detecciones estan por sobre el 0.6.
Revisando la figura anterior, la imagen a) detecta un total de 31 arándanos en un tiempo de 0.020 [s] y en b) detecta 31 arándanos en un tiempo de 0.017 [s], y se distinguen a simple vista **32 arándanos observados**. Existe una diferencia entre cada detección de 1 arándano, ya que la detección incluye dos arándanos en una sola detección. En ambos casos, con un confidence igual a 0.6 o 0.8 todas las detecciones fueron correctas, solo detecta dos arándanos en un único cuadro delimitador. Por lo tanto, el algoritmo fue capaz de identificar la clase de arándanos de forma correcta y con un nivel de confianza superior al 0.8, donde la detección con confidence mas bajo fue de 0.86 y la con mayor valor de confidence es de 0.96.



((a)) Confidence: 0.8 ((b)) Confidence: 0.6

Figura 62: Detección de arándanos con distintos valores de confidence en imagen NIR.

Figura	ID	Clase	Detectados	Total
62	a	Inmaduro	2 arándanos	24 arándanos
		Mmaduro	1 arándanos	
		Maduro	21 arándanos	
	b	Inmaduro	3 arándanos	34 arándanos
		Mmaduro	1 arándanos	
		Maduro	30 arándanos	
			Observados	35 arándanos

Tabla 13: Detección de arándanos con diferentes grados de confidence, a)conf:0.8 y b)conf:0.6

Para el siguiente caso analizamos la respuesta del algoritmo con una imagen en el espectro del infrarrojo cercano (NIR), en la figura 62 las detecciones entre a) y b) son diferentes ya que con un valor de confidence de 0.8 (caso a) en 0.020 [s] detecta un total de 24 arándanos, con solo detecciones incorrectas en la clase medio maduro ya que no existen instancias de esta clase en la imagen. Pero con un valor de confidence de 0.6 (caso b) en 0.020 [s] detecta un total de 34 arándanos con 2 detecciones incorrectas, ya que genera dos cuadros delimitadores en un arándano (maduro e inmaduro) y detecta una baya medio maduro, cuando no existe ninguno en la imagen.
En este caso, el valor de confidence tiene una gran influencia en el número de detecciones, ya que detecta 10 arándanos más que con un valor de confidence mayor.
Estas imágenes son útiles en este estudio, por el hecho de que la vegetación absorbe la luz visible y refleja la luz infrarroja en función de la salud de la planta. Por lo que, en las imágenes capturadas en el espectro NIR las oclusiones generadas por la luz, sombra y solapamiento de bayas son casi imperceptibles y afectan en un grado menor, captando lo esencial y mejorando el rendimiento del algoritmo en el caso de contar el número de arándanos.

- Matriz de confusión

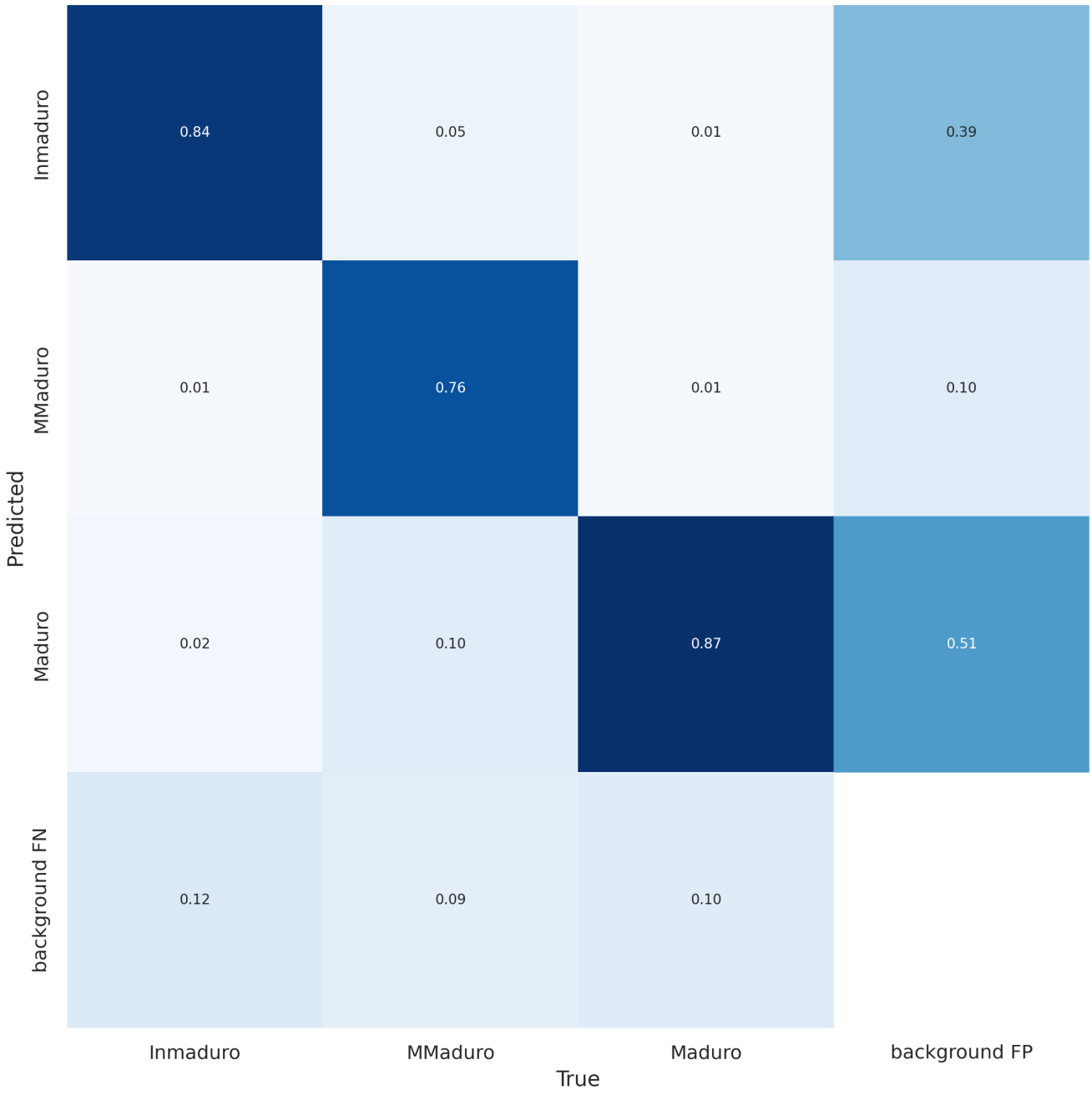


Figura 63: Matriz de confusión de custom YOLOv5, utilizando imágenes VIS-NIR en conjunto.

La matriz de confusión de custom YOLOv5 muestra el rendimiento del modelo de clasificación y el número de predicciones correctas por cada clase (también las incorrectas).
Predicted señala que clase puede predecir el modelo y True señala la clase real, formando una matriz ($4x4$), inicialmente consideramos la matriz de $3x3$ que considera solo las clases Inmaduro, MMaduro y Maduro.
las posiciones esperadas son (1,1) para la clase inmaduro que tiene un valor 0.84, lo que indica que la clase inmaduro tiene el mayor número de predicciones correctas.
La posición (2,2) para la clase MMaduro que tiene un valor de 0.76, por lo que la mayor parte de las predicciones serán correctas, pero en menor cantidad que en la clase anterior, recordemos que mientras más cercano a 1 el valor tendrá mayor número de aciertos.
La última posición (3x3) para la clase Maduro con un valor de 0.87 lo que señala un mayor número de predicciones correctas.
Por otro lado, el borde de la matriz señala que tan frecuente puede predecir predicciones falsas con respecto al fondo, en este caso, tenemos los FP que señala cuando el modelo predice de manera errónea la clase positiva cuando realmente es negativa y FN señala cuando el modelo predice incorrectamente una clase negativa cuando realmente es positiva.
Por lo tanto, el modelo tendrá mayor número de falsos positivos que falsos negativos y en las clases maduro e inmaduro tendrá mayor número de predicciones incorrectas, esto se debe a que los arándanos inmaduros se mezclan con la naturaleza (hojas, ramas y arbusto) y en el caso de la clase maduro se confunden con las sombras y por sobre todo en las variaciones de tonalidades en el caso de las imágenes NIR.

