UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

Localización y conteo de árboles en áreas de interés forestal mediante imágenes aéreas multiespectrales georreferenciadas

POR

Pablo Ignacio Meza Sparza

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título profesional de Ingeniero Civil Industrial

Profesor Guía

Guillermo Cabrera Vives, Ph.D.

Octubre, 2022

Concepción, Chile

Agradecimientos

Quisiera agradecer enormemente a la Unidad de Data Science de la Universidad de Concepción por abrirme las puertas y permitirme aprender de esta área tan interesante, principalmente a mi profesor guía, a Manuel y Alexis por brindarme un ambiente más que agradable.

Agradecimientos especiales al profesor Rodrigo De La Fuente quien nos guió inicialmente en este corto camino recorrido.

Por último y no menos importante, agradecimientos a mi familia y amigos que me apoyaron en este proceso, haciéndolo único y especial. Mención importante a mi hermana Camila y Renata que fueron pilares fundamentales en todos estos años.

Resumen

En la actualidad, una de las industrias que más aporta al Producto Interno Bruto de Chile es la industria forestal, alcanzando un 2 % de su totalidad el año 2021. Teniendo en consideración los esfuerzos que realiza esta industria para mejorar los inventarios de árboles en los distintos bosques. Es que este trabajo presenta y aplica una metodología basada en algoritmos de aprendizaje profundo supervisados para la tarea de conteo y detección de árboles en predios forestales de la Región del Biobío. Con esto, se pretende automatizar el proceso de conteo y detección, mejorar la gestión de los recursos y proveer información para la estimación de proyecciones, que en conjunto guíen la acertada toma de decisiones. Para lograr esto, se hace uso de imágenes multiespectrales georreferenciadas, que son procesadas y pasadas por detectores de objetos de una etapa pertenecientes al área de la visión computacional llamados YOLOv5 y Scaled YOLOv4. En paralelo, se aplican técnicas de escalamientos tales como el aumento en la resolución de la imagen de entrada, o aumento en la profundidad y ancho de la red neuronal convolucional para lograr un buen desempeño en las tareas descritas.

La metodología para automatizar la detección y el conteo de árboles, se compara con un estudio realizado en la misma zona de Chile. Al contrastar los resultados, se puede concluir que se logran mejoras en el conteo y detección con el enfoque propuesto cuando se comparan los detectores utilizados. Además, se obtienen métricas levemente inferiores cuando se compara con el segmentador usado por el estudio guía, pero que son considerablemente más rápido al momento de entrenar e inferir.

Abstract

Nowadays, one of the industries that contributes the most to the Gross Domestic Product in Chile is the forestry industry, which achieved 2 % of GDP in 2021. Considering the efforts that this industry makes to better achieve inventories of trees in the different forests and make good projections for a correct management of resources and appropriate decision making, a methodology based on supervised deep learning algorithms is presented and applied for the task of counting and detecting trees in forest estates in the Biobío Region. This effort aims to automate the counting and detection process, trying to improve resource management, provide information to make projections and allow good decisions to be made. Therefore, using georeferenced multispectral images, two one-stage detectors belonging to the area of computational vision, called YOLOv5 and Scaled YOLOv4, are used, applying scaling techniques such as increasing the resolution of the input image, or increasing the depth and width of the convolutional neural network.

The methodology for automating the detection and the counting of trees, is compared with a research applied in the same zone of Chile. When the both methodologys are compared, it can be seen that the result achieved with the proposed approach are better than the previus research in the object detectors. In addition, slightly lower metrics are obtained when compared to the segmenter used in the other study, but these are considerably faster than the others.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

En el presente capítulo se presenta la descripción del problema que motiva este estudio. Para acotar la problemática, se establece el objetivo general como los objetivos específicos, que en conjunto muestran el alcance de la pesquisa. También se presenta la justificación del tema y se mencionan y explican de forma general el contenido de cada capítulo del informe.

1.1. Descripción del problema

Actualmente, la industria forestal en Chile ocupa uno de los puestos más importante a nivel nacional en cuanto a producción se trata. Al analizar el "Producto Interno Bruto" (PIB) que ha aportado esta industria, se observa que en 2017 el rubro forestal aportó un 1,9 % del PIB total (Cardemil, 2021) y que en 2021 dicha cifra aumentó a un 2,0 % según indica Instituto Forestal.

Esta industria también ha marcado otros hitos importantes, tales como producir el 2,24 % del total de exportaciones de productos madereros tranzados a nivel mundial el 2019 (Cardemil, 2021). Además genera un 7 % del total de exportaciones globales de pulpa para papel, ocupando el cuarto lugar a nivel mundial en exportaciones de esta materia prima. Así, la Región del Biobío juega un papel importante en el cumplimiento de los hitos que se mencionan. Debido a que es la zona que cuenta con mayor área de plantación forestal en la actualidad y también en años anteriores (Instituto Forestal).

Para seguir aprovechando este recurso y sus beneficios económicos, es necesario una correcta

administración y gestión de los escasos recursos forestales. Por ello, hoy en Chile es necesario contar con herramientas modernas de análisis para hacer un mejor uso y realizar una mejor gestión de los recursos naturales. La correcta gestión, se puede lograr controlando y realizando un seguimiento al inicio de los procesos forestales, es decir, analizando las plantaciones de las distintas especies de árboles utilizadas en la industria. Generando de esta forma, datos que resuman el estado actual del predio forestal y, a la vez, permitan realizar estimaciones del volumen de madera que se dispondrá en un futuro para los distintos procesos productivos. Posibilitando la toma de decisiones respecto a la oferta de materia prima, evitando tener un déficit de insumos, o bien, un superávit innecesario que traiga problemas en temas logísticos y de inventario.

Mantener un control y seguimiento de las extensas áreas cultivadas necesarias para el nivel de producción actual, presenta varias dificultades. De ellas se destacan en este estudio: recursos de tiempo, esfuerzo y dinero destinados en la extracción de los datos referidos a la localización y conteo de las distintas especies de árboles. Esto se debe a que, no existen tecnologías en el análisis de los datos, y con ello, la extracción de estas mediciones deseadas es realizada de dos formas manuales. La primera es recorriendo las zonas con cuadrillas de personas y anotando de forma sistemática la ubicación y la cantidad de árboles observados. Mientras que la segunda, realiza la localización y conteo de forma manual con imágenes aéreas capturadas por dispositivos como drones. Sin embargo, ambos métodos son costosos en esfuerzo, tiempo y dinero.

Dentro de la literatura se puede apreciar que este problema se ha abordado utilizando distintos algoritmos y tecnologías para trabajar imágenes aéreas de distintas perspectivas. Con el fin de detectar especies de árboles con mayor rapidez y menor esfuerzo como lo presenta Chen y Shang (2022). Por otro lado, también se observa en la literatura esfuerzos en realizar la tarea de localizar y contar a través de algoritmos de aprendizaje profundo como se expone en el trabajo realizado por Yarak et al. (2021), Ammar et al. (2021), Pulido et al. (2020), entre otros.

En base a la problemática anteriormente mencionada, la Unidad de Data Science (UDS) de la Universidad de Concepción, que es un centro de extensión que busca la resolución de

problemas asociados a la manipulación de datos para mejoras en los procesos productivos a través de soluciones digitales. UDS ha tomado la decisión de involucrarse en esta problemática desarrollando Deep Hub, que es un sistema que busca resolver más de una necesidad en el área forestal. Sin embargo, en esta investigación sólo se aborda el problema de localización y conteo de las distintas especies forestales. Por ello, en el presente estudio, se evalúa distintos algoritmos de aprendizaje profundo para localizar y contar de forma autónoma árboles en predios forestales a través de imágenes aéreas georreferenciadas. Con el fin de proponer o descartar algoritmos para una futura implementación de Deep Hub en empresas o proyectos del área forestal. De manera sencilla y rápida se realiza la tarea de localización y conteo con las imágenes entregadas.

1.2. Objetivos

Objetivo General

Implementar, evaluar y comparar distintos algoritmos de aprendizaje profundo para localización y conteo de árboles en plantaciones forestales a través de imágenes aéreas georreferenciadas.

Objetivos Específicos

I Preprocesar las imágenes georreferenciadas multiespectrales para el posterior análisis.

II Seleccionar, implementar y adaptar algoritmos seleccionados para el análisis de predios forestales.

III Entrenar y ajustar los algoritmos seleccionados sobre el conjunto de datos proporcionado.

IV Validar y comparar la calidad de las predicciones obtenidas por cada uno de los algoritmos implementados a través de métricas utilizadas en el área de visión computacional para la tarea de detección y conteo.

1.3. Justificación del tema

Debido a que la industria forestal tiene un gran nivel de producción para consumo nacional e internacional, es necesario tener herramientas de monitoreo automático de los recursos forestales que permita la generación de reportes del estado actual de los predios seleccionados, en conjunto de métricas de interés, consiguiendo un proceso eficiente y menos costoso.

En este trabajo se busca implementar algoritmos de aprendizaje profundo eficientes y rápidos en comparación a otros métodos que utiliza en la industria para el conteo manual y aplicaciones de modelos matemáticos que se adaptan a las características del predio forestal en estudio como Falk y Campos (2014), Huo y Lindberg (2020) y Pulido et al. (2020). Con esta implementación se presenta una metodología para la detección y conteo de árboles en predios forestales de la Región del Biobío. Con el propósito de aportar una nueva visión que pueda ser replicada para abaratar costos en dicho proceso. Además, se espera proveer información a entidades interesadas, donde puedan observar las métricas obtenidas en el presente estudio y les facilite tomar buenas decisiones para nuevas implementaciones o nuevos estudios en el área.

1.4. Contenidos

El presente trabajo cuenta con los capítulos llamados “Revisión Bibliográfica”, “Marco teórico“, “Metodología”, “Resultados experimentales y discusión” y “Conclusión”. En el primero se habla de las metodologías, tipo de datos y algoritmos que se utilizan para abordar la detección y conteo de árboles presentes en la literatura y estado del arte. En el segundo, se aborda la teoría que hay detrás de las redes neuronales y redes neuronales convolucionales, es decir, la teoría de las estructuras que son la base en el tipo de detectores de árboles utilizados en este estudio. En el capítulo de “Marco teórico” también se menciona y describen los detectores que se utilizan en el desarrollo del presente estudio. En el tercer capítulo, se aborda la metodología seleccionada para enfrentar este trabajo. En otras palabras, se describe el set de datos que se utiliza, cómo se trabajan y procesan estos datos para poder utilizarlos, cómo se implementan los detectores seleccionados y cómo se evalúa el desempeño de éstos. En el capítulo de “Resultados experimentales y discusión” se exhiben los valores que se obtienen en

las métricas de evaluación que se presentan en la metodología para la tarea de conteo y detección sumado a los tiempo de entrenamientos de los algoritmos que se implementan. También se explica y exponen algunos puntos importantes que se asumen para la realización de esta investigación, adicionado de observaciones que afectan a los resultados y pudiesen ser mejorados. Finalmente, en el último capítulo se muestran las conclusiones que se obtienen del estudio, además de las propuestas para próximos trabajos o el mismo desarrollo de éste.

Capítulo 2

Revisión bibliográfica

Dentro de la literatura se puede observar que generalmente el problema de detección y conteo de árboles, ya sea para fines forestales, ambientales u otros, se aborda generalmente usando como datos imágenes RGB, multiespectrales o modelos provenientes de imágenes LiDAR o combinaciones de estos (Sección 2.1). Estos datos son usados a través de dos métodos, (1) métodos clásicos (Sección 2.2) y (2) métodos de aprendizaje profundo para la automatización de este proceso (Sección 2.3).

En esta sección, se presentan los datos generalmente utilizados como tipos de métodos implementados para la resolución del problema de detección y conteo de árboles ya sea para fines forestales, agrícolas u otros.

2.1. Datos utilizados

Generalmente, los datos utilizados en los distintos métodos provienen de la captura de tipos de sensores diferentes que son instalados en vehículos aéreos como aviones, helicópteros o drones que sobrevuelan la zona de estudio. Igualmente, se pueden implementar estos sensores en satélites que obtienen imágenes de distinto tipo, una vez que orbite el área de interés.

El sensor del vehículo que captura la información del área, determina el tipo de dato que se utilice en el estudio. Esto debido a que cada sensor es sensible a un tipo de información distinta y dicha información es guardada y almacenada para su posterior análisis.

Los datos que generalmente se utilizan para el análisis de áreas forestales, son imágenes RGB, imágenes multiespectrales e imágenes LiDAR. Independientemente del tipo de imagen que se utilice, éstas tienen asociados la ubicación espacial de cada pixel capturado en la imagen en un sistema coordenado específico, que permite posteriormente hacer un seguimiento en terreno, o monitoreo a través de distintos medios. Esto gracias al sistema de georreferenciación que posibilitan las tecnologías existentes actualmente.

2.1.1. Imágenes RGB

Las imágenes RGB son generadas por sensores que tienen resolución en tres bandas, es decir, capturan la luz reflejada por el área en estudio en tres longitudes de onda distinta. Una perteneciente al color rojo, otra banda guarda información capturada por el sensor sensible a la luz verde y por último, una tercera banda que almacena información del sensor sensible a la luz azul. Finalmente, estas tres bandas se superponen, creando en su conjunto información para que un mismo pixel tenga tres intensidades de luz diferente en el color verde, azul y rojo, creando distintas tonalidades y colores que el ser humano puede percibir en el diario vivir. Así, cada pixel está asociado a un sistema de coordenadas que corresponde a un punto en el mapa geográfico del país. Este sistema de información geográfica (GIS por sus siglas en inglés) puede visualizarse en un software como es visto posteriormente.

2.1.2. Imágenes multiespectrales

Los datos multiespectrales son similares a las imágenes RGB, sin embargo, su única y gran diferencia es que posee resolución en cinco bandas. Es decir, posee las mismas tres bandas que tienen las imágenes RGB y además, captura información de las longitudes de onda del infrarrojo cercano que es reflejada por el área de estudio. Al igual que las imágenes RGB, estas imágenes multiespectrales asocia cada pixel de la imagen a un GIS para saber a qué parte del mapa corresponde.

2.1.3. Imágenes LiDAR

Por otra parte, los datos LiDAR se obtienen utilizando un vehículo aéreo que se desplaza sobre la zona de estudio, emitiendo rayos láser sobre éste. Las emisiones de energías rebotan en la superficie y al devolverse son captadas por el mismo vehículo aéreo que posee un dispositivo sensible a el rayo emitido (sensor LiDAR). Realizando una medición del tiempo que el láser demora en retornar desde que fue lanzado, se estima la altura de la superficie en la que el haz colisiona. A esta información se adjunta gracias al Sistema de Posicionamiento Global (GPS por sus siglas en inglés) las coordenadas en el espacio. Es decir, se arma un mapa tridimensional con coordenadas (X,Y,Z) de la superficie que se está estudiando. Así se espera que en la práctica, la señal emitida hacia un punto que corresponda a la copa de árbol, tarde menos tiempo en volver al sensor LiDAR en comparación a un punto que se encuentre en el suelo del área de estudio.

2.1.4. DTM, DSM y NDVI

Otro tipo de dato que utiliza Pulido et al. (2020) es el Modelo Digital de Superficie (DSM por sus siglas en inglés) y el Modelos Digital de Terreno (DTM por sus siglas en inglés). DSM es una ortofotografía de cuatro bandas, es decir, la representación fotográfica del terreno estudiado manteniendo la escala y disposición espacial de éste. Esto permite que DSM tenga la misma validez que un plano ortográfico. DTM es un ortomosaico, es decir, es un conjunto de imágenes organizadas y ordenadas que en su composición presentan una sola imagen del terreno en estudio como se muestra en la Figura 2.1. Ambas imágenes son construidas a partir de técnicas de "Estructura del Movimiento" usadas por Özyeşil et al. (2017). Estas técnicas son para ordenar o estimar un espacio tridimensional en base al movimiento del sensor al observar un plano bidimensional. Esto último, se puede observar de manera cercana cuando seres vivos que poseen ojos pueden apreciar ciertas características del espacio al moverse. Como la observación de líneas y puntos que se desplazan más lento o rápido, y percibiendo un espacio de tres dimensiones, a pesar de que la imagen real se proyecta en la retina como un plano de dos dimensiones.