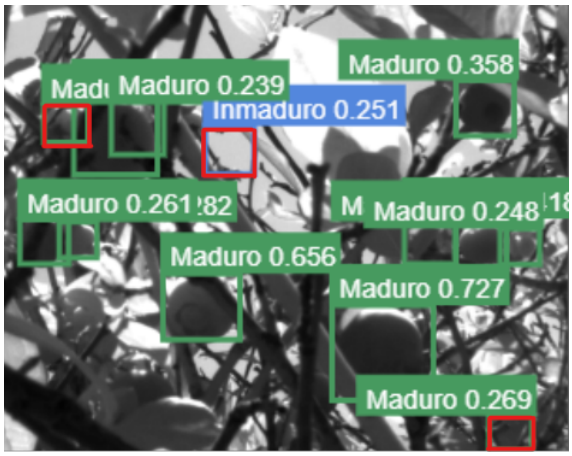


Figura 64: Detecciones en imagen NIR en la etapa 70 de entrenamiento.

La figura 64 muestra algunas detecciones luego de terminar la etapa 70 de entrenamiento, estas detecciones se confunden con clases incorrectas, las que se muestran marcadas con rojo.
En este caso se confunden arándanos maduros e inmaduros con el fondo y con bayas oscurecidas por la sombra. Lo que mejora al continuar el entrenamiento y estableciendo el valor de confidence adecuado.

4.2. YOLOR

YOLOR: You Only Learn One Representation, propone una red unificada multitarea que aprende de una representación general que combina dos aspectos de diferente complejidad como lo son el conocimiento implícito y explicito. Estas dos secciones combinadas son capaces de mejorar el rendimiento a través de inferencias diferenciadoras comparadas con un modelo tradicional, estas inferencias si bien mejoran la precisión, cambian la forma de analizar y procesar una entrada por lo que requieren un coste adicional de procesamiento.
El conocimiento explícito se puede considerar como definiciones claras e imágenes con anotaciones claras o metadatos correspondientes a esas imágenes. Y el conocimiento implícito se adquiere inconscientemente en las capas profundas, es decir no corresponde a las observaciones o etiquetas entregadas inicialmente.
Se menciona en [37] que se requiere de un poco menos de diez mil parámetros y cálculos adicionales, y es por esto que en esta sección se trabajara con la versión inicial de YOLOR (YOLOR p6).

- YOLOR p6
  Este modelo esta compuesto por 655 capas interconectadas y 36.849.216 de parámetros

Backbone		
Filters	Type	Activation
64	Downsample - Convolutional	Silu
128	Downsample - Convolutional	Silu
64	Split - Convolutional	Silu
64	Split - Convolutional	Silu
64	Residual - Convolutional	Silu
64	Residual - Convolutional	Silu
64	Residual - Convolutional	Silu
64	Residual - Convolutional	Silu
64	Transition First - Convolutional	Silu
64	Transition Last - Convolutional	Silu

Tabla 14: Base de modelo YOLOR-p6 (Backbone).

La estructura mostrada anteriormente se repite 4 veces más para luego pasar a la etapa max-pooling y full conectada.

4.2.1. Métricas de desempeño

Las métricas mostradas en las figuras 65 y 66 se obtuvieron luego de entrenar YOLOR-p6 en 300 épocas (etapas), para estos dos casos se utilizó un batch size igual a 20, ya que con un batch size superior se superan los recursos disponibles en Google Colab, y se traduce en 40 etapas de entrenamiento por cada época y 4 etapas de validación.

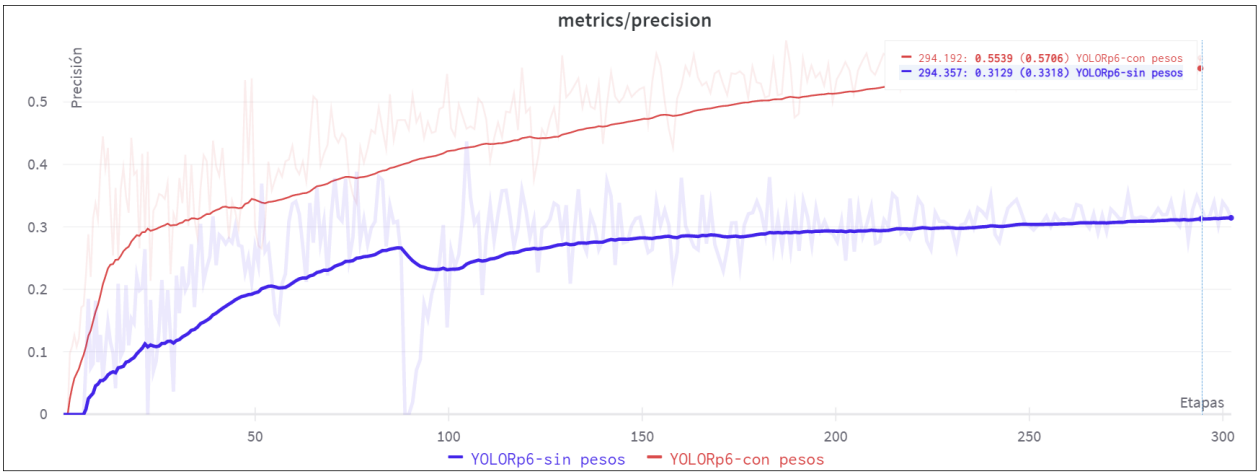


Figura 65: Gráfica comparativa de métrica precisión: pre entrenado y sin carga de pesos.

Como se muestra en las gráficas 65 y 66, la señal de color azul es YOLOR-p6 sin cargar los pesos pre-entrenados del modelo y la señal de color rojo es YOLOR-p6 entrenado a 300 épocas con los pesos pre-entrenados del modelo cargado.
En general las métricas de precisión y recall muestran un mejoramiento con una pendiente positiva, es decir, a mayor numero de épocas la precisión aumenta y el recall se asienta en un valor fijo cercano al 89 % en el caso de YOLOr-p6 pre-entrenado, por el contrario la gráfica de la precisión sigue aumentando, para alcanzar el máximo peak pasando la época 280 con una precisión del 58 %.
YOLOR demostró que tiene una precisión mas baja que el recall, pero una media de precisión (maP) igual al 86 %, mejor que la precisión por si sola, esto dado que se calcula en base al valor de la precisión y el recall.

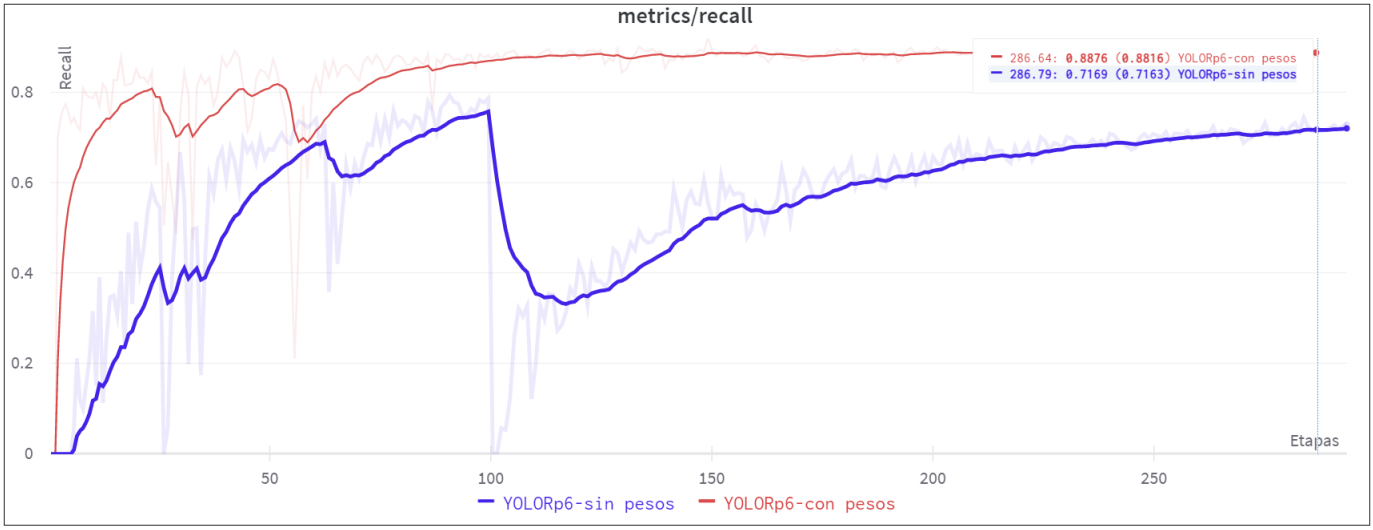


Figura 66: Gráfica comparativa de métrica recall: pre-entrenado y sin carga de pesos.

Epoca	Precisión	Recall	mAP
100	30.9 %	75.7 %	66.9 %
200	47.14 %	85.4 %	82.3 %
290	31.8 %	71.3 %	61.3 %
	YOLORp6-sin pesos		

Epoca	Precisión	Recall	mAP
100	34.1 %	65.4 %	55.1 %
200	53.16 %	88.6 %	85.5 %
290	55.11 %	89.2 %	85.6 %
	YOLORp6-con pesos		

Tabla 15: Métricas con respecto a la época de YOLOR-p6 sin y con pesos (pre-entrenado).

La precisión promedio de YOLOR-p6 con los pesos pre-entrenados (color rojo, YOLORp6-con pesos) alcanza el valor máximo en la época 290, con un 85.6 % de mAP y como se muestra en la tabla 15, la precisión sigue mejorando pasando el numero de épocas, lo que nos sugiere que si aumentamos las épocas la precisión podría mejorar gradualmente y el porcentaje de recall podría lograr asentarse en un valor fijo cercano al 90 %.

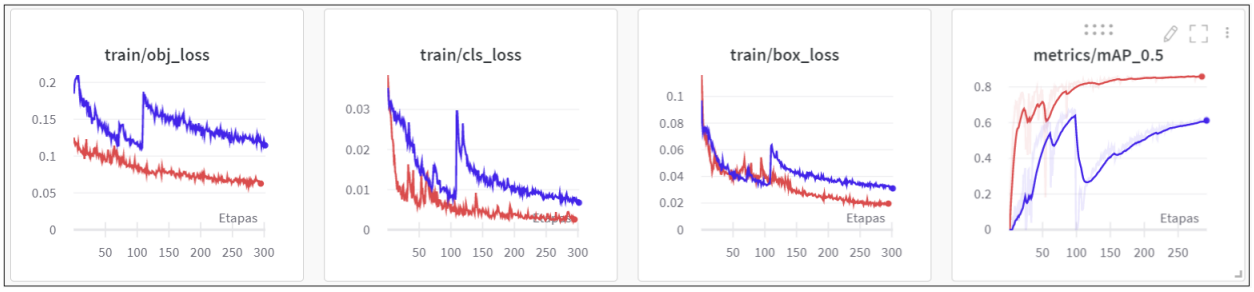


Figura 67: Métricas generales de entrenamiento.

Las métricas generales de YOLOR-p6 con la data aumentada se muestran en la figura 67, las primeras tres gráficas muestran las funciones de pérdida en el entrenamiento. Estas funciones miden la diferencia entre la salida predicha y la salida deseada, es decir que tan cerca se encuentra de una detección correcta, por esto mismo, se espera que estas funciones de pérdida tiendan a 0 o sean lo mas cercanas a 0.
Las primeras tres gráficas de la figura 67 muestran que luego de superar las 250 épocas cada una tiende a disminuir lo más cercano posible a 0, pero en nuestro caso la segunda gráfica (Cls-classifcation) es la que más cercana a 0 se encuentra, esto nos indica que la probabilidad de que el algoritmo falle al detectar una clase es casi nula. Como también en la tercera gráfica (Box) se asienta en 0.02 aproximadamente, por lo que, el desempeño para generar una caja delimitadora en el objeto podrá ajustarse proporcionalmente al objeto detectado. Y se muestra como la señal roja (YOLORp6-con pesos) tiende a acercarse mas a cero que la señal azul, lo que se traduce en una mejora en el entrenamiento.
En lineas generales el desempeño de YOLOR-p6 es preciso, pero dado que se obtiene un recall alto (89 %) y una precisión relativamente baja (56 %), se vuelve más probable que existan detecciones falsas, por lo que se espera una detección precisa de la clase pero que también incluirá muestras de clases que no correspondan.
Sabemos que, un algoritmo tendrá en su mayoría detecciones correctas si tiene un valor de precisión y recall altos, en nuestro caso se esperan muchos resultados con etiquetas predichas incorrectas y/o que no existan detecciones, ya que el valor de la precisión es un 38 % inferior al recall. Pero el testeo en imágenes será la prueba final para sacar conclusiones definitivas en cuanto al desempeño.

4.2.2. Testeo en imágenes

((a)) Confidence: 0.8 ((b)) Confidence: 0.6

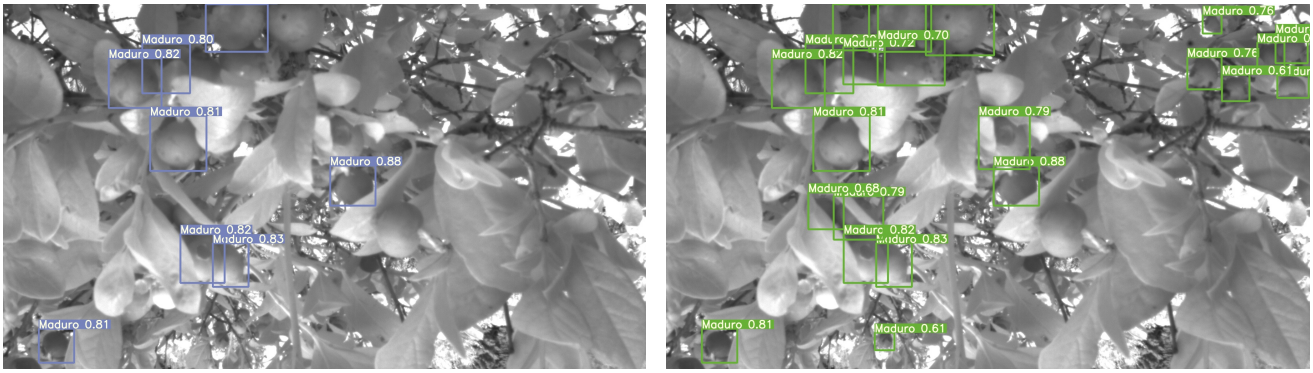
Figura 68: Detección de arándanos con distintos valores de confidence.

Figura	ID	Clase	Detectados	Total
68	a	Inmaduro	7 arándanos	24 arándanos
		Mmaduro	0 arándanos	
		Maduro	17 arándanos	
	b	Inmaduro	8 arándanos	29 arándanos
		Mmaduro	0 arándanos	
		Maduro	21 arándanos	
			Observados	32 arándanos

Tabla 16: Detección de arándanos con confidence : a) 0.65 y b) 0.41.

La figura 68 y tabla 16 muestran que una variación en el valor de confidence mejora ampliamente la detección de arándanos por clase, en este caso con un valor alto de confidence (0.8) detecta un total de 24 arándanos en un tiempo de 0.027[s], donde la clase con más detecciones es la de mayor madurez y para un valor de confidence inferior (0.6) detecta 29 arándanos en un tiempo de 0.027[s] de un total de 32 arándanos observados y nuevamente la clase con mayor detecciones es la de mayor madurez.

Porcentualmente, se observa que al reducir el valor de confidence mejora aproximadamente en un 20 % el número de detecciones. Esto se debe a que a un bajo nivel de confianza el algoritmo muestra mayor número de posibilidades y ajusta estas a diferentes clases. Y aunque con confidence igual a 0.6 las detecciones aumentan, dos detecciones son erróneas, ya que asigna la clase maduro a una baya medio madura y detecta dos arándanos maduros en un solo cuadro delimitador. En general detecta todos arándanos de manera precisa y con un confidence máximo de 0.95.



((a)) Confidence: 0.8 ((b)) Confidence: 0.6

Figura 69: Detección de arándanos con distintos valores de confidence en imagen NIR.

Figura	ID	Clase	Detectados	Total
69	a	Inmaduro	0 arándanos	8 arándanos
		Mmaduro	0 arándanos	
		Maduro	8 arándanos	
	b	Inmaduro	0 arándanos	22 arándanos
		Mmaduro	0 arándanos	
		Maduro	22 arándanos	
			Observados	35 arándanos

Tabla 17: Detección de arándanos con confidence : a) 0.65 y b) 0.41.