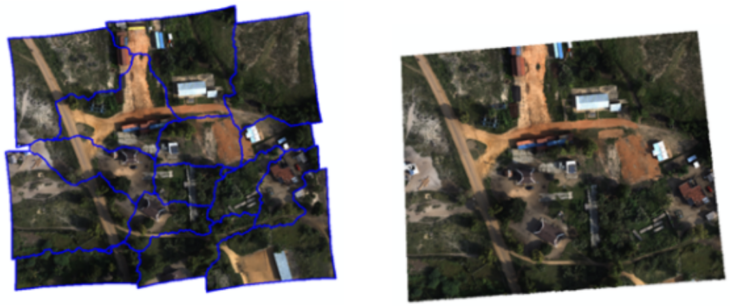


Figura 2.1: Ejemplo de ortomosaico. Fuente: ArcGIS

Los pasos principales utilizados por Özyeşil et al. (2017) son (1) extracción de características en imágenes y comparación de estas, (2) estimación del movimiento de la cámara y (3) recuperación de la estructura tridimensional utilizando el movimiento y las características estimadas minimizando el error de reproyección.

Junto a DTM y DSM, se agrega el índice de vegetación de diferencia normalizada que indica la cantidad y estado del desarrollo de la vegetación en el área de estudio, midiendo algunas longitudes de ondas del espectro electromagnético reflejadas y se calcula mediante la Ecuación (2.1).

$$NDVI = \frac{IRCercano - Rojo}{IRCercano + Rojo} \tag{2.1}$$

El conjunto de estos datos pueden ser procesados por algoritmos de aprendizaje profundo para la detección de árboles y el conteo de ellos.

2.2. Métodos clásicos para la detección y conteo de árboles

Dentro de los métodos utilizados para la detección y conteo de árboles de manera automatizada con los datos que se mencionan. Se encuentran los métodos clásicos, que se

caracterizan por encontrar características ya establecidas o conocidas que generan los árboles en los distintos tipos de datos. Dentro de los métodos clásicos más conocidos se encuentran Local maxima filtering (Wulder et al., 2000), Template matching (Pollock, 1996), Valley-following (Gougeon, 1995), Hough y HOG + SVM (Dalal y Triggs, 2005).

2.2.1. Local maxima filtering

Es un método que detecta los árboles dentro de una imagen, buscando puntos de mayor brillo. Por ello, es que este brillo se debe probablemente a la existencia de un árbol cuya copa está más cerca del sensor que el suelo y, por consecuencia, refleja mayor cantidad de luz por la altura y al estar menos expuesto a sombras. Para esto, el método utiliza una ventana deslizante que recorre toda la imagen. Por cada área que esta ventana evalúa, busca el punto de mayor brillo el cual designa como el centroide del árbol (Ke y Quackenbush, 2011a; Pouliot et al., 2002).

Uno de los principales problemas de este método está relacionado al tamaño de la ventana deslizante, debido a que una ventana muy pequeña, puede obtener un error por exceso, puesto que identifica el mismo árbol como dos o más árboles distintos. Caso contrario, ocurre cuando el tamaño es muy grande y deja de identificar y localizar árboles. Esto último es denominado como un error por defecto u omisión.

Otro problema que se puede identificar en este tipo de método es la existencias de puntos brillantes que no sean exactamente la copa del árbol. O bien, que la ventana deslizante busque un máximo local, es decir, un punto de mayor brillo en un área donde no existan árboles, lo cual estaría perjudicando el desempeño del algoritmo. Además, puede darse el caso de la existencia de cuerpos externos que proyecten sombra hacia los árboles perjudicando la detección de árboles.

2.2.2. Template matching

Este método intenta poner en plantillas las características de los árboles que se intentan detectar, ya sea textura, tamaño, formas, entre otros (Hung et al., 2012; Huo y Lindberg, 2020). Una vez construidas las plantillas, la imagen completa se subdivide en varias subimágenes, donde se evalúa cada pixel para calcular la similitud con las características de cada pixel de la plantilla

y decidir si en dicha área existe la presencia de un árbol. En la Figura 2.2 se ejemplifica este proceso donde se observa la imagen completa ("Plot image I") es subdivida en una subimagen ("Subimage $S_{i,j}$") la cual se compara con las plantillas generadas ("Template T_n").

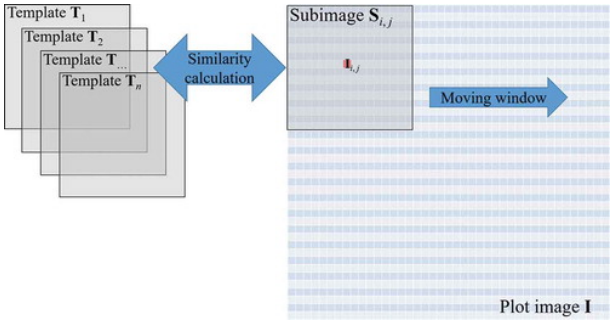


Figura 2.2: Comparación de una subimagen con las distintas plantillas mostrada en Huo y Lindberg (2020).

En la Ecuación (2.2) y (2.3) se observa la manera de evaluar la similitud de cada pixel en cada subimagen con las plantillas existentes.

$$D_m(i,j) = \sum_{a=1}^{A}\sum_{b=1}^{B}[S_{i,j}(a,b) - T_m(a,b)]^2 \tag{2.2}$$

$$P(i,j) = 1 - \text{mín}\left\{\ D_m(i,j)\right\}, m = 1,2,3,...,n \tag{2.3}$$

Donde $I_{i,j}$ un pixel dado en la imagen I, $S_{i,j}$ una subimagen centrada en el pixel $I_{i,j}$ como se ve en la Figura (2.2). T_1, T_2, T_3, ..., T_m son las plantillas generadas. $D_m(i,j)$ es la diferencia entre la subimagen S y la plantilla T_m $\text{mín}\left\{\ D_m(i,j)\right\}$ es la mínima diferencia que hay para todas las plantillas. Finalmente, $P(i,j)$ es la similitud del pixel (i,j) con la posición de un árbol. Con este nivel de similitud se genera un mapa de similaridad como se presenta en la Figura (2.3) donde las zonas rojas son las zonas más parecidas y las azules las más diferentes a las plantillas.

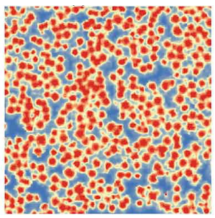


Figura 2.3: Mapa de similaridad creado al comparar las plantillas.

2.2.3. Valley-following

Este enfoque evalúa pixel por pixel la imagen de entrada y distingue los pixeles con mayor luminosidad como un pixel perteneciente a un árbol. Encontrando los límites de dicho árbol por la sombra que éste y el resto de árboles proyecta en el suelo, percibiendo esta sombra como límite de los árboles detectados (Ke y Quackenbush, 2011b; Leckie et al., 2003). En el fondo, se asume que las copas de los árboles reflejan más luz en comparación al suelo, el cual es más oscuro y sombrío.

Uno de los problemas de este método, es el asumir que hay una diferencia notoria entre la luminosidad que refleja el suelo y la copa de los árboles. Este supuesto, queda sin efecto al momento de ver árboles de edad temprana que reflejan poca luz, o bien, datos que provienen de fotografías realizadas en el momento que el sol irradia luz en dirección paralela a la dirección en la que crece el árbol.

2.2.4. Hough

Este método propuesto por Koc-San et al. (2018), muestra una forma de aplicar detección de árboles usando la imagen DVEM, multiespectrales y RGB aplicándole detección de bordes Canny usando Python con OpenCV para identificar las líneas o bordes que conforman la imagen completa. Éste se muestra en la Figura (2.4), donde los dos retratos superiores evidencian el área de estudio y a la derecha, los bordes que conforman a cada una de las formas presentes en la imagen. Una vez detectados los bordes en la imagen, se hace uso de la técnica “Transformación Hough” para detección de círculos. Este método traza circunferencias, usando de centro cada pixel de las líneas detectadas como borde, se puede observar en la franja inferior de la misma

figura. Sobre cada intersección de estas circunferencias trazadas, se realizan "votaciones" que son el conteo de cuantas intersecciones se acumulan en dicho punto en específico. Como se nota, al mostrar apenas cuatro de todas las circunferencias hechas, se evidencia que el punto del centro es que recibe más intersecciones o "votaciones", estableciéndose como el centroide de un árbol detectado.

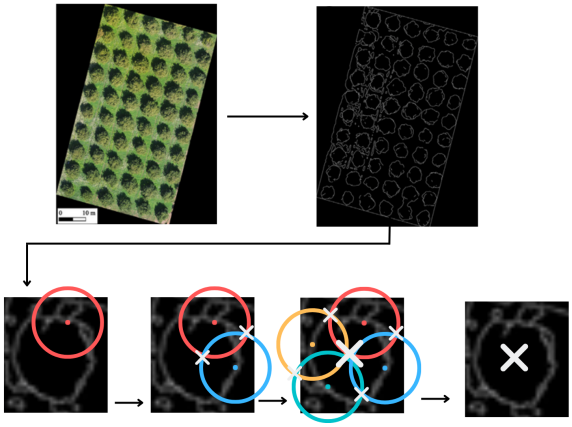


Figura 2.4: Proceso Hough descrito.

2.2.5. HOG + SVM

Este método es propuesto por Wang et al. (2019b) y utiliza un método jerarquizado. La primera jerarquía realiza una clasificación, en categorías "vegetación" y "no vegetación" dentro de la imagen a nivel de pixel, utilizando Support vector machine (SVM por sus siglas en inglés) en la imagen RGB capturada por el dron. En la segunda jerarquía, se aplica el método "Histogram of Oriented Gradients" (HOG por sus siglas en inglés). Este método busca resumir la información de la imagen intentando capturar la información más relevante de la fotografía subdividiendo la imagen en áreas de $n \times n$ pixeles. En cada una de éstas se obtiene el gradiente de cada pixel en la dirección horizontal y vertical como lo señala la Ecuación (2.4) y (2.5) . Teniendo el gradiente del eje x e y se puede obtener la magnitud y orientación (ángulo) como lo reflejan las Ecuaciones (2.6) y (2.7).

$$G_x(x, y) = H(x + 1, y) - H(x - 1, y) \tag{2.4}$$

$$G_y(x,y) = H(x,y+1) - H(x,y-1) \tag{2.5}$$

Donde $H(x,y)$ es el valor del pixel y $G_x(x,y)$ y $G_y(x,y)$ denota el gradiente horizontal y vertical, respectivamente. Por otra parte, $G(x,y)$ junto con su respectivo ángulo $\alpha(x,y)$. Los vectores de magnitud G y orientación α, que se calcula dentro de cada área, son clasificados dentro de un histograma de nueve bins que contienen las orientaciones de 0 a 180 grados.

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \tag{2.6}$$

$$\alpha(x,y) = \tan^{-1}(\frac{G_y(x,y)}{G_x(x,y)}) \tag{2.7}$$

Con el algoritmo HOG se realiza la extracción de características dentro de las áreas categorizadas como "vegetación" para detectar las palmeras existentes con SVM.

2.3. Métodos de aprendizaje profundo para la detección y conteo de árboles

Actualmente, hay auge en el uso de herramientas como el aprendizaje profundo en máquinas que permite aprender a distinguir las características de los árboles de forma autónoma. De modo que modela abstracciones de alto nivel por medio de una red neuronal que imita las conexiones cerebrales humanas.

El área, dentro del aprendizaje profundo que se enfoca en el análisis de imágenes, es la visión computacional, y lo hace a través de redes neuronales convolucionales (Le Cun et al., 1989) que se explican en la Sección 3.1. El uso de estas estructuras permiten que la visión computacional sea capaz de procesar e inferir la información contenida dentro de imágenes, para entregar como salida datos de interés a quien lo solicite.

Cabe mencionar que las redes neuronales convolucionales son encargadas del análisis de

la imagen, tal como se explica en el Capítulo 3. Estas redes neuronales mantienen el orden o disposición espacial de los datos de entrada, debido a que la disposición de los pixeles en la imagen tienen una importancia relevante para poder distinguir características dentro de ella. Por otra parte, las redes convolucionales se caracterizan por realizar operaciones llamadas convoluciones a la imagen de entrada que recibe la red, las cuales se encargan de extraer distintas características. Posibilitando que en las últimas capas de la red, las características obtenidas sean cada vez más abstractas y sea posible obtener las propiedades más importantes de la imagen de entrada, logrando detectar los objetos y propiedades de interés.

En base a lo explicado anteriormente, se pretende que el algoritmo de aprendizaje profundo en el problema de detección y conteo de árboles de predios forestales, sea capaz visualizar la imagen a través de la arquitectura de redes neuronales convolucionales, es decir, pueda aprender a distinguir las características más representativas de los árboles independientemente de la especie. Cuando el algoritmo aprenda a distinguir tales características, es posible predecir la ubicación del objeto de interés, señalando las coordenadas dentro de la imagen donde éste se encuentre.

Actualmente en el estado del arte, los trabajos enfocados en resolver el problema de detección y conteo a través del aprendizaje profundo. Esto evidencia que la mayoría ocupan arquitecturas de redes neuronales convolucionales denominados detectores de dos etapas y detectores de una etapa. Los primeros son los más ocupados, debido a que el primero se ha estado desarrollando y potenciando en los últimos ocho años.

Los detectores de dos etapas abordan el problema de la detección, tomando una imagen de entrada y extrayendo en una arquitectura de redes convolucionales características dentro de ésta para poder proponer regiones de interés, donde probablemente haya un objeto buscado. Todas estas áreas propuestas, que identifica el algoritmo, son pasadas a otra red convolucional que intenta predecir el objeto que se encuentra dentro de ésta. Así, se genera como salida final el tipo de objeto detectado en conjunto de su ubicación dentro de la imagen, es decir, sus coordenadas. La Figura 2.5 muestra un detector de dos etapas y explica de manera general cómo funciona.

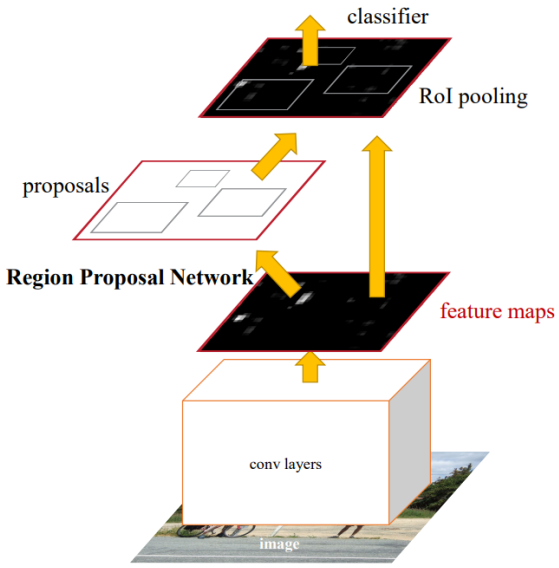


Figura 2.5: Detectores de dos etapas presentado en Ren et al. (2015).

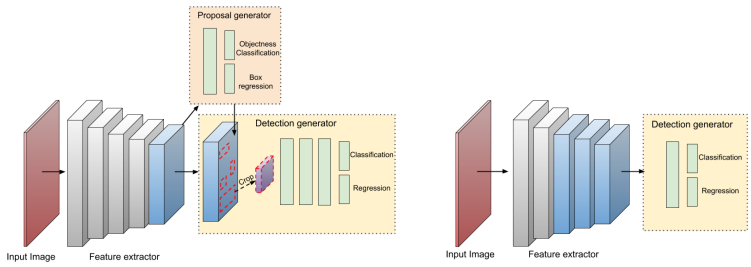
En la Figura 2.5 se puede ver cómo la imagen que se encuentra en la base, es una entrada de la red convolucional ("conv layers"), y realiza operaciones llamadas convoluciones sobre ésta para obtener un mapa de características ("feature maps") que captura y resume la información importante de la imagen de entrada. En este mapa de características, la red es capaz de proponer imágenes donde probablemente se encuentre un objeto de interés ("proposals"). Luego, éstos pasan por otra red neuronal para clasificar el objeto que se encuentra en dicha área de interés entregada. Hecho lo anteriormente, se obtiene como salida los distintos objetos detectados, la clase a la que pertenece y la ubicación dentro de la imagen a la que pertenece.