La figura 69 y tabla 17 demuestran el desempeño de YOLOR-p6 en imágenes en el espectro infrarrojo cercano, estas pruebas requieren un análisis más minucioso, ya que la madurez asociada a cada fruto se diferencia a simple vista por la tonalidad de gris de cada uno, pero en múltiples casos, el ojo humano tiende a confundirse con este tipo de imágenes.
Para un valor de confidence igual a 0.8 detecta 8 arándanos en 0.027[s] y el mayor número de detecciones lo tiene la clase maduro. Y ante una variación del valor de confidence (confidence: 0.6) la detección es de 22 arándanos en 0.026[s], con solo detecciones de la clase maduro y no detecta los arándanos de la clase inmaduro presentes en la imagen. Por lo que el valor de confidence tiene un gran impacto en este análisis.

4.3. Comparativa entre YOLOv5 y YOLOR

En nuestro caso utilizamos YOLOv5 y YOLOR para la detección de arándanos y su respectiva etapa de maduración.
Como estos tienen algunas similitudes, son comparables, por lo que se consideran dos pruebas generales, la primera desarrollada anteriormente y para la segunda se seleccionan dos imágenes, una imagen con un grado de dificultad mucho mayor, con un campo mas amplio de captura y una imagen ya desarrollada anteriormente.
Una diferencia clara, es el tiempo que demora en entrenar YOLOv5(6 horas aproximadamente) y YOLOR (20 horas aproximadamente), con una diferencia cercana a las 14 horas.

((a)) Custom YOLOv5 ((b)) YOLOR-p6

Figura 70: Detección de arándanos con confidence: 0.6.

Figura	ID	Clase	Detectados	Total
70	a	Inmaduro	9 arándanos	31 arándanos
		Mmaduro	2 arándanos	
		Maduro	20 arándanos	
	b	Inmaduro	8 arándanos	29 arándanos
		Mmaduro	0 arándanos	
		Maduro	21 arándanos	
			Observados	32 arándanos

Tabla 18: Detección de arándanos con confidence : a) Custom YOLOv5 y b) YOLOR-p6.

Como se muestra en la tabla 18, las detecciones con un valor de confidence igual a 0.6 son similares, ya que en el caso de Custom YOLOv5 (a) detecta un total de 31 arándanos y la clase con mayor número de detecciones es maduro con 20 arándanos. Por otro lado, YOLOR-p6 (b) detecta un total de 29 arándanos y la clase con mayor numero de detecciones es maduro con 21 arándanos. La diferencia mas notoria esta en el valor de confidence de cada detección.

((a)) Custom YOLOv5 ((b)) YOLOR-p6

Figura 71: Detección de arándanos con confidence: 0.6.

Figura	ID	Clase	Detectados	Total
71	a	Inmaduro	102 arándanos	164 arándanos
		Mmaduro	8 arándanos	
		Maduro	54 arándanos	
	b	Inmaduro	36 arándanos	72 arándanos
		Mmaduro	6 arándanos	
		Maduro	30 arándanos	
			Observados	166 arándanos

Tabla 19: Detección de arándanos con confidence : 0.41, a) Custom YOLOv5 y b) YOLOR-p6.

La figura 71 muestra la capacidad de la version personalizada de YOLOv5 y YOLOR-p6 de detectar arándanos con un valor de confidence igual a 0.6, en este caso YOLOv5 es capaz de detectar un total de 164 arándanos y YOLOR-p6 detecta un total de 72 arándanos, de un total de 166 arándanos. Este tipo de imágenes con un campo de visión más amplio presenta arándanos de tamaño muy pequeño y también muestra arándanos ocluidos, por lo que es de una dificultad alta. Notamos que YOLOv5 presenta valores de confidence superiores a 0.7 a 0.95, con la mayoría de las detecciones superiores a 0.9 y YOLOR-p6 varia desde 0.6 hasta 0.93, con la mayoría de las detecciones superiores a 0.8.
Es concluyente que YOLOv5 es muy superior a YOLOR generando un 45 % más de etiquetas en el primer caso, esto se debe a la diferencia en las métricas, ya que YOLOv5 obtiene un mAP cercano al 85 % con una precisión y recall altos y YOLOR alcanza un máximo mAP cercano al 85 % pero con un recall alto y una precisión baja, por esto mismo inferimos que un numero mayor de detecciones correctas esta relacionado con un valor alto de precisión y recall alto.

Además, YOLOR se entrena en 20 horas y YOLOv5 en casi 6 horas, por lo que este último logra un mejor desempeño con menor tiempo de entrenamiento, esto se debe a que YOLOR requiere un costo computacional mayor, ya que tiene un numero de parámetros más alto.

4.4. Mask R-CNN

Los científicos e investigadores de datos de Facebook IA fueron pioneros en la arquitectura Mask R-CNN, que es capaz de crear una mascara de pixeles para cada objeto en una imagen, este algoritmo agrega otra rama a la salida del algoritmo Faster R-CNN ya existente, Faster R-CNN genera dos cosas para cada objeto en la imagen, su clase y las coordenadas del cuadro delimitador, y la tercera rama que agrega Mask R-CNN es la que genera la máscara del objeto.

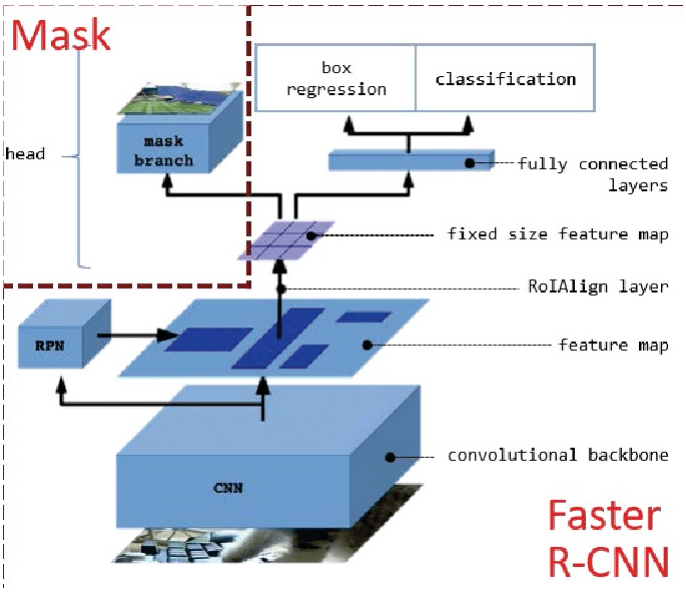


Figura 72: Diagrama de arquitectura Mask R-CNN.
Fuente: Estudio [48].

4.4.1. Funcionamiento

El primer paso es tener una imagen como entrada, la cual será recibida por la arquitectura ResNet 101 (Red neuronal convolucional), ResNet 101 devolverá un mapa de características para esa imagen. A estos mapas le aplicaremos la red propuesta de región (RPN), esto nos devolverá la propuesta de objetos y su puntaje de objetividad, es decir nos dirá si hay un objeto en la región seleccionada o no. A la propuesta de objetos le aplicaremos una capa de agrupación RoI (Región de interés), esto se hace para reducir todas las propuestas al mismo tamaño porque las regiones obtenidas del RPN pueden tener diferentes formas. Para finalizar la propuesta se pasa a una capa completamente conectada para que se predigan las etiquetas de clase y los cuadros delimitadores.

Hasta este punto es similar a como funciona Faster R-CNN, ahora viene la diferencia entre ambos. Primero se calcula la región de interés para poder reducir el calculo, entonces calculamos la intersección sobre la unión (loU) con la siguiente formula.

- IoU : $\frac{Áreadelaintersección}{Áreadelaunión}$

Si IoU es mayor a 0.5 se considera como región de interés, si es menor a 0,5 descartamos esa región, este proceso se hace para todas las regiones. Una vez obtenido los RoI basado en los valores de loU, agregamos una rama de máscara a la arquitectura, esta nos devolverá la máscara de segmentación para cada región que contiene un objeto.

Figura 73: Imagen de arándanos con sus respectivas máscaras.
Fuente: Elaboración propia.

La arquitectura es grande; solo se muestran las capas principales a continuación. La capa final denominada *mrcnn_mask* solo devuelve las máscaras de los 100 mejores ROI (Region of interest).

```
fpn_c5p5               (Conv2D)
fpn_c4p4               (Conv2D)
fpn_c3p3               (Conv2D)
fpn_c2p2               (Conv2D)
fpn_p5                 (Conv2D)
fpn_p2                 (Conv2D)
fpn_p3                 (Conv2D)
fpn_p4                 (Conv2D)
In model:  rpn_model
    rpn_conv_shared        (Conv2D)
    rpn_class_raw          (Conv2D)
    rpn_bbox_pred          (Conv2D)
mrcnn_mask_conv1       (TimeDistributed)
mrcnn_mask_bn1         (TimeDistributed)
mrcnn_mask_conv2       (TimeDistributed)
mrcnn_mask_bn2         (TimeDistributed)
mrcnn_class_conv1      (TimeDistributed)
mrcnn_class_bn1        (TimeDistributed)
mrcnn_mask_conv3       (TimeDistributed)
mrcnn_mask_bn3         (TimeDistributed)
mrcnn_class_conv2      (TimeDistributed)
mrcnn_class_bn2        (TimeDistributed)
mrcnn_mask_conv4       (TimeDistributed)
mrcnn_mask_bn4         (TimeDistributed)
mrcnn_bbox_fc          (TimeDistributed)
mrcnn_mask_deconv      (TimeDistributed)
mrcnn_class_logits     (TimeDistributed)
mrcnn_mask             (TimeDistributed)
```

Figura 74: Arquitectura de Mask R-CNN.

Una vez clonado el repositorio de Mask R-CNN en Google colab, tenemos que cargar las imágenes con sus respectivas anotaciones, en la figura 75 podemos ver imágenes cargadas al azar con sus anotaciones, gracias a esto confirmamos que las imágenes se cargaron correctamente.

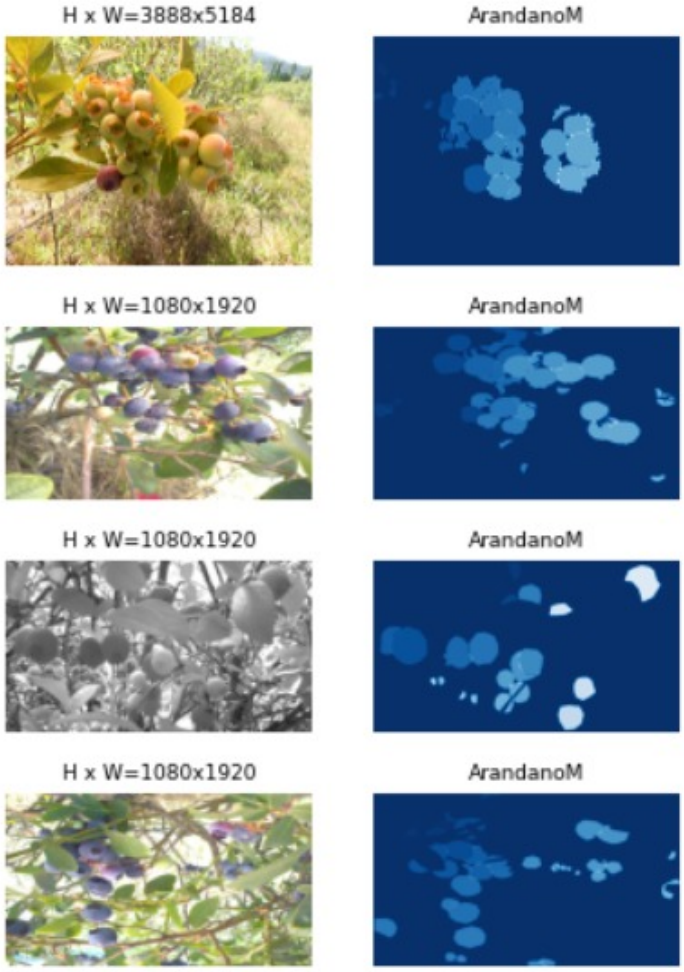
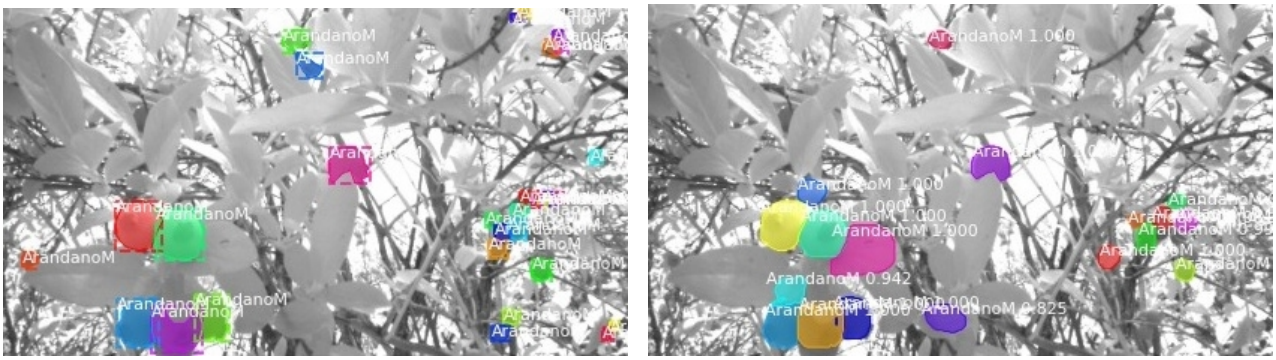


Figura 75: imágenes con sus respectivas etiquetas.
Fuente: Elaboración propia.

Se separaron las imágenes en 70 % entrenamiento y 30 % de validación.
Una vez iniciado el entrenamiento, los pesos de cada época se guardan en una carpeta en drive, en una primera instancia el modelo se entrenó con 8 épocas y 500 etapas por época, con 300 etapas de validación, se definen estos parámetros ya que requiere un costo computacional elevado para ejecutarse, y los resultados se muestran en la figura 76, en el lado izquierdo tenemos la imagen con anotaciones manuales y en el lado derecho las máscaras generadas por el modelo. Como podemos apreciar en las imágenes NIR, el modelo detecta bastantes arándanos, pero tiene problemas con los arándanos más pequeños, además presenta un par de errores donde las hojas son identificadas como arándanos. En las imágenes RGB, el modelo se comportó de mejor forma, detecto todos los arándanos que fueron etiquetados manualmente y además identifico arándanos que no estaban etiquetados por lo desenfocado y borroso de la imagen (77).



((a)) ((b))

Figura 76: Imagen NIR: a) Anotaciones generadas manualmente, b) Detecciones generadas.

((a)) ((b))

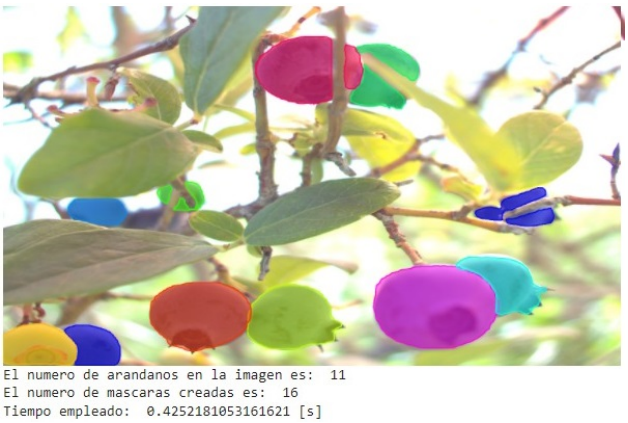
Figura 77: Imagen VIS: a) Anotaciones generadas manualmente, b) Detecciones generadas.

La tabla 20, muestra que el modelo entrenado presenta falencias, por sobre todo en el primer caso, ya que detecta solo 18 arándanos, de los 30 arándanos anotados, es decir el modelo detecta solo el 60 % de arándanos, lo que sugiere mejorar el modelo. Por otro lado en la imagen VIS, es capaz de detectar el mismo número de los arándanos que fueron etiquetados.