Las redes convolucionales, a pesar de que tienen buenos resultados en la detección de objetos suelen ser lentos tanto en el entrenamiento como la inferencia. Intentando solucionar este problema de tiempo es que en los últimos años se han desarrollado y perfeccionado los detectores de una etapa, que a diferencia de los detectores de dos etapas. Éste enfoque ingresa la imagen a la red convolucional para realizar una extracción de características para hacer la detección y clasificación de los objetos encontrados. Esto permite una red más rápida y eficiente, además de una visión más global en comparación a detectores de dos etapas debido a que no evalúa a la imagen por separado.

Un contrastre entre los detectores descritos pueden verse gráficamente en la Figura 2.6,

donde en la izquierda se observa al igual que en la descripción de la Figura 2.5, una imagen de entrada (“Input image”) que entra a una red convolucional y realiza una extracción de características depositadas en un mapa de características. Este mapa de características ilustrado por el primer “tensor azul” de izquierda a derecha, es la entrada a una red neuronal que realiza el proceso que genera las áreas propuestas (“Proposal generator”), indicando dichas zonas donde probablemente exista un objeto que se esté buscando. Luego es recortado y pasado por otra red neuronal que clasifica el objeto que está dentro del él (“Detection generator”).

Mientras, en el lado derecho de la Figura 2.6, se observa la estructura general de un detector de una etapa. Comienza en la imagen de entrada (“Input image”), luego, se hace una extracción de características en una red neuronal convolucional (“Feature extractor”). Al mapa de característica generado, se le realiza inmediatamente la detección y clasificación de objetos dentro del objeto encontrado (“Detection generator”).



(a) Two-stage Faster R-CNN (b) One-stage RetinaNet

Figura 2.6: Comparación de estructura detectores de una y dos etapas usada en Carranza-García et al. (2021).

En la literatura, uno de los trabajos que utiliza ambos detectores es el realizado por Pulido et al. (2020), que detecta dos tipos de árboles en distintas zonas de cultivos. Este trabajo propone una metodología con imágenes DTM, DSM y NDVI obteneiendo un ”Modelo Digital de Vegetación Elevada” (DEVM) como se muestra en la Figura 2.7. Una vez se obtienen las imágenes DEVM, se hace uso de los siguientes algoritmos con arquitecturas de dos etapas. El primero es denominado detector de una etapa, DetectNet, que hace uso de un extractor de características llamado GoogleNet. Esta red neuronales convolucional tiene 22 capas que reciben de entrada una imagen de tres bandas (RGB) y de salida la clase del objeto presente y las coordenadas de pixeles de las esquinas del cuadro delimitador de dicho objeto que se detecta

en relación con el centro de la cuadrícula. Como se menciona, este modelo utiliza aumentación de datos en el proceso de entrenamiento. Una técnica donde se incrementa la cantidad de datos existentes usando distintas herramientas que permiten generar datos desconocido a partir de uno conocido, debido a que el tensor de entrada ya no será el mismo (Pulido et al., 2020).

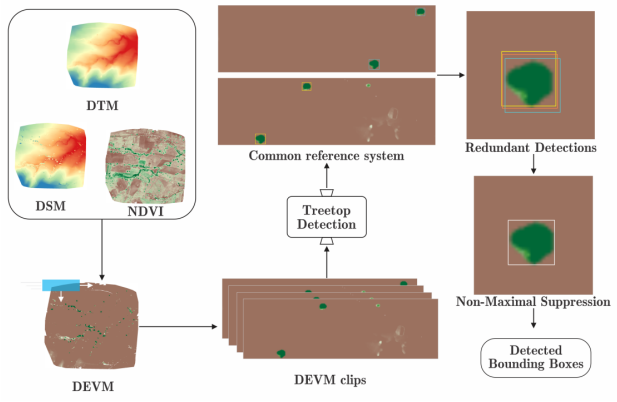


Figura 2.7: Procesamiento de datos realizado por Pulido et al. (2020) para lograr modelo DEVM.

Por otro lado, en este trabajo mencionado se hace uso de un detector de dos etapas llamado Faster R-CNN propuesto por Ren et al. (2015). Este detector es utilizado con dos extractores de características distintas, v2 y ResNet-101, es decir, una red residual que tiene 101 capas convolucionales. Esto al ser residual, pese al gran número de convoluciones no afecta al rendimiento debido a las conexiones "skip connection" que proporcionan funciones identidades evitando el sobre procesamiento de los datos cuando no es necesario.

Los últimos detectores que se utilizan en Pulido et al (2020) son Single Shot Multibox Detector publicado por Liu et al. (2016) con el extractor Inception v2 y R-FCN Dai et al. (2016) con el extractor de características ResNet-101. Estos son entrenados y comparados los resultados en la detección con algunos de los modelos clásicos mencionados en la Sección 2.2.

Los resultados evidencian una notable superioridad de los detectores de aprendizaje profundo como lo es la métrica $AP_{0,5}$, que se explica en detalle en la Sección 4.4. Sin embargo, a grandes rasgos, $AP_{0,5}$ mide en una sola imagen la "calidad" de las detecciones hechas contabilizando las detecciones hechas correctamente, las detecciones incorrectas y las faltantes. En base a eso, se genera una curva llamada "Precision-recall" cuya área será el valor de AP. Ahora bien,

cuando se tiene un conjunto de imágenes, el promedio de AP en ese conjunto de imágenes será denominada mAP del conjunto de datos, es decir, se suma el AP obtenido por cada imagen y se divide por el número de muestras.

En base a estos resultados se buscan detectores de una etapa en la literatura y se encuentra que losmás destacados en la comunidad de visión computacional es YOLO, propuesto por Redmon et al. (2016). Su nombre proviene de la sigla de "You Only Look Once", haciendo énfasis en ser un detector de una sola etapa al "mirar una sola vez" en la imagen. YOLO ha tenido bastantes mejoras en su arquitectura y métodos de realizar el entrenamiento e inferencia. Desde el 2016 ha tenido varias versiones, siendo las más conocidas YOLOv1, YOLOv2, YOLOv3, YOLOv4 y YOLOv5. Arquitecturas "end to end" que a lo largo del tiempo han evolucionado en técnica como el uso de "anchor boxes", bibliotecas utilizadas como Pytorch, técnicas potentes de aumento de datos para el entrenamiento y la validación, entre otros cambios que han hecho de esta arquitectura en una de las más rápidas y eficientes en los últimos tiempos.

Otro tipo de arquitectura interesante es el propuesto por Google Brain, quien ha competido en el área de detectores de una etapa, sacando a la luz el detector llamado EfficientDet (Tan et al., 2019). Esta arquitectura, que al igual que su extractor de características llamado EfficientNet, tiene la capacidad de cambiar las dimensiones de su estructura para lograr las mejores combinaciones de ancho, profundidad y resolución de la imagen de entrada con el fin de encontrar la mezcla de rapidez y precisión deseada.

En Ammar et al. (2021), la detección de palmeras en cultivados en Arabia Saudita en imágenes RGB tomadas por drones usa algoritmos de una etapa mencionados YOLOv3, YOLOv4 y EfficientDet y un algoritmo de dos etapas ya mencionado llamado Faster R-CNN. En este estudio se obtuvieron 13 071 instancias etiquetadas manualmente de las cuales 80 % es destinada al entrenamiento y el resto al testeo. Se obtienen como resultado que EfficientDet obtiene el mejor $mAP_{0,5}$

Otra investigación destacada, es la realizada Jintasuttisak et al. (2022), quienes observan el desempeño logrado principalmente por las últimas versiones de YOLO en la detección de

palmeras en la Península Arábiga, Medio Oriente y África del Norte. Además, se prueba el efecto que tiene la altura de la captura de la fotografía en el desempeño del algoritmo al intentar realizar las detecciones. La metodología utilizada se observa en la Figura 2.8 y en general no se ve una variación disruptiva en comparación a las metodologías anteriores. Se puede observar que de las capturas fotográficas obtenidas por el vehículo aéreo no tripulado, se obtienen distintas imágenes de las cuales, un gran porcentaje son destinadas al entrenamiento y otros a la validación del modelo. Se entrena el modelo con el set de datos destinados para este proceso y finalmente se realizan las detecciones pertinentes para obtener métricas y reportar.

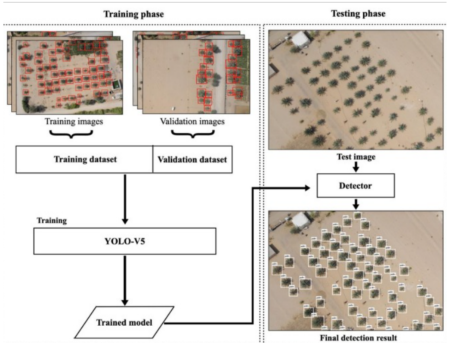


Figura 2.8: Metodología utilizada por Jintasuttisak et al. (2022).

Las métricas que se obtienen por estos detectores en análisis de imágenes aéreas RGB a 122 m muestran que YOLOv5m logra el mejor $mAP_{0,5}$ y se confirma que a menor estructura, la inferencia y entrenamiento se hace más rápida.

Finalmente, en Pérez et al. (2021), aplican el detector YOLOv3 y el segmentador Mask R-CNN para el conteo, detección y segmentación de árboles en plantaciones de bosques en Chile con distinto tiempo de crecimiento y maduración que comprenden entre 1 a 10 años de edad. En este trabajo, a las imágenes aéreas las organizan en una grilla compuesta por varias celdas como se muestra en la Figura 2.9. Se toman 10 celdas etiquetadas de la forma ilustrada en la Figura 2.10 para poder entrenar los dos modelos que se mencionan. Cabe mencionar que las etiquetas son cuadros delimitadores debido a que lo único que interesa es la ubicación y dimensión del árbol.

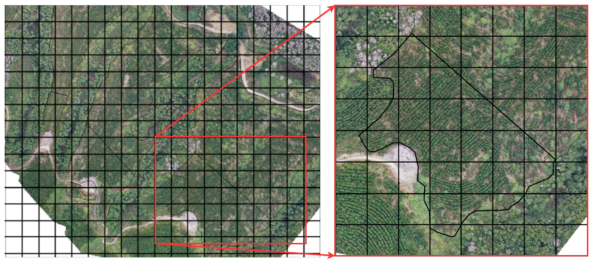


Figura 2.9: Grilla usada por Pérez et al. (2021) en uno de sus predios forestales.

Figura 2.10: Etiquetado usada por Pérez et al. (2021) en uno de sus predios forestales.

Como se explica en detalle en el Capítulo 4, a estas celdas, se le realizan varios recortes aplicando una ventana deslizante que la recorre de izquierda a derecha y de arriba hacia abajo, como se ilustra en la Figura 2.11. Estos recortes que mantienen las respectivas etiquetas, son el set de imágenes que se utilizan para el entrenamiento y testeo de los distintos modelos.

Figura 2.11: Recortes hechos a la celdas hechos por Pérez et al. (2021).

Luego, al igual que en las metodologías anteriores se entrenaron y validaron los modelos. Sin embargo, debido a que existe solape entre las imágenes creadas, se reconstruye la imagen original una vez hecha las detecciones en cada imagen individual utilizando un área de confianza en cada una de ellas. Con ello se deja de asumir que las imágenes son independientes y se evita la sobreestimación de árboles en el recuento general. Debido a que este estudio es previo al actual, la metodología utilizada en el 2021 presentada es pilar para la elaboración de la metodología de esta investigación.

Se obtiene como resultado en la tarea de conteo para los bosques de 1, 5, 8 y 10 años que el mejor modelo es Mask R-CNN logrando como mínima diferencia un 0,0 % respecto al número real de árboles existente en la zona de estudio, mientras que YOLO logra como mínima diferencia un 5,0 %. En la detección Mask R-CNN logra un $AP_{0,5}$ máximo de 91 % y YOLO un 85 %.

Capítulo 3

Marco teórico

En base a la revisión de la literatura, se puede ver que los detectores de una etapa suelen ser muy eficientes, rápidos y capaces de lograr buenos resultados. Además, se evidencia la novedosa arquitectura EfficentDet que logra competentes, mientras que la familia YOLO es la más usada. Por lo anteriormente mencionado, se utiliza esta familia de detectores en el presente trabajo, específicamente YOLOv5 que contiene las últimas mejoras y actualizaciones de la familia. También, se utiliza Scaled YOLOv4 diseñado por Wang et al. (2020a), que usando la arquitectura y mejoras de esta familia, ocupa además técnicas de escalamiento basándose en EfficientDet. Al comparar dichos modelos se observa que Scaled YOLOv4 puede ser más preciso que EfficientDet y que YOLOv5 puede ser más rápido logrando resultados similares.

En este capítulo, se presenta el marco teórico necesario para entender estas dos arquitecturas de redes neuronales convolucionales y lograr hacer inferencia en imágenes multiespectrales de cinco bandas.

3.1. Redes neuronales

La visión computacional es una parte del campo de aprendizaje profundo donde se utilizan algoritmos que usan redes neuronales convolucionales para el análisis de las imágenes. Estas redes son, redes neuronales que operan sobre los tensores que conforman los datos que entran a la red con operaciones llamadas convoluciones, respetando la disposición espacial de cada

dato en el tensor. Esto se realiza debido a que la posición de cada pixel en la imagen, entrega información importante. Es decir, si los datos en la matriz cambian de posición, entonces las formas y la disposición de colores en la imagen que se observan no serían los mismos.

3.1.1. Red neuronal artificial

Antes de las redes convolucionales, nacieron las redes neuronales artificiales (Fitch, 1944; Fukushima, 1980). Una red neuronal artificial es un conjunto de éstas, que por separado simulan el funcionamiento de una neurona cerebral (Krogh, 2008). En su conjunto le da la complejidad para realizar tareas que a otros métodos logran. Una neurona es la unidad básica de la red neuronal y, por si sola, es una función matemática que recibe valores de entrada. Además, realiza una suma ponderada de éstos para dar un valor de salida como se muestra en la Figura 3.1. En esta figura se puede observar como tres características (x_1), (x_2) y (x_3) ingresan a la neuronas y obtiene un valor al realizar una suma ponderada con pesos dados (w_1), (w_2) y (w_3). Por ejemplo, si las distintas características tienen valor 5, 4 y 7 y los pesos dados son 2, 8 y 3, entonces la suma ponderada es la siguiente.

$$Y = 2 \times 5 + 4 \times 8 + 7 \times 3 = 63 \tag{3.1}$$

Este valor Y obtenido es ingresado a una función de activación. Para ilustrar el proceso, se escoge la función sigmoide que se ve en la Ecuación (3.2).

$$\sigma(Y) = \frac{1}{1 + e^{-Y}} \tag{3.2}$$

Este valor final obtenido en la función de activación, para este caso esta en el rango de [0,1], indicando la probabilidad de que un objeto pertenezca a una clase dada.

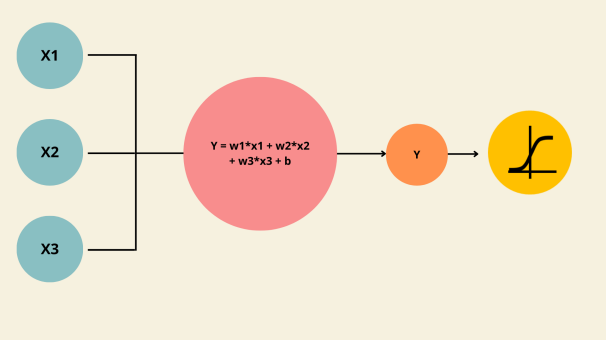


Figura 3.1: Neurona: Unidad básica de una red neuronal.

Ahora, una sola neurona por sí sola no logra tareas complejas como se observa en el caso a de la Figura 3.2. Se evidencia que utilizando este perceptrón y sabiendo de forma a priori la clasificación que se quiere lograr, se pueden ajustar los pesos para realizar un problema de regresión hasta lograr dividir los tipos de datos árbol y no árbol. Sin embargo, en el caso b, se evidencia la limitación del perceptrón por sí solo, ya que no puede dividir los tipos de datos árbol y no árbol de forma lineal. Por esto, se recurre a otra neurona, las cuales, de en conjunto ajustan los pesos de entrada. Así, pueden lograr la división deseada, evidenciando la complejidad que adquiere la interacción de varias neuronas (Yanling et al., 2002). Ahora bien, una red de varias neuronas ordenadas por capas como se observa en la Figura 3.3 pueden resolver tareas compleja, como una red compuesta por una capa de entrada (primeras tres neuronas de la izquierda). Cada una de estas neuronas pertenecientes a esta capa tiene una entrada, donde se realiza una suma ponderada y se obtiene a un valor, que se evalúa en una función de activación. Cada neurona da como salida un valor que es la variable de entrada a cada neurona perteneciente a la siguiente capa llamada capa oculta, que pasan sus valores a la siguiente capa oculta, repitiendo el procedimiento hasta llegar a la capa de salida. La última capa arroja el valor final, que se compara con el valor deseado y se calcula el error. Lo anteriormente descrito pasa en la red de la Figura 3.3.