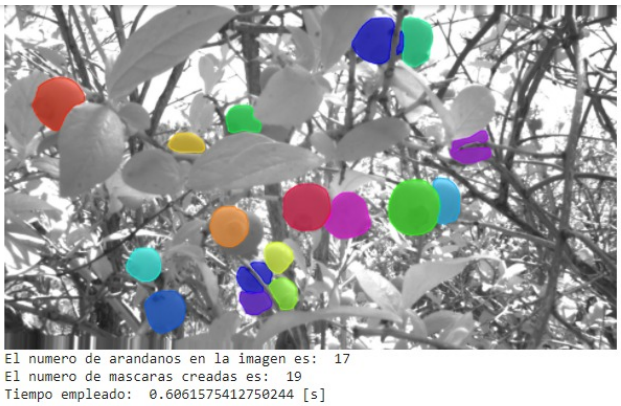
Figura	Etiquetados	Detectados
76	30 arándanos	18 arándanos
77	10 arándanos	10 arándanos

Tabla 20: Comparación de detecciones en imágenes VIS-NIR

El modelo se volvió a entrenar con un mayor número de imágenes y épocas, en una segunda instancia con 18 épocas de 300 etapas por cada época, 150 etapas de validación y además una reducción en el nivel de confianza (confidence) de 0.9 a 0.7 para que detectara arándanos a objetos que el modelo no estaba tan seguro si lo eran. El modelo alcanzo un valor máximo del 70 % de precisión media. Y los resultados mostrados en la figura 78 demuestran una mejora en la máscara, cubre de mejor manera el arándano, y lo más destacado es que genera más mascaras que arándanos en la imagen, esto se debe a que hay arándanos que se muestran partidos en dos porque hay ramas que están por delante (fucsia y azul en la figura 78 imagen (a)), pero el modelo es capas de inferir que se trata del mismo arándano y desecha la detección de la rama, la cual no se considera para la detección de la clase.



((a))



((b))

Figura 78: Detección de arándanos en imágenes VIS-NIR.

4.4.2. Métricas de desempeño

El desempeño de Mask R-CNN se estabiliza con el pasar de las épocas, en nuestro caso al llegar a las 18 épocas el valor de mAP alcanza un valor de 67.5 %, y el mayor peak se alcanza al llegar a la época 16 alcanzando una precisión media del 70 % (mAP).
Para entender de mejor manera las métricas de la figura 79 definimos los siguientes ítems.

- rpn class loss : Pérdida del clasificador de anclaje RPN.
- rpn bbox loss : Gráfico de pérdida del cuadro delimitador RPN.
- class loss : Pérdida para el clasificador de Mask R-CNN.
- bbox loss : Pérdida para el refinamiento de la caja delimitadora Mask R-CNN.
- mask loss : Pérdida de entropía cruzada binaria de la máscara para las las máscaras generadas.

Cada una de estas métricas de pérdida es la suma de todos los valores de pérdida calculados individualmente para cada una de las regiones de interés. La métrica de pérdida general dada en el registro es la suma de las otras cinco pérdidas (se puede verificar sumándolas) según lo definido por los autores de Mask R-CNN.
Los valores de pérdida de clasificación dependen básicamente de la puntuación de confianza de la clase verdadera, por lo tanto, las pérdidas de clasificación reflejan la confianza del modelo al predecir las etiquetas de clase, o en otras palabras, qué tan cerca está el modelo de predecir la clase correcta.
En el caso de class loss, todas las clases de objetos están cubiertas, mientras que en el caso de rpn class loss la única clasificación que se hace es etiquetar las cajas de anclaje como primer plano o fondo (que es la razón por la cual esta pérdida tiende a tener valores más bajos, ya que conceptualmente solo hay 'dos clases' de las que se pueden predecir).
En el caso de rpn bbox loss muestra que tan bueno es el modelo para localizar objetos dentro de la imagen. Por otro lado bbox loss muestra qué tan bueno es el modelo para predecir con precisión el área (s) dentro de una imagen correspondiente a los diferentes objetos que están presentes.

La pérdida de máscara, de manera similar a la pérdida de clasificación, penaliza las clasificaciones binarias incorrectas por píxel (primer plano / fondo, con respecto a la etiqueta de clase verdadera). Se calcula de manera diferente para cada una de las regiones de interés: Mask R-CNN codifica una máscara binaria por clase para cada uno de los RoIs, y la pérdida de máscara para un RoI (región de interés) específico se calcula basándose solo en la máscara correspondiente a su clase verdadera, lo que evita que la pérdida de máscara se vea afectada por las predicciones de clase.

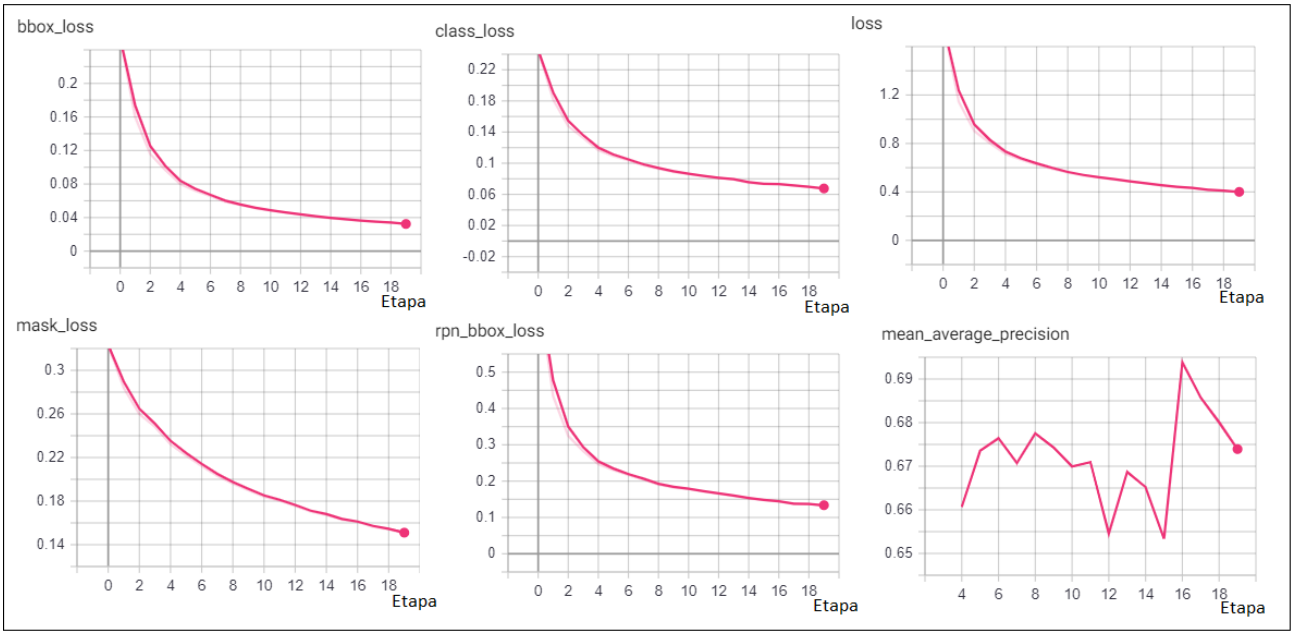


Figura 79: Métricas de desempeño.

Las métricas de pérdida (loss, class loss, bbox loss, mask loss y rpn bbox loss) se espera que sean lo más cercanas a cero, dado que estas métricas indican el valor de perdida mientras se entrena el modelo. En nuestro caso la gráfica bbox loss es la más cercana a cero al llegar a la época 18, esto nos indica que mask R-CNN ajusta con una gran precisión las cajas delimitadoras iniciales como también identifica la clase con una alta precisión ya que el valor de pérdida es aproximadamente 0.065. Para el caso de mask loss el valor alcanzado en la época 18 es de 0.15 lo que sugiere una pérdida en la precisión al generar las mascaras en las detecciones, este factor de pérdida lo vemos en el testeo de imágenes, ya que en algunos casos la mascara no se ajusta totalmente a la silueta del arándano.

Aunque la precisión media se genera en base a la intersección sobre la unión y regiones de interés, debemos considerar que las anotaciones en las imágenes de entrenamiento se realizan de forma manual, por lo que añade ciertas imprecisiones al modelo. Es por esto, que la gráfica de mAP no tiende a estabilizarse en un valor fijo, mas bien oscila entre 0.655 y 0.7, a pesar de esto con una precisión media del 70 % mask R-CNN demostró la capacidad de detectar y contar arándanos con gran precisión, siendo capaz de diferenciar cuando dos mascaras correspondían al mismo arándano.

4.5. Comparativa general

Para generalizar el testeo practico con imágenes, utilizaremos un set de datos idéntico para testear cada uno de los algoritmos, como documentación usamos tres imágenes para analizar cuantos arándanos detecta por imagen cada uno de los algoritmos.