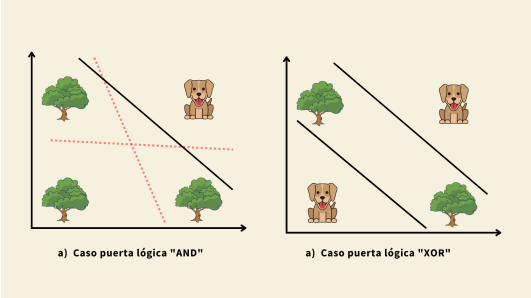


Figura 3.2: Uso de neuronas para dividir tipos de datos.

Para lograr la correcta clasificación de datos, como se ilustra en la Figura 3.2, se deben ponderar los datos de forma certera. Para ello, se entrena inicializando los pesos de forma aleatoria y obteniendo un resultado final. Dicho valor de salida, se compara con el valor deseado a través de una función de perdida que cuantifica la diferencia de la predicción con el valor real. Calculado el error, se analiza la red neuronal para determinar cuál es la responsabilidad que tiene cada neurona en el error a través de la retropropagación. Luego, se obtiene el vector gradiente que sirve para otros algoritmos como el Descenso del gradiente (Robbins y Monro, 1951) que busca minimizar el error de un problema de optimización no lineal, actualizando los pesos de cada neurona para que en su conjunto como red, obtengan valores cada vez más cercanos al esperado Abraham (2004).

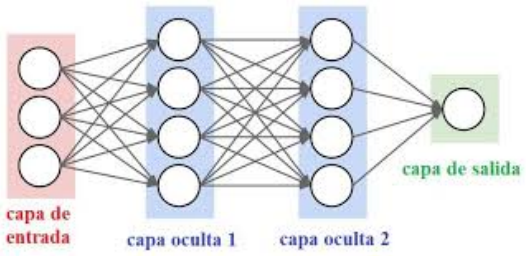


Figura 3.3: Conformación de una red neuronal en base al conjunto de varias neuronas.

Finalmente, la función de activación es un concepto que soluciona el problema de la suma de neuronas, que en el fondo realizan una regresión lineal, en su conjunto, funciona como una única regresión lineal. Para abordar este problema, la suma ponderada de cada neurona, pasa por una función de activación que quita la linealidad a la operación de cada neurona, provocando

que ésta sea no lineal, dando la posibilidad de realizar tareas complejas. Un ejemplo de estas funciones de activación es la función de sigmoide, que sólo puede tener valores comprendidos entre 0 y 1. Sin embargo, existen varias de funciones de activaciones que cumplen con esta propiedad (Wang et al., 2020b).

3.1.2. Red neuronal convolucional

Las redes neuronales convolucionales funcionan de manera similar a las redes neuronales artificiales. Sin embargo, se especializan en recibir imágenes como entrada, las cuales son matrices o tensores cuya disposición espacial de cada pixel, importa para darle un sentido semántico al conjunto de datos en si mismo. De aquí, nace el principal beneficio de las redes neuronales convolucionales, pues reduce el número de parámetros en comparación a las redes neuronales artificiales e impulsa a investigadores del área a diseñar arquitecturas más grandes y complejas (Albawi et al., 2017).

Las redes convolucionales se caracterizan por aplicar operaciones llamadas convoluciones sobre la imagen. Esta operación, como se muestran la Figura 3.4, consiste en aplicar un "kernel" a través de toda la imagen, extrayendo las característica de la imagen y formando un mapa de características respetando la disposición espacial de los datos. Ahora bien, al aplicar distintos "kernels" en una capa convolucional, la red se encarga de que cada uno de estos logre identificar distintas característica en la imagen, ya sea formas, texturas, patrones, contrastes, relieves, etc.

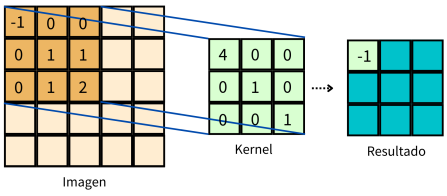


Figura 3.4: Red convolucional aplicando convoluciones a una imagen dada como entrada.

Otra operación importante que se realiza en las redes neuronales convolucionales es la reducción de dimensionalidad de los mapas de características para identificar de mejor manera las características más importantes. Esto se puede lograr a través de las operaciones

Mean-Pooling y Max-Pooling, que funcionan de igual manera que un "kernel" pero obteniendo el promedio o máximo valor de la zona analizada, respectivamente.

3.2. Aprendizaje supervisado

Otra característica importante de los algoritmos que se utilizan en este estudio, es que pertenecen a la categoría de aprendizaje supervisado, es decir, algoritmos que necesitan de humanos para que se les proporcionen la entrada. Se debe señalar las características que deben aprender a través de etiquetas, y a su vez, se indica la salida que el algoritmo debe dar (Saravanan y Sujatha, 2018). En el caso de los trabajos de detección de árboles, se provee de un set de entrenamiento que consiste en imágenes, cuyos árboles son indicados a través de cuadros delimitadores, con el fin que el algoritmo identifique las características de los elementos que se intentan detectar. En el caso que la predicción no sea similar a lo que se pide, se realiza un proceso de retropropagación que actualiza los pesos dentro de la red neuronal para que en la próxima iteración exista en lo posible una mejor precisión.

3.3. Detectores

La detección es la tarea de señalar espacialmente dónde se encuentra los objetos, como se menciona anteriormente, los algoritmos más conocidos se separan en dos categorías. Detectores de una etapa, donde en su arquitecturas de redes neuronales convolucionales entra la imagen, se extraen las características y en el mismo paso por la red convolucional, se predice la detección y la clase a la que pertenece cada objeto encontrado. Mientras que en los detectores de dos etapas, la imagen entra a un tipo de arquitectura de red neuronal convolucional para predecir las regiones dentro de la imagen donde puede existir un objeto. Luego, otra red convolucional predice el tipo de objeto que se encuentra en aquellas áreas propuesta de la imagen (Wang et al., 2020a).

En este trabajo se abordan únicamente detectores de una etapa, es por ello que se ahonda en este tipo de arquitecturas. En general, tienen la siguiente estructura: primero, existe la imagen

o dato de entrada, que siempre ingresa a una estructura llamada “backbone” o extractor de características. Éste es una red convolucional ya diseñada y armada que sirve para extraer, agregar y formar características de la imagen que ha entrado a la red. Además red ya viene preentrenada en distintos set de datos que están ya establecidos como ImageNet o COCO, lo que permite que identifique características importantes de una imagen en general (Jacob Solawetz, 2020). Posteriormente, sigue la estructura denominada “neck” que es una serie de capas que combina las características de esta imagen para realizar detecciones sobre esta. Finalmente, la última estructura es el detector o “head” que toma las características mezcladas por la estructura “neck” y realiza la predicción de la detección y la clase a la que pertenece dicha detección.

3.4. YOLO

Se menciona en este informe a rasgos generales la estructura de un detector de una etapa como lo es la familia YOLO. Sin embargo, ahora se muestra cómo funciona YOLO en general, los algoritmos YOLOv1 a YOLOv3.

En la Figura 3.5 se puede observar el proceso de YOLO, que espera como input una imagen a la cual la divide en una grilla de tamaño S×S (Redmon et al., 2016). Para saber el número de celdas que tiene la grilla, se divide el alto y ancho de la imagen por 32. Por ello la entrada debe tener una dimensión obligada que sea divisible por ese número. Para dar un ejemplo, si se tiene una imagen de 512×512, entonces la grilla sobre la imagen es de 16 celdas de largo por 16 de ancho. Por otro lado, si una imagen es de 224×224 entonces la grilla tal como se ve en la siguiente imagen sería de 7×7 celdas, tal como se muestra en la Figura 3.5.

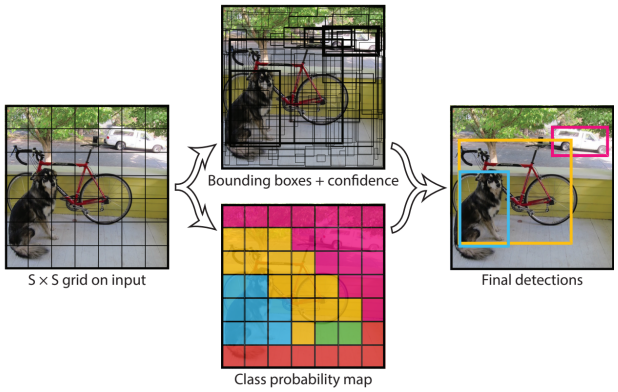


Figura 3.5: Proceso de YOLO para analizar la imagen. Fuente: Redmon et al. (2016).

El proceso de YOLO continua y la imagen es pasada por la arquitectura formando un mapa de características reducidos en comparación a la imagen real, pero mantiene las proporciones. Esto quiere decir, que si la primera celda de la esquina superior izquierda de la imagen original ocupa un 2 % del área, entonces en el mapa de características final, el 2 % del "área" correspondiente a la esquina superior izquierda contiene la información resumida de la celda mencionada anteriormente. En cada celda de esta grilla del mapa de características final, se realiza la predicción de B cuadros delimitadores que consta de cinco coordenadas, las cuales corresponden al centro $(x,\ y)$ de la celda que predice el cuadro delimitador. A estas dos coordenadas se suman el ancho y alto del cuadro delimitador que encierra al objeto en las coordenadas w y h. Finalmente, se adjunta una confianza que es la certeza que tiene el modelo de contener a un objeto dentro del cuadro delimitador. Aparte de estas cinco coordenadas se suman a la predicción de cada celda, C probabilidades condicionales que son la probabilidades de que exista un objeto de tipo C, dado que existe un objeto en el cuadro delimitador.

La Figura 3.5 muestra que cada celda debe predecir un cuadro delimitador, entonces, se observan muchas predicciones, lo que lleva a detectar más de una vez un objeto. Para eliminar este problema se hace uso de la técnica "Non Maximum Suppression" explicado por Hosang et al. (2017), que elimina cuadros delimitadores que pertenezcan a la misma clase y compartan una cantidad considerable de área en común. Este indica que son detecciones del mismo objeto como se puede ver en la Figura 3.6, la cual muestra cómo distintas predicciones que indican un mismo objeto, son finalmente descartadas, hasta dejar la que tiene mayor confianza (Jain y

Nandy, 2019).

Figura 3.6: Proceso Non Maximum Suppression. Fuente: Jain y Nandy (2019).

En la estructura de YOLO, el backbone, neck son diferentes en función de la versión que se está utilizando, al igual que las funciones de activación y técnicas de aumento de datos en el entrenamiento. Por lo tanto, se asume que cada versión, posee mejores estructuras y técnicas que la versión anterior. Sin embargo, en el entrenamiento, cada versión de YOLO busca optimizar la pérdida de la detección y la predicción de la clase. Es decir, busca que la diferencia entre los cuadros delimitadores predichos, en comparación a los cuadros delimitadores de verdad, sean lo menor posible. A esto se suma que se espera que la arquitectura erre lo menos posible a la hora de predecir qué tipo de objeto se encuentra en cada cuadro delimitador (Redmon y Farhadi, 2016, 2018).

La función de pérdida original de YOLOv1 presentada en la Ecuación (3.3), demuestra el objetivo que toda la familia busca optimizar. Esta función de pérdida ha sufrido cambios para obtener mejor desempeño en el entrenamiento e inferencia, sin embargo, la idea sigue siendo la misma.

$$
\begin{aligned}
Loss_{function} = \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \eta_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
+ \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \eta_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\
+ \sum_{i=0}^{s^2} \sum_{j=0}^{B} \eta_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \eta_{ij}^{noobj} (C_i - \hat{C}_i)^2 \\
+ \sum_{i=0}^{s^2} \eta_i^{obj} \sum_{c} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
\tag{3.3}
$$

Donde λ_{coord} es la ponderación a la pérdida de localización. λ_{noobj} es la ponderación específica de la pérdida de confianza para las cajas que no contienen objetos. η_{ij}^{obj} es una variable binaria, que igual a 1 si el objeto existe en la celda i para el objeto j, y 0 en caso contrario. η_{i}^{obj} también es una variable binaria, que igual a 1 si el j-ésimo cuadro delimitador es responsable de la predicción en la celda i y 0 en caso contrario. x_i es la etiqueta de x en la celda i, mientras que $\hat{x_i}$ es su valor predicho. De manera similar para otras variables: y, w y h. $p_i(c)$ es probabilidad básica de que la i-ésima celda de la cuadrícula pertenezca a la clase c, mientras que $\hat{p_i}(c)$ denota la probabilidad predicha por YOLO y C_i es la puntuación de confianza estimada para el cuadro delimitador j proveniente de la celda i, mientras que $\hat{C_i}$ es la confianza en la celda i para el objeto j.

Las primeras tres versiones de YOLO, definen la diferencia de coordenadas, generando problemas cuando los objetos son muy grandes. Debido a que las diferencias son mayores en comparación con las predicciones de los cuadros delimitadores para objetos más pequeños. El problema mencionado es solucionado en YOLOv4. Una vez entrenada la arquitectura completa, es decir, una vez que la estructura "backbone", "neck" y "head" hayan obtenido pesos para sus neuronas que optimicen la pérdida, puede realizar inferencia sobre imágenes, no antes vistas, utilizando dichos pesos adquiridos. Esta inferencia funciona de igual forma que el entrenamiento, diferenciándose en cómo se realiza la detección. Por una parte, el entrenamiento considera como una detección, aquella cuyo cuadro delimitador predicho es el más parecido al cuadro delimitador real. Mientras que en la detección hecha en la inferencia, se considera todas las detecciones a través de un procedimiento llamado Non Maximum Suppression. Este proceso consiste en tomar todas las detecciones hecha por el modelo, y eliminar las predicciones cuya confianza no sobrepasa un umbral dado. Luego, de las detecciones que quedan, se eliminan todas las que compartan mucho espacio en común dejando la detección que tenga mayor confianza.

3.4.1. YOLOv4

YOLOv4 es presentado por Bochkovskiy et al. (2020), y no tiene cambios estructurales en la arquitectura. Se enfoca principalmente en cambios en módulos de la inferencia, estrategia

de entrenamiento y otros cambios que mejoran la precisión. Entre estos cambios, resaltan el aumento de datos en la etapa de entrenamiento donde se utilizan distintos tipos de técnicas, siendo la que destaca el aumento mosaico. Esta técnica consta en tomar cuatro imágenes del set de datos de entrenamiento y crear un mosaico como nueva imagen para el modelo. Otro cambio es la activación Mish que mejora las métricas en el entrenamiento de YOLO. Esta función de activación que es propuesta por Misra (2019), es una función no monotónica que tiene un dominio positivo ilimitado y un dominio negativo limitado. Esto permite un rápido aprendizaje y evita la muerte de neuronas. Finalmente, uno de los últimos cambios importantes es el uso de CIoU-loss (Zheng et al., 2020), un regresor para la obtención de los cuadros delimitadores que converge más rápido al basarse en tres factores geométricos. El primero es el área de superposición entre el cuadro predicho y el cuadro delimitador de la etiqueta que mide el porcentaje de error en la detección de los cuadros delimitadores predichos. El segundo factor es el punto central entre el cuadro predicho y el cuadro perteneciente a la etiqueta. Por último, la relación de aspecto entre el cuadro delimitador predicho y el real.

Cabe mencionar que YOLOv4 utiliza como estructura CSPDarknet-53 (Wang et al., 2019a) como “backbone”, que es una red neuronal convolucional diseñada para la detección de objetos que utiliza Darknet-53 compuesta por 53 capas convolucionales. Usando una estrategia de Cross-Stage-Partial-connections (CSP por sus siglas en inglés) Wang et al. (2019a), es decir, dividiendo el mapa de características de la capa base en dos partes y luego fusionándolas a través de una jerarquía entre etapas. YOLOv4, luego de tomar las características extraídas por el “backbones”, utiliza la estructura “neck” para agruparlas y pasarlas como entrada a la estructura “head” que se encarga de hacer las distintas predicciones. La estructura “neck” es PANet del método “Path Aggregation Network” Liu et al. (2018) y YOLOv3 como “head”, respectivamente.

3.5. Scaled YOLOv4

Scaled YOLOv4, que será utilizada en el presente estudio, utiliza como base la estructura de YOLOv4, rediseñándolo para proponer YOLOv4-CSP, donde aparte de aplicar CSP en la estructura “backbone” también lo hace en la estructura “neck” llamada PANet como se ve en la Figura 3.7, lo que logra una reducción del 40 % en operaciones de cálculo. En esta figura

se observa cómo se aplica la estrategia CSP. A la izquierda de la imagen se observa como una entrada pasa por una red neuronal convolucional. Esta entrada, pasa por la primera capa de la red y genera un mapa de características que será la entrada de la siguiente capa, volviendo a hacer el mismo procedimiento y creando una entrada para la siguiente capa hasta llegar al final y dar un resultado. En la parte derecha de la imagen se observa la estrategia CSP donde el mapa de característica creado por una capa es dividido. De esta partición una mitad será la entrada de la siguiente capa de la red y la mitad restante será concatenada capas más adelante. Esto reduce el nivel de cómputo y mantiene buenos resultados.

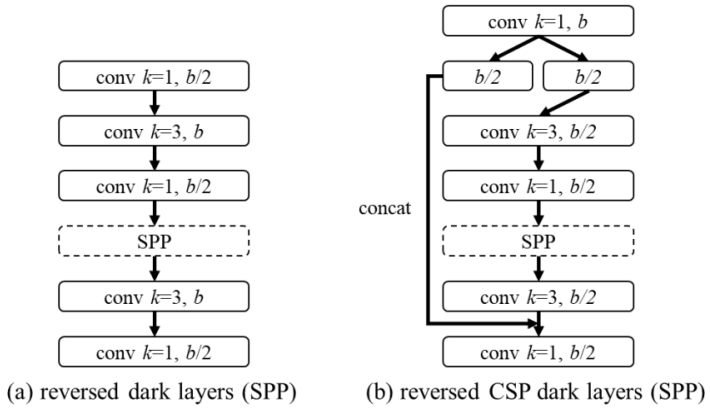


Figura 3.7: Aplicación CSP en PANet. Fuente Wang et al. (2020a).

En cuanto al escalamiento, es común observar que dichas técnicas aumentan la profundidad de la arquitectura en el ”backbone”, tal como lo hace Simonyan y Zisserman (2014) o bien, la arquitectura ResNet propuesta por He et al. (2016). En resumen, se observa que una técnica de escalamiento es el aumento en la profundidad de la red aumentando el número de capas. Como vemos en el segundo trabajo citado de este párrafo, la arquitectura pasa de tener 50 capas a un máximo de 152, pasando por arquitecturas intermedias. Otra método de escalado es el propuesto por Zagoruyko y Komodakis (2016) como lo menciona Wang et al. (2020a), donde cambia el número de kernel de la red convolucional.

Scaled YOLOv4 se basa en el trabajo realizado por Google Brain donde usan método de escalado hacia arriba y abajo en profundidad, lo que quiere decir que puede aumentar o disminuir la cantidad de capas convolucionales del modelo. También puede variar el ancho de la red

completa, o sea el número de kernel en cada convolución. Por último, este modelo también utiliza en la entrada distintas resoluciones de la imagen de entrada, es decir, cambios en el tamaño y nivel de detalle de la imagen que entra a la red para ser procesada por las diferentes capas. De esta manera se logra encontrar una combinación óptima entre rapidez y precisión, donde a mayor número de capas convolucionales o mayor número de kernels se observará un tiempo mayor de entrenamiento e inferencia.

La estructura de Scaled YOLOv4 large se muestra en la Figura 3.8, donde YOLOv4-p7 es la estructura más ancha, profunda y que espera la entrada con resolución más alta de todas las estructuras. A su vez, YOLOv4-p5, es el modelo escalado más pequeño en ancho y profundidad que espera como entrada la imagen con resolución más pequeña y es demarcado con una línea roja.

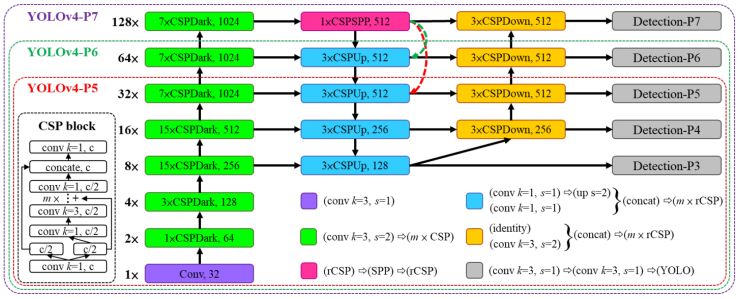


Figura 3.8: Arquitectura Scaled YOLOv4 large que incluye YOLOv4-P5, YOLOv4-P6 y YOLOv4-P7. Fuente: Wang et al. (2020a).

La columna verde representa el extractor de características mientras que las columnas celestes y amarillas representan la estructura que procesan la información entregada por el "backbone" y le da como entrada a la detección representada por la columna gris. Se puede observar que a medida que el modelo va aumentando en tamaño y complejidad (YOLOv4-p6 y YOLOv4-p7), el número de capas en el extractor y "neck" aumentan al igual que el número en el detector. A la izquierda de la figura se vuelve a representar la estrategia CSP para hacer énfasis en que todo el modelo sigue esta idea.

3.6. YOLOv5

YOLOv5 es un modelo donde se publica su repositorio, sin embargo, este modelo no tiene un paper publicado donde se muestren sus cambios oficiales. Sumado a esto, cabe mencionar que dicho repositorio fue lanzado en menos de dos meses una vez fue publicado YOLOv4.

Según algunos datos destacados pertenecientes a blogs de Nelson y Solawetz., autores de las primeras tres versiones de YOLO, indica que la quinta versión de esta familia de detectores es escrito en Pytorch lo que al ser una biblioteca bastante utilizada por la comunidad, provocando que el modelo se beneficie del ecosistema, haciendo además al modelo más simple (Nelson y Solawetz.).

Otro cambio notorio indicado por Nelson y Solawetz, es el tamaño de esta última versión, donde se ve reducida en comparación a YOLOv4 en casi un 90 % lo que se evidencia notoriamente en los tiempos de entrenamiento. Sin embargo, en cuanto a los resultados de la inferencia no se nota una mayor variación (Nelson y Solawetz).

Capítulo 4

Metodología

En este capítulo se presenta la metodología utilizada para el desarrollo del presente estudio, es decir, desde la adquisición de los datos hasta las métricas obtenidas en la implementación de los distintos algoritmos utilizados. Para ello, se explica el tipo de dato utilizado, el procesamiento de estos, con el fin de obtener una entrada válida para los distintos modelos y se muestra la implementación de éstos.

4.1. Set de datos

Los datos son proporcionados por Celulosa Arauco, y constan de imágenes aéreas multiespectrales de distintos predios forestales ubicados en la Región del Biobío, como se observa en la Figura 4.1. Los terrenos mostrados en las distintas imágenes presentan cultivos de dos tipos, pino y eucalipto, árboles típicos en las plantaciones forestales chilenas de la región. Sin embargo, en el presente trabajo, se utiliza sólo predios forestales de eucalipto.

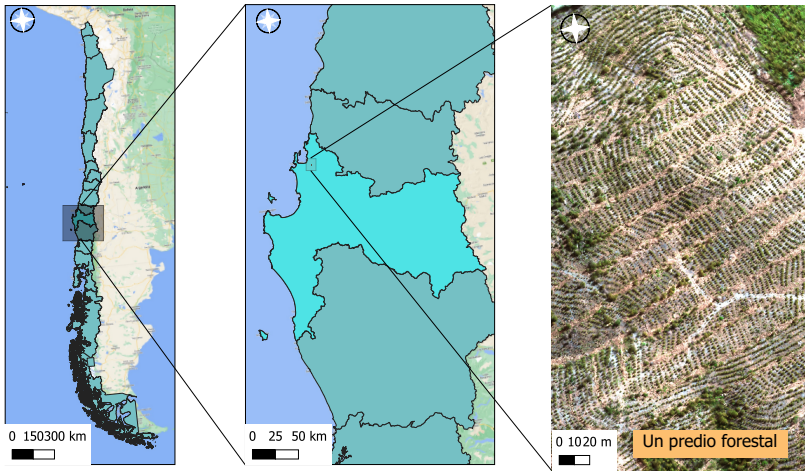


Figura 4.1: Zona de estudio.

Al ser imágenes multiespectrales, tienen cinco bandas de distintas longitudes de ondas reflejadas por los distintos predios forestales, los cuales son las tres bandas RGB más dos bandas del infrarrojo cercano.

4.2. Preprocesamiento

La imagen proporcionada posee información de una superficie bastante extensa, que contiene una cantidad de árboles considerables y por ende, el análisis de toda la información en un solo paso perjudicaría el desempeño del modelo usado. A su vez, al tener una gran cantidad de pixeles que analizar, los tiempos de cómputo serían bastantes extensos. Por ello, se toma una pequeña porción de esta imagen multiespectral para crear un set de entrenamiento, validación y testeo haciendo posible la aplicación de los distintos algoritmos y la metodología en general. En las Subsecciones 4.2.1, 4.2.2 y 4.2.3 se explican procesos que suceden en paralelo. Primero, el ordenamiento espacial de la imagen entregada, el etiquetado de las zonas de interés y el recorte de estas zonas.

4.2.1. Ordenamiento de la imagen original

Para trabajar con una porción más pequeña de la imagen completa, se usa un archivo Shapefile que ordena la imagen en celdas como se muestra en la Figura 4.2 y se seleccionan algunas de ellas para usarlas como set de entrenamiento, validación y testeo. Para ello, se seleccionarán N celdas, donde el 60 % es usada para el entrenamiento, 20 % para la validación y 20 % para el conjunto de prueba. Cabe mencionar que un archivo shapefile es un formato no topológico para almacenar la ubicación geométrica y de atributos de entidades geográficas. Éstos archivos pueden ser representados por puntos, líneas o polígonos. Estos últimos, es decir, los multilátero, vienen a representar áreas en el espacio que contienen características de interés para quienes lo estudian. Por ejemplo, para la grilla armada, cada celda son un polígono que contiene información y atributos como la ubicación geográfica, si corresponde o no a una celda de interés o no, si en esa celda hay árboles etiquetados, entre otros.

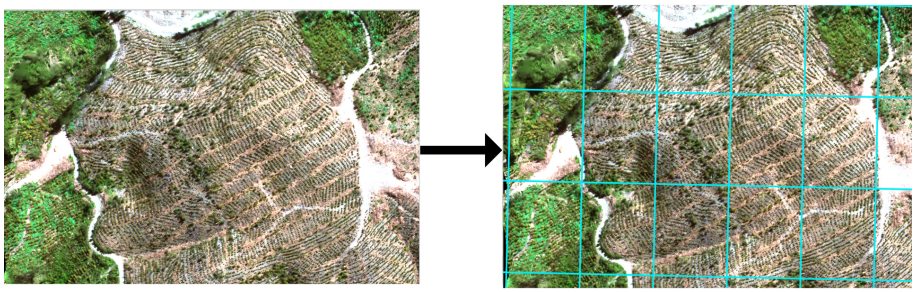


Figura 4.2: Grilla sobre la imagen completa.

4.2.2. Etiquetado de la imagen

Por otra parte, debido a que se trabaja con algoritmos de aprendizaje reforzado, es necesario proveerle de etiquetas a cada set de datos para permitir que los algoritmos puedan distinguir qué características deben aprender. Además, con estas etiquetas, se comparan las predicciones con las etiquetas reales, lo que posibilita la obtención de métricas en el proceso de entrenamiento,

validación y testeo. Para tener estas etiquetas, se hace necesario un proceso de etiquetado de cada una de las celdas seleccionadas. Este procedimiento se realiza a través de el software QGIS que trabaja con sistemas de información geográfica.

Este procedimiento consiste en la creación de un archivo shapefile que tiene la mismas dimensiones de la imagen original y dentro de ella se establecen polígonos rectangulares que indican la ubicación de cada elemento de interés. Es decir, se ubica dentro del polígono rectangular o cuadro delimitador cada árbol existente en la celda seleccionada. Este archivo shapefile complementa la información contenida en la imagen, señalando las ubicaciones espaciales en las que se encuentra cada árbol.

Es importante mencionar que un buen proceso de etiquetado, puede ser clave para el correcto funcionamiento de los algoritmos. Debido a que aunque el modelo tenga la capacidad de aprender lo que se le está pidiendo, si existe un mal etiquetado, el modelo predecirá incorrectamente. Por otra parte este proceso de ordenamiento y etiquetado es realizado en el software QGIS que trabaja con sistemas de información geográfica (QGIS Development Team).

4.2.3. Proceso de recorte

Con el fin de tener un set de datos aún más numeroso, y compuesto por imágenes que logren un nivel de detalle que facilite el análisis a los modelos, se hacen subrecortes a estas celdas seleccionadas recortadas de la imagen original. Cada una de estas celdas seleccionadas, es recorrida a través de una ventana deslizante de tamaño igual a 128 pixeles, generando recortes en las cinco bandas de la celda seleccionada del tamaño de dicha ventana. Posteriormente, la ventana se desliza de izquierda a derecha y de arriba hacia abajo, en desplazamientos de 32 pixeles recorriendo toda la celda y generando varios recortes como se observa en la Figura 4.3. Una vez realizado los recortes, se puede observar que cada recorte tiene dimensión 128×128×5, es decir, un recorte de 128×128 por cada banda. Tres para la banda rojo, verde y azul respectivamente más dos pertenecientes a bandas del infrarrojo cercano. A su vez, en dicho recorte, se guarda la información de las etiquetas de los árboles contenidos guardando las coordenadas del centro del objeto de interés más el alto y ancho de éste en un archivo de texto que permite el funcionamiento de los algoritmos. Luego, se

normalizan los subrecortes en el rango de 0 a 255 para la entrada de los algoritmos. Este proceso se realiza siguiendo la lógica de la Ecuación (4.1) para cada pixel de cada banda. O sea, se calcula el valor máximo y mínimo. Posteriormente, a cada pixel de la imagen se le resta el valor mínimo y se le divide por la diferencia entre el valor máximo y mínimo contenido en la imagen.

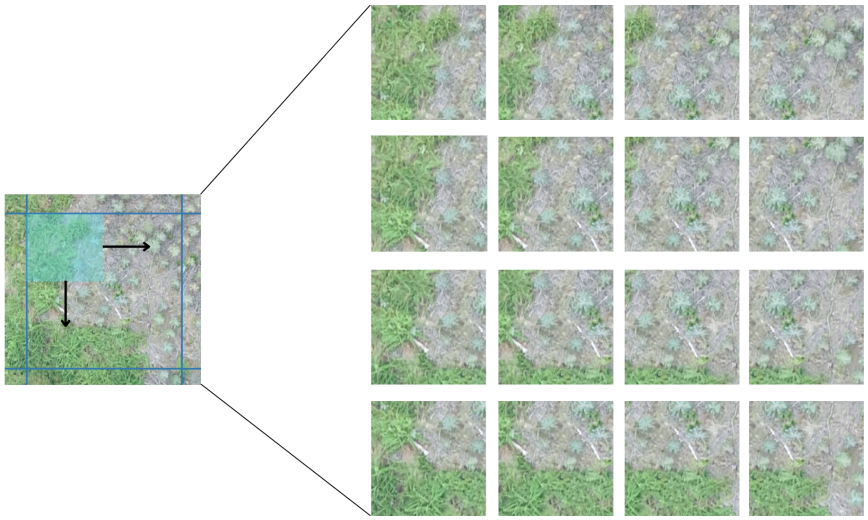


Figura 4.3: Proceso de recorte de una celda.

$$Img = 255 \times \frac{Img - Img_{min}}{Img_{max} - Img_{min}} \tag{4.1}$$

4.3. Implementación de Scaled YOLOv4 y YOLOv5

Una vez realizado el procesamiento de datos, se tiene lo suficiente para comenzar a utilizar los distintos modelos Scaled YOLOv4 y YOLOv5. Para ello, se utilizan ambos repositorios respectivos. A estos repositorios se le realizan modificaciones que se ajusten al tipo de imagen utilizado y a las necesidades de la investigación.