Algoritmo \Métrica	**Accuracy**	**Recall**	**mAP**
Custom YOLOv5	85 %	82 %	85.3 %
YOLOR	55.11 %	89.2 %	85.6 %
Mask R-CNN	-	-	70 %

Tabla 21: Métricas generales por cada algoritmo.

La tabla 21 muestra que YOLOv5 es superior a YOLOR en la métrica accuracy y YOLOR es superior a Custom YOLOv5 y Mask R-CNN en la métrica Recall y también en la métrica mAP, pero la diferencia entre YOLOv5 y YOLOR es de 0.3 %.

Figura 80: Muestra de 3 imágenes de prueba.

Algoritmo \Imagen	**1 (DSCN3025)**	**2 (DSCN2962)**	**3 (DSCN2959)**	**Total**
Custom YOLOv5	23 arándanos	48 arándanos	76 arándanos	147 arándanos detectados
YOLOR-p6	22 arándanos	43 arándanos	60 arándanos	125 arándanos detectados
Mask R-CNN	20 arándanos	35 arándanos	46 arándanos	101 arándanos detectados
Arándanos por imagen	28 arándanos	56 arándanos	77 arándanos	161 arándanos observados

Tabla 22: Arándanos detectados por imagen con confidence: 0.6.

YOLOv5 en promedio demora 0.018[s] en realizar la detección en una imagen, YOLOR realiza la detección en una imagen en un tiempo de 0.028[s] y Mask R-CNN en promedio demora 7[s] en realizar la detección en una imagen llegando hasta 14 [s] en casos puntuales. En este

mismo orden, YOLOv5 detecta un total de 147 arándanos en las tres imágenes testeadas, lo sigue YOLOR-p6 con un total de 125 arándanos detectados y finalmente Mask R-CNN con 101 arándanos detectados.
Estas tres imágenes presentan 161 arándanos observados, por lo que el más cercano a este número es YOLOv5 y con un tiempo de testeo por imagen inferior comparado con los otros dos algoritmos.

Figura 81: Detecciones generadas en la imagen 3 por Custom YOLOv5.

La versión personalizada de YOLOv5 es superior a YOLOR y Mask R-CNN, lo que se aprecia de mejor forma en la tercera foto, ya que solo existe un diferencial de 1 arándano entre los arándanos detectados y los observados, y como se aprecia en la figura 81 todas las detecciones generadas por Custom YOLOv5 están correctas y con un confidence superior a 0.9 en gran parte de las detecciones.

CONCLUSIONES

En el presente proyecto de título exploramos el potencial de utilizar métodos de aprendizaje automático junto con imágenes multiespectrales en un ambiente natural al aire libre para la clasificación multiclase y conteo de arándanos de la especie Highbush Legacy en diferentes etapas de crecimiento con técnicas no destructiva.

Se recolectaron imágenes multiespectrales en un entorno tan desafiante como puede ser un huerto agrícola, en la Región de Ñuble comuna de Quillón, donde se utilizaron los espectros VIS-NIR, ya que el grado de madurez de un fruto se puede determinar con ayuda de imágenes capturadas en el rango del infrarrojo cercano, estas imágenes nos permitieron extraer información de los cultivos que el ojo humano no puede ver, esto ya que la vegetación absorbe la luz visible y refleja la luz infrarroja en función del estado de la salud de la planta. reduciendo el numero de oclusiones y reflejos en el fruto.

Recolectamos 370 imágenes capturadas con 3 cámaras diferentes 250 imágenes con una Nikon Coolpix B700, 60 imágenes con una Basler acA2440-20gc y 60 imágenes con una Basler acA2000-50gmNIR, para luego aumentar el dataset aplicando transformaciones como rotaciones, recortes y zoom, para finalmente contar con un dataset de 869 imágenes (incluye 96 imágenes de fondo).

Identificamos y contamos tres clases de arándanos: maduros, medio maduros e inmaduros, (por las pocas instancias de medio maduro que teníamos en comparación con las instancias maduro e inmaduro, se generaron imágenes auxiliares que solo tenían instancias media maduro para equiparar las instancias) por medio de los algoritmos YOLOv5 y YOLOR, y con Mask R-CNN identificamos una clase, estas arquitecturas son de última generación en cuanto a detección de objetos en caso de YOLO y segmentación semántica en el caso de Mask R-CNN.

En una primera instancia se utilizaron dos arquitecturas de YOLOv5 para determinar cual de ellas obtenía el mejor resultado, custom YOLOv5 y YOLOv5s, para luego continuar con la que obtuviera el mejor desempeño, para el entrenamiento con el dataset aumentado.

Custom YOLOv5 obtuvo el mejor desempeño con un mAP de 81.3 %, alcanzando un 88.42 % (data original) en la clase maduro, mientras que YOLOv5s obtuvo un mAP de 79 %, siendo la clase maduro su mejor resultado con 85.9 %.
Con el dataset aumentado, custom YOLOv5 mejoro significativamente sus resultados, con un mAP de 85.3 %, siendo la clase maduro su mejor resultado con un 88.1 % de precisión. En cuanto al conteo se probaron dos grados de confianza 0.8 y 0.6, en múltiples imágenes al azar, detectando los mismos arándanos para una imagen RGB y detectando muchos mas arándanos en una imagen NIR para un grado de confianza menor.

YOLOR por su parte alcanzo una precisión máxima de 85.6 %, detectando menos arándanos que custom YOLOv5 en las mismas imágenes. Por lo tanto YOLOv5 es superior a YOLOR generando un 45 % mas de etiquetas, esto se debe a la diferencia en las métricas, ya que custom YOLOv5 tiene un mAP cercano al 85 % con una precisión y recall altos, mientras que YOLOR alcanza un mAP cercano al 85 %,pero con un recall alto y una precisión baja otro punto donde custom YOLOv5 es superior es el tiempo de entrenamiento, YOLOR se entrena en 20 horas aproximadamente mientras que YOLOv5 solo en 6 horas aproximadamente, por lo tanto, necesita menor tiempo de entrenamiento,menor costo computacional y posee un mejor desempeño.

Para Mask R-CNN es importante tener en cuenta que las anotaciones del tipo cuadro delimitador para la detección de objetos, son significativamente más rápidas en YOLO, ya que, para Mask R-CNN las anotaciones se deben dibujar por medio de polígonos, llegando a tardar una hora en imágenes con demasiados arándanos, por lo tanto se debe asignar un tiempo mucho mayor para sus anotaciones. Otra limitante es que al usar Google Colab tenemos una cantidad de RAM limitada a 25 GB, la cual se vio sobrepasada por la cantidad de imágenes utilizadas, además de la cantidad ingente de tiempo necesitada para poder entrenar el modelo, por lo tanto, nos vimos en la obligación de reducir considerablemente el dataset de entrenamiento, razón por la que no se logró sacar el máximo rendimiento pasible al Mask R-CNN. Aun así con 18 épocas de 300 etapas por época y una confianza de 0.7 (confidence) se logró alcanzar el 70 % de precisión media, siendo lo más destacado del modelo el hecho de poder determinar que se trata del mismo arándano cuando este tiene por delante ramas que lo dividen.

Si bien YOLO es uno de los sistemas de detección con mejor proyección utilizando redes neuronales, en vista de los resultados que hemos obtenidos, podemos inferir que en aplicaciones donde las imágenes sean capturadas cercanas al objeto (arándanos en nuestro caso) o con un campo de visión relativamente acotado, funciona sin problemas, confirmamos la hipótesis inicial descrita en estudios anteriores, en los cuales dicen que YOLO tiene dificultades para poder detectar objetos tan pequeños. Mask R-CNN indudablemente es un algoritmo con gran potencial, pero en este caso se vio limitado, por la cantidad de recursos que disponíamos, y aun así obtuvo un buen rendimiento. En términos generales los modelos funcionan y detectan gran cantidad de arándanos en imágenes con un campo de visión acotado y una menor cantidad en imágenes con campos de visión amplio. La estimación de tiempo de cosecha puede ser llevada a cabo con YOLOv5x y Mask R-CNN con una buena precisión, pero pueden ser mejorados con mayor cantidad de imágenes, en especial imágenes que contengan la clase medio maduro, para lograr generar un mayor número de instancias en esta clase.

Tenemos total claridad que esta misma metodología para otro tipo de cultivos donde el fruto sea de mayor tamaño como manzanas o naranjas, tendrá un desempeño superior. Por el momento se debe seguir mejorando y detallando los algoritmos para poder hacer frente de manera más robusta a las oclusiones y a frutos pequeños.