Ambos modelos esperan como entrada imágenes RGB para el entrenamiento, validación y testeo. Como en este caso de estudio son usadas imágenes multiespectrales que contienen cinco bandas. Por lo tanto, es necesaria la modificación de la capa de entrada de la red neuronal

para que cada imagen en forma de tensor pueda ingresar a la red. Además se realiza otros tipos de modificaciones, en scripts para el correcto funcionamiento de ambos modelos, tales como la lectura de datos, detección y funciones auxiliares. Estas modificaciones se realizan para YOLOv5 y para cada configuración obtenido en Scaled YOLOv4.

Para que el algoritmo comience a entrenar con pesos preentrenados y reducir los tiempos de entrenamiento, se realiza transferencia de aprendizaje de los pesos obtenidos por los diseñadores de YOLO, quienes realizaron anteriormente distintos entrenamientos en el set de datos de COCO. Los pesos obtenidos en dichos entrenamientos, son traspasados a la arquitectura completa. Dado que para este proyecto el modelo se usa para tareas específicas, una vez hecha la transferencia de pesos, se dejan inicializados de tal manera y se comienza a entrenar para la nueva tarea que es la detección de árboles. A este proceso se le denomina "Fine tuning". Con ello, el modelo comienza a entrenar para la detección de árboles con ciertas formas, líneas, texturas, colores entre otras características, permitiendo un rápido entrenamiento. Cabe mencionar que los pesos adquiridos por los creadores de YOLO, al entrenar el algoritmo con el set de datos de COCO, se logra generalizar y aprender con mayor facilidad las características de las imágenes en general.

4.3.1. Búsqueda de hiperparámetros

Para realizar el entrenamiento, se realiza previamente una búsqueda de hiperparámetros que se ajusta de mejor manera a los datos utilizados en el estudio. En la Tabla 4.1, se muestra los hiperparámetros buscados por cada modelo seguido del seleccionado. En esta tabla, el término "batch size" hace referencia al número de muestras (imágenes del set de datos) que pasa por la red mientras este es entrenado para realizar todo el proceso. El tamaño de imagen es la cantidad de pixeles que posee cada imagen que entra a la red, donde a mayor tamaño tiene un mayor nivel de detalle y con ello una mayor resolución. El concepto de "learning rate" hace referencia a la tasa de ajuste de un algoritmo de optimización que busca minimizar la función de pérdida. Por tanto, a un mayor "learning rate" mayor es la variación de los pesos de una iteración a otra, en comparación a una tasa de aprendizaje más pequeña. Por último Optimizador en base al

learning rate se encarga de variar el valor de los parámetros de la red. Notar que Scaled YOLOv4 mantiene el tamaño de imagen fijo dependiendo del tamaño de la red según lo señalado por los autores, o sea, si se trabaja con Scaled YOLOv4-p5, entonces se debe utilizar un tamaño de imagen igual a 896, mientras que para YOLOv4-p6 y YOLOv4-p7 se debe utilizar un tamaño de imagen igual a 1280 y 1536, respectivamente. Además, es importante mencionar que no se prueba con un número de batch size mayor de los que aparecen en la tabla debido a la memoria disponible en la máquina utilizada para los distintos experimentos.

	Hiperparámetro	**Set de búsqueda**	**Seleccionado**
YOLOv5	Optimizador	{SGD ; Adam ; AdamW}	AdamW
	Tamaño de imagen	{640 ; 896 ; 1280 ; 1536}	896
	Batch size	{2 ; 4}	2
	Learning rate	{0,0001 ; 0,001 ; 0,01 ; 0,1}	0,001
Scaled YOLOv4-p5	Optimizador	{SGD ; Adam ; AdamW}	AdamW
	Tamaño de imagen	{896}	896
	Batch size	{2 ; 4}	4
	Learning rate	{0,0001 ; 0,001 ; 0,01 ; 0,1}	0,0001
Scaled YOLOv4-p6	Optimizador	{SGD ; Adam ; AdamW}	Adam
	Tamaño de imagen	{1280}	1280
	Batch size	{2}	2
	Learning rate	{0,0001 ; 0,001 ; 0,01 ; 0,1}	0,001
Scaled YOLOv4-p7	Optimizador	{SGD ; Adam ; AdamW}	AdamW
	Tamaño de imagen	{1536}	1536
	Batch Size	{2}	2
	Learning rate	{0,0001 ; 0,001 ; 0,01 ; 0,1}	0,0001

Tabla 4.1: Búsqueda de hiperparámetros.

La búsqueda se realiza a través de un proceso denominado validación cruzada. Se utiliza 4 fold cross validation, es decir, por cada combinación de hiperparámetros, la red se entrena cuatro veces utilizando cuatro celdas y se varia en cada iteración las celdas destinadas al entrenamiento y la validación como se ilustra en la Figura 4.4. Una vez realizado los cuatro entrenamientos, se obtiene el promedio y desviación estándar de la métrica $mAP_{0,5}$ que es discutida posteriormente. Por tanto, se selecciona la combinación de hiperparámetros que obtuvo el promedio mayor y a su vez desviación estándar menor.

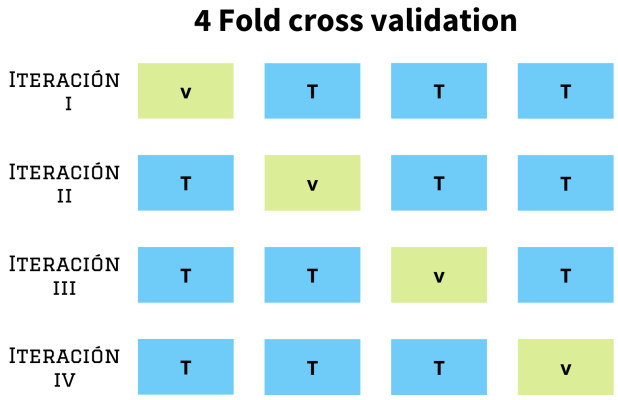


Figura 4.4: 4 Fold cross validation.

4.3.2. 4 Fold cross validation

La Figura 4.4 muestra la conformación del conjunto de entrenamiento para cada iteración para la búsqueda de los mejores hiperparámetros. Donde, de todos los datos proporcionados para el entrenamiento, se hace una partición en cuatro partes iguales (cada cuadro de cada fila). De esas cuatro particiones, cada una corresponde a una celda etiquetada, se escoge una celda para la validación y se deja el resto para el entrenamiento. La primera iteración es entrenada con el conjunto de datos destinada al entrenamiento, es decir, los tres cuartos de datos seleccionados y se obtiene el mejor modelo. Luego, se procede con la segunda iteración (segunda fila), donde los conjuntos de datos son cambiados, es decir, el set de validación debe ser totalmente distinto y proporcional en número al set de validación de la iteración uno. Se entrena el modelo con ese set de datos y se vuelve a obtener resultados. Se procede a realizar la misma metodología hasta completar cuatro iteraciones. Cada iteración ocupa un conjunto de validación distinta. En cada iteración, se obtienen métricas del modelo en general sobre el conjunto de datos completos. De manera que haciendo esta partición y validando en cada "split", se puede observar como se generaliza el modelo sobre los datos completos. Evitando el caso que el modelo este entrenando, haciendo validación y testeo sobre un set de datos sencillo de interpretar o inferir. Por lo que los número entregados, serían poco confiables y representativos al momento de ingresar más datos del mismo tipo.

4.3.3. Entrenamiento de los modelos

Se realiza el entrenamiento de los distintos algoritmos, que consiste en realizar N veces el siguiente procedimiento, con el fin de lograr que estos aprendan a identificar las características dentro de los tensores que contienen la información de cada imagen. En cada una de estas N épocas, el modelo recibe como entrada cada imagen del set de datos de entrenamiento en lotes definidos previamente y en base a los pesos que tiene la arquitectura neuronal en ese momento dado, se realiza distintas operaciones hasta predecir los cuadros delimitadores y las respectivas clases del objeto contenido en dicho cuadro que estén dentro de la imagen que está de paso por la red. Estas predicciones, se comparan con las etiquetas reales de cada árbol que contienen el cuadro delimitador real y su respectiva clase. Al comparar estas dos conjuntos de valores, es decir, los predichos y los reales, se estima el error con la función de pérdida que sirve para ajustar los pesos de la red neuronal en el proceso de retropropagación qué cuantifica la responsabilidad de cada neurona en cada capa para haber obtenido dicho error calculado. Utilizando el learning rate en conjunto del optimizador, se utilizan algoritmos que minimizan la función de error como el descenso del gradiente que intenta encontrar los parámetros del modelo que disminuyan la diferencia entre el valor predicho y el real. Realizado el procedimiento anterior, para todo el conjunto de entrenamiento, se guardan los pesos finales adquiridos al final de este proceso y se realiza una evaluación con el conjunto de datos de validación, los cuales en el entrenamiento no fueron vistos. Se obtienen distintas métricas que se discuten en la Sección 4.4 con el fin de medir qué tanto mejora el algoritmo en la etapa de entrenamiento en esa época dada. Realizado el procedimiento, se finaliza una época para comenzar con la siguiente hasta completar el número de N épocas definidas a priori. Una vez finalizadas todas las épocas, se guardan los pesos que obtuvieron mejores métricas en la validación y sirven para realizar inferencia en el set de datos de testeo. Los cuales son datos que jamás ha procesado el algoritmo y por ende, se puede observar qué tan bien generaliza el modelo y qué tan bien realiza su tarea en datos que nunca ha visto. Este set de testeo servirá para reportar las métricas finales.

4.3.4. Inferencia de los modelos

La tarea de predecir sobre cada una de las imágenes recortadas de la celda de testeo, tal como se hace con las imágenes del set de entrenamiento y validación, no se alinea con el objetivo del problema que es detectar y contar sobre el predio completo. Esto debido a que las imágenes tienen un cierto solape, y por consecuencia, un árbol, no necesariamente está en una sola imagen. Por ello, se estaría detectando y contando un árbol dos veces. Para resolverlo se utiliza la propuesta de Pérez et al. (2021), donde se realiza una detección sobre la imagen del set de testeo y en base a dicha detección, se reconstruye la celda original considerando el centroide de la predicción como se muestra en la Figura 4.5. En esta figura se observa el proceso inverso al recorte de imágenes. En la imagen ubicada más a la izquierda se observa el recuadro marcado en negro que simboliza un recorte. En el centro de este cuadro negro, se observa un área azul que simboliza el área de confianza que es utilizada para la reconstrucción de la imagen completa. Esto se hace debido a que todos los recortes tienen un solape y no se quiere repetir la información. En la imagen del medio, se toma el segundo recorte con solape y se toma en consideración solo el área de confianza, concatenándola a la región de confianza de la imagen anterior. Este procedimiento se realiza con todos los recortes hechos en un inicio posibilitando la reconstrucción de la imagen completa sin repetir información duplicada en todos los recortes.

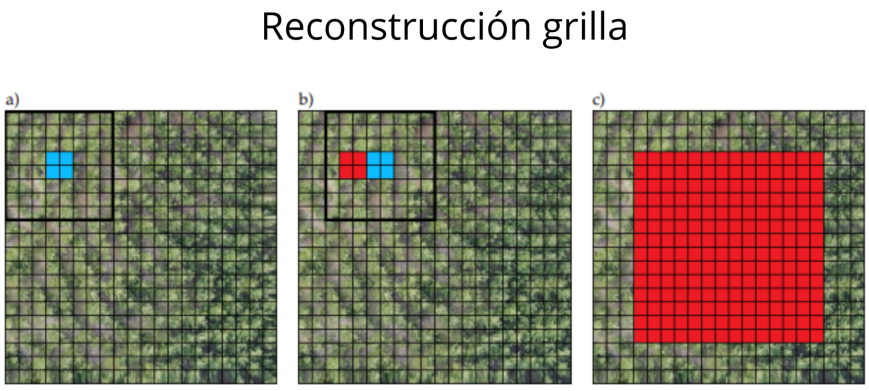


Figura 4.5: Reconstrucción de la celda en base a las predicciones hechas en el set de testeo. Fuente: Pérez et al. (2021).

4.4. Evaluación de la metodología

La metodología anteriormente mencionada se evalúa en base a los resultados de las detecciones realizadas en el set de testeo, las cuales también son aplicadas para el entendimiento de las métricas obtenidas en cada entrenamiento en base al set de validación.

Dentro de las principales métricas para realizar evaluaciones se encuentra los verdaderos positivos, falsos positivos, falsos negativos, precision, recall y AP (Ammar et al., 2021). Cabe mencionar que AP, contienen en su cálculo a todas las métricas mencionadas anteriormente. Por ello, AP o $AP_{0,5}$ es el principal indicador para evaluar qué tan bien están realizando la tarea de detección los distintos algoritmos. Mientras, la tarea de conteo se evalúa qué tan cerca está el número estimado de árboles respecto al número existente en el área de evaluación. La base de todas las métricas mencionadas anteriormente es el concepto de "Intersection over Union" (IoU). IoU es la razón del área de intersección sobre el área de la unión entre un cuadro delimitador predicho y uno real como se muestra en la Figura 4.6 (Ammar et al., 2021).

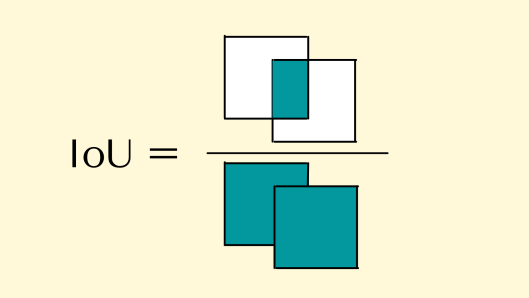


Figura 4.6: Intersection over Union.

Por otro lado, el concepto de "verdadero positivo" hace referencia a todo elemento que se clasifica dentro de una categoría y realmente lo es. Llevando este concepto a la tarea de detección por medio de la visión computacional, una predicción hecha por el algoritmo es considerada como verdadero positivo cuando dicha detección realmente indique la existencia de un árbol. Sin embargo, tal como un cuadro delimitador podría contener completamente a un árbol, se pude dar el caso que un cuadro delimitador, sólo contenga una parte pequeña del árbol que dice detectar. Planteado este problema, surge la siguiente pregunta: ¿cuál es el límite o línea para considerar una detección correcta? Para saber si la detección es considerada un verdadero

positivo (VP). Por tanto, se recurre al uso de IoU, estableciendo un umbral apriori generalmente mayor o igual a 0,5 debido a que se lograría que más del 50 % de la predicción sea similar a la etiqueta real. En tanto, si el área de intersección sobre la unión entre la predicción con el cuadro delimitador real es mayor a 0,5 o el umbral dado, entonces es considerado una detección correcta. mientras, un "falso positivo" (FP) es una detección hecha por el algoritmo que no supera el umbral, al calcular el IoU con el cuadro delimitador real. Finalmente, un "falso negativo" (FN) es todo árbol existente en la imagen y que no es detectado por el algoritmo.

Las métricas precision y recall, tienen sus enfoque en la Ecuación (4.2) y (4.3). Donde, se observa que la primera indica la razón de los verdaderos positivos entre todas las predicciones hechas y la segunda muestra los verdaderos positivos que fueron detectados del total de predicciones hechas.

$$Precision = \frac{VP}{VP + FP} \tag{4.2}$$

$$Recall = \frac{VP}{VP + FN} \tag{4.3}$$

En base a la métrica precision y recall, se puede armar la curva PR que muestra el valor de cada métrica en cada detección hecha para una imagen de entrada dada. Bajo este procedimiento, se observa que en una imagen con n árboles a detectar, la primera detección considerada como "verdadero positivo" (VP), la métrica recall sería $\frac{1}{n}$. Además, si se asume una cantidad infinita de predicciones, en la última detección hecha, la métrica precision es el número de detecciones consideradas VP sobre infinito ($\frac{VP}{\infty}$). Obtenida la métrica precision y recall en cada detección, se puede graficar la curva PR dada las detecciones realizadas en la imagen de entrada que ha sido analizada. Esta curva, generalmente tiene una forma similar como la que se ve en la Figura 4.7.

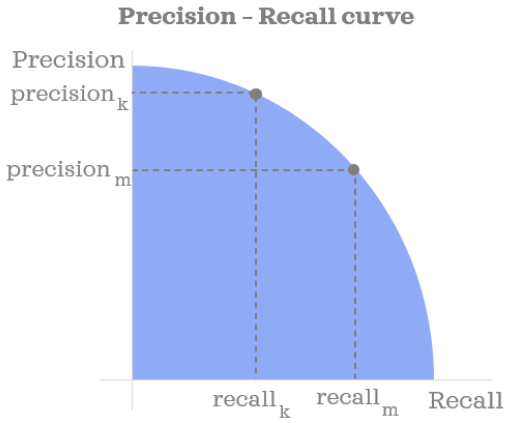


Figura 4.7: Curva Precision-Recall.

El área bajo la curva PR se denomina Average Precision (AP), el cual es diferente según el umbral con el que se define un verdadero positivo. Por ejemplo, si dicho umbral para obtener el IoU es 0,5, entonces, se dice que la métrica con la que se evalúa es $AP_{0,5}$. Dado que la evaluación del modelo se hace sobre la reconstrucción de una sola imagen, la métrica para evaluar es $AP_{0,5}$. Sin embargo, es importante conocer el concepto de mean Average Precision (mAP) y sus derivados para entender cómo se midió el entrenamiento.

Cuando se tiene un conjunto de imágenes, donde se hace detecciones como es el conjunto de entrenamiento o evaluación en el entrenamiento, el promedio de todos los AP obtenidos en cada imagen del set de datos es el mAP del conjunto de datos de i imágenes. mAP es calculada según la Ecuación (4.4).

$$mAP = \frac{\sum_{i=1}^{n} AP_i}{n} \tag{4.4}$$

Donde AP_i es la métrica AP obtenida en la imagen i y n es el total de muestras.

Por otra parte $mAP_{0,5}$ es el promedio de todos los AP obtenidos en el set de datos dado. A su vez, tal como $mAP_{0,5}$, puede existir el promedio de AP para diferentes umbrales como 0,55; 0,6; 0,65; 0,7; 0,75; 0,8; 0,85; 0,9 y 0,95.

Capítulo 5

Resultados experimentales y discusión

En este capítulo se muestran los resultados obtenidos en la tarea de conteo y detección de YOLOv5 y Scaled YOLOv4 al aplicar la metodología presentada en el capítulo anterior. Para la obtención de estos resultados, se utilizan los datos provenientes de la celda de testeo, donde ninguno de los dos modelos ha procesado en ningún momento y se puede evaluar qué tan bien está generalizando cada uno de los detectores.

5.1. Conteo

Al evaluar estos modelos en la tarea de conteo, se mide la capacidad de poder predecir la cantidad de árboles existentes en el predio forestal o una parte de éste que sea señalado como se muestra en la Figura 5.1. Siendo este número predicho la única salida del modelo, es decir, en esta tarea, no toma importancia la ubicación espacial del árbol ni la predicción referida al área que abarcaría cada uno de éstos en la imagen. Las métricas obtenidas por cada modelo se presentan en la Tabla 5.1, junto con la diferencia del valor predicho respecto al número total de árboles existentes en la única celda de testeo igual a 936. Cabe mencionar que sólo se ocupa un predio forestal para obtener las métricas a diferencia del trabajo realizado por Pérez et al. (2021).

Modelo	Conteo	Dif (%)
YOLOv5	773	-17,4 %
Scaled YOLOv4-p5	928	-0,9 %
Scaled YOLOv4-p6	**937**	**0,1 %**
Scaled YOLOv4-p7	888	-5,1 %

Tabla 5.1: Métricas en la tarea de conteo.

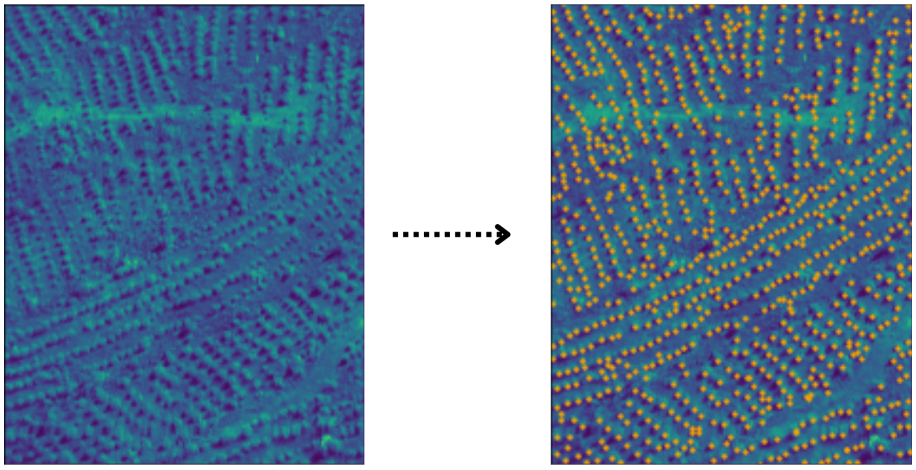


Figura 5.1: Conteo en la celda de testeo.

Se puede observar que Scaled YOLOv4-p6, logra la menor diferencia entre la cantidad de árboles predichas y las existentes en la celda de testeo, sobrestimando por una unidad la cantidad real de árboles plantados en esta celda. A esta mejor estimación, le siguen los resultados obtenidos por Scaled YOLOv4 en sus distintos tamaños. YOLOv5 en esta tarea, es el que posee el registro más alto en cuanto a la diferencia de la estimación de árboles en la celda de testeo y el número real de éstos. Si se observan las métricas obtenidas por este modelo en la tarea de detección, se podría llegar a pensar de que la estrategia que utiliza para minimiza el error fue ser cauteloso al momento de "inventar" árboles, provocando la mínima cantidad de falsos positivos y la máxima cantidad de falsos negativos, afectando a la tarea de conteo estimando una cantidad de árboles muy por debajo de la real.

5.2. Detección

Al momento de evaluar la tarea de detección, se mide en cada modelo qué tan bien puede predecir la ubicación espacial de cada árbol en la imagen, en conjunto del área que éste está abarcando. En otras palabras, cada modelo debe predecir un cuadro delimitador en la imagen para indicar que ha realizado una detección de un objeto de interés tal como se muestra en la Figura 5.2. Este cuadro debe coincidir en su totalidad o gran parte, con la ubicación y dimensiones de la etiqueta del árbol para clasificar dicha predicción como un verdadero positivo.

La Tabla 5.2 muestra el modelo utilizado en conjunto de las métricas obtenidas, VP, FP, FN, Precision, Recall y $AP_{0,5}$. Este análisis se realiza teniendo en cuenta que el número de árboles existentes en la celda de testeo son 936, tal como en la tarea de conteo. Se puede evidenciar que el área bajo la curva "Precision-Recall" (PR) mayor es obtenida por Scaled YOLOv4-p6, logrando 76 % en la métrica $AP_{0,5}$ y registrando los mejores resultados en "Verdaderos positivos" y "Recall".

Modelo	**VP**	**FP**	**FN**	**Precision**	**Recall**	$AP_{0,5}$
YOLOv5	591	**182**	345	**76 %**	63 %	71 %
Scaled YOLOv4-p5	666	262	270	72 %	71 %	69 %
Scaled YOLOv4-p6	**693**	244	**243**	74 %	**74 %**	**76 %**
Scaled YOLOv4-p7	611	277	325	69 %	65 %	68 %

Tabla 5.2: Métricas en la tarea de detección.

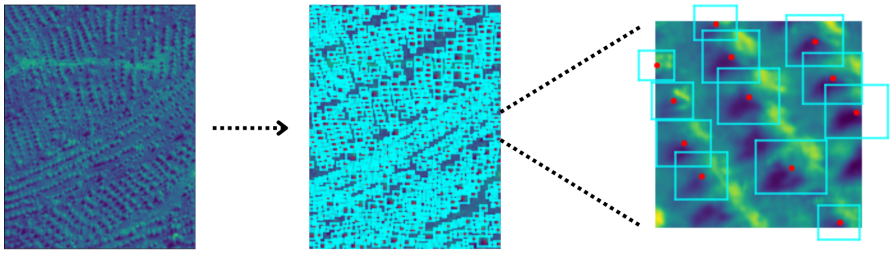


Figura 5.2: Detección en la celda de testeo.

Además, se puede observar que YOLOv5 es el modelo que menos se equivoca al momento de detectar árboles donde realmente no los hay (FP). Esto permite obtener el mejor puntaje en la métrica "Precision" pero a su vez el peor conteo registrado. También, este modelo registra el segundo mejor $AP_{0,5}$ de los modelos estudiados.

Los resultados visuales respecto a las detecciones son mostrados en la Figura 5.3. En la parte izquierda, se evidencian los centroides de las predicciones consideradas como VP y FP, mientras que a la derecha se sobreponen los baricentros de las etiquetas. Se puede observar que prácticamente en la mayoría de los casos, el centro de cada detección considerada como "verdadero positivo" se ve cubierto por la etiqueta como lógicamente debería suceder. En los casos en que estas predicciones de tipo VP no son cubiertas, se debe a que el árbol en cuestión tiene un área considerable y por ende, los centroides tienen mayor margen de distancia. Se observa también el caso cuando algunos centroides de detecciones consideradas como "falsos positivos" son cubiertos por las etiquetas. Dando la contradicción, debido que detección debiese ser correcta, pero el modelo está diciendo lo contrario. Esto se debe a casos que se presentan en la Sección 5.5.

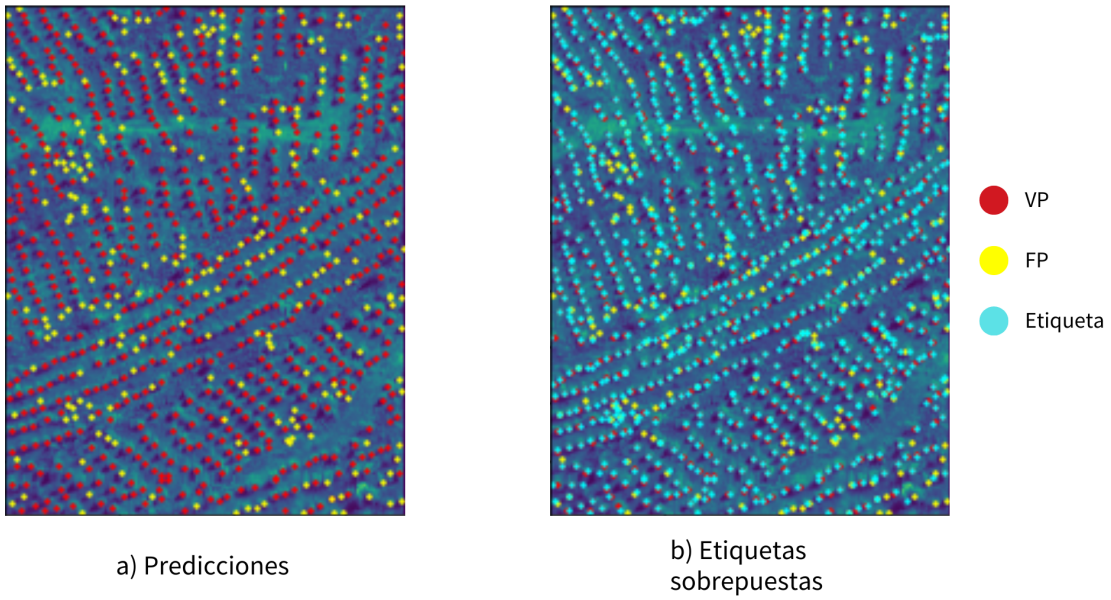


Figura 5.3: Predicciones YOLOv5.

5.3. Tiempos de entrenamiento

En esta sección, se indican los tiempos de entrenamiento que tuvieron los distintos modelos. Cabe mencionar que los entrenamientos se realizaron con una técnica de regularización llamada "parada temprana" utilizada por Ji et al. (2021) con una paciencia de 40 épocas. Esto significa que el modelo deja de entrenar una vez que el mejor mAP no es superado durante una cantidad de épocas dadas por la paciencia. El objetivo de aplicar la técnica de parada temprana es que se asume que el modelo converge y que no encuentra un mejor valor en las siguientes épocas.

En cuanto al tiempo de entrenamiento más corto obtenido es el de YOLOv5 con 0,338 horas, le sigue Scaled YOLOv4-p5, Scaled YOLOv4-p6 y Scaled YOLOv4-p7 con 0,854 , 1,608 y 3,944 horas de entrenamiento. Donde se evidencia que a mayor complejidad de la estructura, mayor tiempo de cómputo necesita.

5.4. Comparación de modelos propuestos

Realizando una evaluación global de los dos modelos respecto a la tarea de conteo y detección, se pude comentar que Scaled YOLOv4-p6 logra los mejores resultados en la tarea de detección y conteo. A su vez, Scaled YOLOv4, logra los mejores puntajes en el conteo en todos sus tamaños superando siempre a YOLOv5. En cuanto a la tarea de detección, los resultados son todos similares, sin embargo YOLOv5 obtiene el segundo mejor resultado.

Finalmente, cabe mencionar que las arquitecturas propuestas por los distintos autores, YOLOv5, Scaled YOLOv4-p5, Scaled YOLOv4-p6 y Scaled YOLOv4-p7, obtienen resultados relativamente similares. Sin embargo, YOLOv5 lo hace en un tiempo considerablemente menor.

En cuanto a la técnica de escalamiento que utiliza Scaled YOLOv4, se evidencia que son efectivas a excepción del modelo más grande Scaled YOLOv4-p7, que muestra un decaimiento en el desempeño. Esta observación, en conjunto de que registra el máximo $mAP_{0,5}$ más bajo en el entrenamiento en comparación a los otros modelos como muestra la Figura 5.4, se puede estimar que el modelo se hace tan complejo, que sumado a la poca cantidad de datos, éste tiende a "memorizar" más que generalizar.

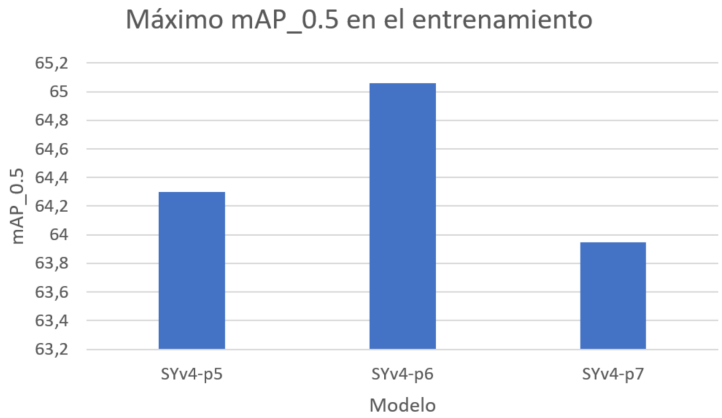


Figura 5.4: Mejor $mAP_{0,5}$ obtenido en el entrenamiento durante la validación.

5.5. Discusión

En base al método utilizado por Pérez et al. (2021), se presenta una metodología para realizar conteo y detección de árboles en predios forestales con algoritmos de aprendizaje profundo utilizando imágenes multiespectrales georreferenciadas. La metodología mencionada, puede ser usada en cualquier tipo de plantación de árboles mientras se usen el mismo tipo de datos. Si se compara los resultados del presente trabajo con los resultados obtenidos por el detector YOLOv3 en la investigación de Pérez et al. (2021) en la detección y conteo de árboles de 5 años que son los más similares en tamaño y forma a los de este estudio como se ve en la Figura 5.5, se puede evidenciar mejoras en ambas tareas. Mientras que en el conteo se logra una mejora de un 4,9 % con Scaled YOLOv4-p6, en la tarea de detección se logra una mejora con todos los modelos, obteniendo la mayor diferencia de un 12 % con el mismo modelo que logra la mejor métrica en el conteo.

A pesar de que en la detección no se pudo lograr una mejora frente al segmentador Mask R-CNN que utilizaron Pérez et al. (2021), los resultados obtenidos en el presente informe son competentes con este tipo de modelo y a su vez considerablemente más rápidos en cuanto a tiempo de entrenamiento e inferencia.