Con este estudio, creamos una nueva base de datos con anotaciones para la detección y segmentación de instancias de arándanos, y el repositorio de Github con el código utilizado, disponible para todo público. Además, de las imágenes testeadas en cada uno de los algoritmos [Test].

REFERENCIAS

[1] A. Franco, A; Nicolau, C; Alcaraz, "ANUARIO ARÁNDANO 2020 - 2021,cambio en los mercados ante aumento en la oferta del hemisferio sur," p. 50, 2021.

[2] J. Pefaur Lepe, "Evolución de la Fruticultura Chilena en los Últimos 20 Años." 2020.

[3] C. Becerra, B. Defilippi, A. France, A. González, J. Hirzel, C. G. Morales, A. Pedreros, J. Riquelme, P. Robledo, and H. Uribe, "Manual de manejo agronómico del arándano," *Boletín INIA*, p. 98, 2017.

[4] C. Yang and W. S. Lee, "Blueberry fruit detection by Bayesian classifier and support vector machine based on visible to near-infrared multispectral imaging," *American Society of Agricultural and Biological Engineers Annual International Meeting 2012, ASABE 2012*, vol. 5, no. 11, pp. 3714–3725, 2012.

[5] W. A. L. Aguirre, "Espectroscopia infrarrojo y técnicas de machine learning y deep learning para la detección y clasificación de arándanos," *Pueblo Continente*, vol. 30, no. 2, pp. 555–570, 2019.

[6] S. Gonzalez, C. Arellano, and J. E. Tapia, "Deepblueberry: Quantification of blueberries in the wild using instance segmentation," *IEEE Access*, vol. 7, pp. 105 776–105 788, 2019.

[7] P. D. M. C. M. Q. Z. T. E. Arnold W Schumann, Negar S Mood, "Detection of Three Fruit Maturity Stages in Wild Blueberry Fields Using Deep Learning Artificial Neural Networks." pp. 1–20, 2019.

[8] W. Zheng, Y. Bai, H. Luo, Y. Li, X. Yang, and B. Zhang, "Self-adaptive models for predicting soluble solid content of blueberries with biological variability by using near-infrared spectroscopy and chemometrics," *Postharvest Biology and Technology*, vol. 169, no. November, 2020.

[9] H. Fiszman and F. Vilella, "PRECIOS DEL ARÁNDANO FRESCO EN EL MERCADO MAYORISTA," vol. 4453, no. 1992, pp. 195–205, 2003.

[10] "Oportunidades y retos en la exportación de Arándanos."

[11] C. HENKEN, A .M., VAN DER HEL, W., HOOGERBRUGGE, A. AND SCHEELE, "Desarrollo de productos de arándanos con propiedades antioxidantes y probióticas," *Ocean Modelling*, vol. 22, no. 3, pp. 261–287, 2005.

[12] B. Defilippi, P. Robledo, and C. Becerra, "Cosecha y Postcosecha," pp. 89–97, 2017.

[13] "Arandanos-Produccion-Mercado," pp. 1–20, 2004.

[14] I. Challenger-Pérez, Y. Díaz-Ricardo, and R. A. Becerra-García, "El lenguaje de programación python," *Ciencias Holguín*, vol. 20, no. 2, pp. 1–13, 2014.

[15] R. Finger, S. M. Swinton, N. El Benni, and A. Walter, "Precision farming at the nexus of agricultural production and the environment," *Annual Review of Resource Economics*, vol. 11, pp. 313–335, 2019.

[16] P. P. Shinde and S. Shah, "A Review of Machine Learning and Deep Learning Applications," *Proceedings - 2018 4th International Conference on Computing, Communication Control and Automation, ICCUBEA 2018*, pp. 1–6, 2018.

[17] E. Anderson, "Inteligencia artificial y sistemas expertos," *Ius et Praxis*, no. 26, pp. 94–99, 1996.

[18] J. P. D. Forsyth, *Computer Vision - A Modern Approach*, 2012.

[19] P. Ongsulee, "Artificial intelligence, machine learning and deep learning," *International Conference on ICT and Knowledge Engineering*, pp. 1–6, 2018.

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

[21] I. G. Courville, Y. Bengio, and Aaron, "Deep Learning adaptive computation and machine learning," *Nature*, vol. 29, no. 7553, pp. 1–73, 2016.

[22] K. Makantasis, E. Protopapadakis, A. Doulamis, N. Doulamis, and C. Loupos, "Deep Convolutional Neural Networks for efficient vision based tunnel inspection," *Proceedings - 2015 IEEE 11th International Conference on Intelligent Computer Communication and Processing, ICCP 2015*, no. January 2018, pp. 335–342, 2015.

[23] A. Felipe, G. Rivera, F. V. Clavijo, M. Andrés, and F.énez López@, "Agricultura De Precisión Y Sensores Multiespectrales Aerotransportados," no. October 2016, p. 7, 2016. [Online]. Available: http://cici.unillanos.edu.co/media2016/memorias/CICI{_}2016{_}paper{_}106.pdf

[24] A. Gómez, F. Clavijo, and A. Jimenez, "Agricultura de precisión y sensores multiespectrales aerotransportados," 10 2016.

[25] L. Pantaleone and M. Tosini, "Clasificación de cultivos a partir de imágenes satelitales landsat em."

[26] V. Olaya, "Procesado de imágenes." [Online]. Available: http://volaya.github.io/libro-sig/chapters/Imagenes.html

[27] N. Ketkar, "Introduction to keras," in *Deep learning with Python*. Springer, 2017, pp. 97–111.

[28] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "{TensorFlow}: A system for {Large-Scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[29] W. McKinney *et al.*, "pandas: a foundational python library for data analysis and statistics," *Python for high performance and scientific computing*, vol. 14, no. 9, pp. 1–9, 2011.

[30] E. Bisong, "Introduction to scikit-learn," in *Building machine learning and deep learning models on Google cloud platform*. Springer, 2019, pp. 215–229.

[31] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, pp. 381–386, 2020.

[32] J. Redmon and A. Farhadi, "Yolo V2.0," *Cvpr2017*, no. April, pp. 187–213, 2017. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/9789812771728{_}0012

[33] ——, "Yolo9000: Better, faster, stronger." [Online]. Available: http://pjreddie.com/yolo9000/

[34] ——, "Yolov3: An incremental improvement." [Online]. Available: https://pjreddie.com/yolo/.

[35] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 4 2020. [Online]. Available: http://arxiv.org/abs/2004.10934

[36] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, TaoXie, J. Fang, imyhxy, K. Michael, Lorna, A. V, D. Montes, J. Nadar, Laughing, tkianai, yxNONG, P. Skalski, Z. Wang, A. Hogan, C. Fati, L. Mammana, AlexWang1900, D. Patel, D. Yiwei, F. You, J. Hajek, L. Diaconu, and M. T. Minh, "ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference," Feb. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6222936

[37] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "You only learn one representation: Unified network for multiple tasks," *arXiv preprint arXiv:2105.04206*, 2021.

[38] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 6 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[39] W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow," https://github.com/matterport/Mask_RCNN, 2017.

[40] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[41] V. Badrinarayanan, A. Kendall, R. Cipolla, and S. Member, "SegNet : A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," vol. 39, no. 12, pp. 2481–2495, 2017.

[42] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD : Deconvolutional Single Shot Detector," no. January, 2017. [Online]. Available: http://arxiv.org/abs/1701.06659

[43] L. Ale, N. Zhang, and L. Li, "Road Damage Detection Using RetinaNet," *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, pp. 5197–5200, 2019.

[44] A. Rastogi and P. K. Yalavarthy, "SpiNet: A deep neural network for Schatten p-norm regularized medical image reconstruction," *Medical Physics*, vol. 48, no. 5, pp. 2214–2229, 2021.

[45] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," 2020.

[46] Basler, "General Specifications," vol. 2056, pp. 1–15, 2019.

[47] B. Ag, "Basler ace acA2040-55u," vol. 000, no. October, 2015. [Online]. Available: https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca2040-55um/

[48] C. Cabrera, D. Cervantes, F. Muñoz, G. Hirata, P. Juárez, and D.-L. Flores, "Mask r-cnn to classify chemical compounds in nanostructured materials," in *Latin American Conference on Biomedical Engineering*. Springer, 2019, pp. 401–411.