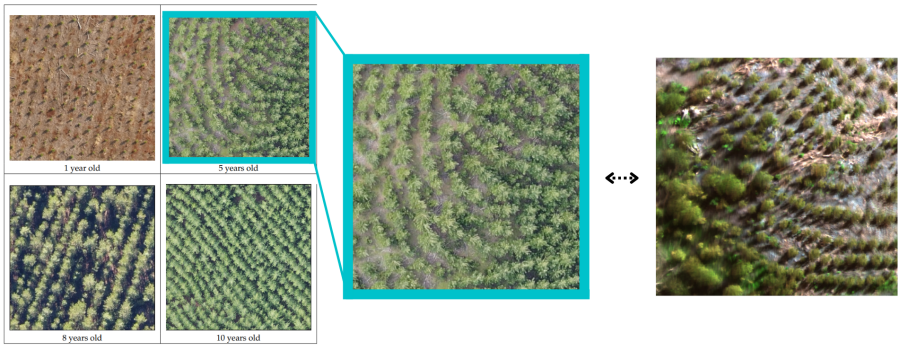


Figura 5.5: Comparativa de bosques entre el estudio de referencia y el presente.

Por otra parte, es importante mencionar que una de las características más destacadas

de YOLOv4 y Scaled YOLOv4 son sus técnicas de regularización llamada Aumentación de datos. Esto consiste en generar un volumen de datos mayor y más variado a partir del set de datos de entrenamiento proporcionado que busca cumplir el objetivo de generalizar y aprendan mejor. Tal como lo expone Bochkovskiy et al. (2020), el uso de estas distintas técnicas en el entrenamiento como "random scaling", "cropping", "flipping, and rotating" que son las más clásicas, en conjunto de estrategias más sofisticadas como "CutMix" y "Mosaic" pueden mejorar significativamente los resultados.

Dado que en el presente trabajo se hizo uso de datos multiespectrales, en consecuencia se reconfiguró la red y las funciones para trabajar con este tipo de datos, las funciones internas para realizar las aumentaciones mencionadas fueron deshabilitadas ya que no eran compatible con las bibliotecas que se implementaron. Por esto, se espera que en el futuro, una vez estén implementadas estas técnicas en el entrenamiento, tener mejores resultados de Scaled YOLOv4.

En cuanto a la detección hubiese sido mejor si se usa el umbral de detección como un hiperparámetro, es decir, probar con umbrales de detección desde 0,05 hasta 0,95, esperando probar y determinar cuál es la combinación que da mejores resultados. Sin embargo, para comparar con el trabajo principal realizado por Pérez et al. (2021) se fijó en 0,5 para hacer en lo posible trabajos "comparables". Además, si se calcula el promedio de AP para cada nivel de umbral dentro de los siguientes umbrales 0,5; 0,55; 0,6; 0,65; 0,7; 0,75; 0,8; 0,85; 0,9 y 0,95, tal como se ve en la Tabla 5.3 se obtienen resultados similares a solo evaluar en AP_0,5. Por tanto, para saber cuál es el mejor modelo es posible hacerlo con un umbral único.

Modelo	**AP_0,5:0,95**
YOLOv5	26 %
Scaled YOLOv4-p5	22 %
Scaled YOLOv4-p6	27 %
Scaled YOLOv4-p7	23 %

Tabla 5.3: Promedio de AP de cada modelo en la imagen de evaluación para distintos niveles de umbral comprendidos entre 0,5 a 0,95.

Finalmente, algunas detecciones realizadas al aplicar la presente metodología consideradas como "Falso positivo", podrían ser consideradas de tipo "Verdadero positivo", si en el proceso

de etiquetado, cada una de las etiquetas se hubiesen ajustado perfectamente al área cubierta por cada árbol. Como se ve en la Figura 5.6, hay algunas detecciones hechas que se ajustan demasiado al contorno del árbol como el caso a , mientras que su etiqueta es demasiado holgada, lo cual la intersección sobre la unión es menor al 50 % categorizándola como una predicción incorrecta cuando visualmente se evidencia de forma correcta. Otro caso similar como el expuesto en el caso b y caso c, ocurre cuando la etiqueta es un poco holgada respecto al contorno del árbol y la predicción es levemente menor al perímetro de este, lo cual, a pesar de que podría considerarse una detección correcta, al no superar el umbral de IoU, se descarta. Sin embargo, si la etiqueta se ajusta perfectamente al contorno de cada árbol, estos casos donde se subdimensiona el área predicha, tendrían mayor probabilidad de ser considerada como un "Verdadero positivo".

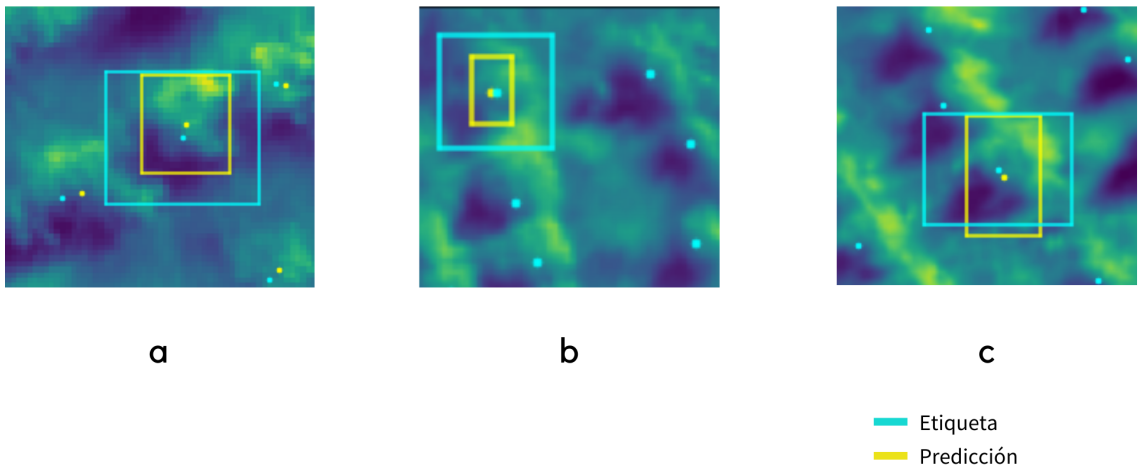


Figura 5.6: Detecciones incorrectas perfectibles.

En base a lo anteriormente explicado, se presenta una de las desventajas de utilizar esta metodología. El proceso de etiquetado es bastante costoso en tiempo y esfuerzo para lograr un volumen de datos apto para estos algoritmos de aprendizaje profundo supervisados y, que a su vez, cada etiqueta logre ajustarse lo más posible al contorno de cada uno de los árboles para tener mejores métricas del desempeño de cada modelo.

Capítulo 6

Conclusión

El presente trabajo realiza la tarea de conteo y detección de árboles dentro de un predio forestal en la Región del Biobío, utilizando algoritmos de aprendizaje profundo supervisados haciendo uso de imágenes multiespectrales georreferenciadas. Para el uso de este tipo de datos en los detectores de una etapa YOLOv5 y Scaled YOLOv4, la investigación se apoya de QGIS para la generación de etiquetas compatibles con este tipo de datos.

La metodología utilizada se basa principalmente en el trabajo realizado por Pérez et al. (2021), quienes utilizan el detector YOLOv3 para la detección y conteo de árboles. Los resultados obtenidos en el presente trabajo, comparando con el predio forestal de características similares al que se trabaja en dicha investigación, evidencia mejoras en los dos tipos de tarea logrando un 5 % de mejora en el conteo y un 12 % en la detección.

En cuanto a los resultados propios del presente trabajo, se concluye que YOLOv5 es el detector más rápido en tiempos de entrenamiento con un tiempo de 0,38 horas, en comparación a 0,85, 1,61 y 3,94 horas requeridas por los otros modelos. Además, obtiene resultados competentes en ambas tareas como un 71 % de $AP_{0,5}$ en la detección y un 17 % de diferencia en la tarea de conteo. Por otra parte, se evidencia las mejoras que tiene las técnicas de escalamientos usadas por Scaled YOLOv4 y búsqueda de hiperparámetros con validación cruzada logrando los mejores resultados en la detección con un 76 % en el $AP_{0,5}$ y un 0,1 % en el conteo.

Un aspecto interesante a investigar para trabajos futuros en el área, sería cuantificar el porcentaje que afecta en la mejora el uso de modelos distintos a YOLOv3 usado en el 2016 y el

uso de imágenes multiespectrales en vez de RGB.

En cuanto a líneas de trabajos que se esperan realizar en el futuro para profundizar esta investigación, en primer lugar, replicar las técnicas de aumentación de datos en Scaled YOLOv4 y que no son utilizados por compatibilidad con las bibliotecas y datos empleados. En la misma línea, se espera doblar el set de datos para determinar el efecto que tiene tanto en el entrenamiento como en los resultados al inferir sobre un set de datos de testeo.

Una línea de investigación más profunda en busca de mejoras es combinar la metodología presentada en este trabajo con la usada por Montenegro y Flores-Calero (2022), quienes utilizan dos redes en paralelos que analizan las bandas RGB e infrarrojo por separado para luego unirlas e inferir según las combinación de características de estos.

Finalmente, se espera seguir desarrollando y mejorando los resultados presentados en el presente informe, aprovechando la velocidad a la que avanza el campo de la visión computacional y las tecnologías de captura y manejo de datos. Con ello, se espera lograr un manejo de plantaciones forestales mucho más preciso, rápido y menos costoso en tiempo, esfuerzo y dinero.

Bibliografía

A. Abraham. Meta learning evolutionary artificial neural networks. *Neurocomputing*, 56:1–38, 2004. ISSN 0925-2312.

S. Albawi, T. A. Mohammed, y S. Al-Zawi. Understanding of a convolutional neural network. En *2017 International Conference on Engineering and Technology (ICET)*, páginas 1–6, 2017.

A. Ammar, A. Koubaa, y B. Benjdira. Deep-learning-based automated palm tree counting and geolocation in large farms from aerial geotagged images. *Agronomy*, 11(8), 2021.

ArcGIS. `https://pro.arcgis.com`.

A. Bochkovskiy, C.-Y. Wang, y H.-Y. M. Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

M. Cardemil. Industria forestal en Chile. 2021. URL `https://obtienearchivo.bcn.cl/obtienearchivo?id=repositorio/10221/32419/1/N_68_21_Industria_Forestal_en_Chile.pdf`.

M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez, y J. García-Gutiérrez. On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data. *Remote Sensing*, 13(1), 2021.

G. Chen y Y. Shang. Transformer for tree counting in aerial images. *Remote Sensing*, 14(3), 2022.

J. Dai, Y. Li, K. He, y J. Sun. R-fcn: Object detection via region-based fully convolutional networks, 2016.

N. Dalal y B. Triggs. Histograms of oriented gradients for human detection. En *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, páginas 886–893 vol. 1, 2005.

D. Falk y A. Campos. Algoritmo semiautomático para el conteo de árboles en plantaciones forestales mediante el uso de imágenes áereas. *Sociedad Argentina de Informática e Investigación Operativa*, páginas 186–194, 2014.

F. B. Fitch. Warren s. mcculloch and walter pitts. a logical calculus of the ideas immanent in nervous activity. bulletin of mathematical biophysics, vol. 5 (1943). *Journal of Symbolic Logic*, 9(2):49–50, 1944.

K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Neural Networks*, 1980.

F. A. Gougeon. A crown-following approach to the automatic delineation of individual tree crowns in high spatial resolution aerial images. *Canadian Journal of Remote Sensing*, 21(3): 274–284, 1995.

K. He, X. Zhang, S. Ren, y J. Sun. Deep residual learning for image recognition. En *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 770–778, 2016.

J. Hosang, R. Benenson, y B. Schiele. Learning non-maximum suppression. En *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

C. Hung, M. Bryson, y S. Sukkarieh. Multi-class predictive template for tree crown detection. *ISPRS Journal of Photogrammetry and Remote Sensing*, 68:170–183, 2012. ISSN 0924-2716.

L. Huo y E. Lindberg. Individual tree detection using template matching of multiple rasters derived from multispectral airborne laser scanning data. *International Journal of Remote Sensing*, 41(24):9525–9544, 2020.

Instituto Forestal. Estadísitca forestales. 2021. https://wef.infor.cl/sector$_f$*orestal*/*sectorforestal.php*.

Jacob Solawetz. Yolov5 new version - improvements and evaluation, 2020.

H. Jain y S. Nandy. Incremental training for image classification of unseen objects. *Summer Research Fellowship Programme of India's Science Academies*, 08 2019.

Z. Ji, J. Li, y M. Telgarsky. Early-stopped neural networks are consistent. En M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, y J. W. Vaughan, editores, *Advances in Neural Information Processing Systems*, volume 34, páginas 1805–1817. Curran Associates, Inc., 2021.

T. Jintasuttisak, E. Edirisinghe, y A. Elbattay. Deep neural network based date palm tree detection in drone imagery. *Computers and Electronics in Agriculture*, 192:106560, 2022.

Y. Ke y L. Quackenbush. A review of methods for automatic individual tree-crown detection and delineation from passive remote sensing. *International Journal of Remote Sensing*, 32: 4725–4747, 09 2011a.

Y. Ke y L. Quackenbush. A comparison of three methods for automatic tree crown detection and delineation from high spatial resolution imagery. *International Journal of Remote Sensing*, 32: 3625–3647, 07 2011b.

D. Koc-San, S. Selim, N. Aslan, y B. T. San. Automatic citrus tree extraction from uav images and digital surface models using circular hough transform. *Computers and Electronics in Agriculture*, 150:289–301, 2018.

A. Krogh. What are artificial neural networks? *Nature Biotechnology*, 26, 02 2008.

Y. Le Cun, L. Jackel, B. Boser, J. Denker, H. Graf, I. Guyon, D. Henderson, R. Howard, y W. Hubbard. Handwritten digit recognition: applications of neural network chips and automatic learning. *IEEE Communications Magazine*, 27(11):41–46, 1989.

D. Leckie, F. Gougeon, D. Hill, R. Quinn, L. Armstrong, y R. Shreenan. Combined high-density lidar and multispectral imagery for individual tree crown analysis. *Canadian Journal of Remote Sensing*, 29:633–649, 10 2003.

S. Liu, L. Qi, H. Qin, J. Shi, y J. Jia. Path aggregation network for instance segmentation, 2018. URL `https://arxiv.org/abs/1803.01534`.

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, y A. C. Berg. SSD: Single shot MultiBox detector. En *Computer Vision – ECCV 2016*, páginas 21–37. Springer International Publishing, 2016.

D. Misra. Mish: A self regularized non-monotonic activation function, 2019.

B. Montenegro y M. Flores-Calero. Detección de peatones en el día y en la noche usando YOLO-v5. *Ingenius. Revista de Ciencia y Tecnología*, páginas 85 – 95, 06 2022.

Nelson y Solawetz. Blog roboflow, 2020a. https://blog.roboflow.com/yolov5-is-here/.

Nelson y Solawetz. Blog roboflow, 2020b. https://blog.roboflow.com/yolov4-versus-yolov5/.

R. Pollock. *The automatic recognition of individual trees in aerial images of forests based on a synthetic tree crown image model*. PhD tesis, University of British Columbia, 1996.

D. Pouliot, D. King, F. Bell, y D. Pitt. Automated tree crown detection and delineation in high-resolution digital camera imagery of coniferous forest regeneration. *Remote Sensing of Environment*, 82(2):322–334, 2002.

D. Pulido, J. Salas, M. Rös, K. Puettmann, y S. Karaman. Assessment of tree detection methods in multispectral aerial images. *Remote Sensing*, 12(15), 2020.

M. Pérez, B. Karelovic, R. Molina, R. Saavedra, P. Cerulo, y G. Cabrera. Precision silviculture: Use of uavs and comparison of deep learning models for the identification and segmentation of tree crowns in pine crops. 2021.

QGIS Development Team. *QGIS Geographic Information System*. QGIS Association.

J. Redmon y A. Farhadi. Yolo9000: Better, faster, stronger, 2016.

J. Redmon y A. Farhadi. Yolov3: An incremental improvement, 2018.

J. Redmon, S. Divvala, R. Girshick, y A. Farhadi. You only look once: Unified, real-time object detection, 2016.

S. Ren, K. He, R. Girshick, y J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015.

H. Robbins y S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, páginas 400–407, 1951.

R. Saravanan y P. Sujatha. A state of art techniques on machine learning algorithms: A perspective of supervised learning approaches in data classification. En *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, páginas 945–949, 2018.

K. Simonyan y A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

M. Tan, R. Pang, y Q. V. Le. Efficientdet: Scalable and efficient object detection. 2019.

C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, y J.-W. Hsieh. Cspnet: A new backbone that can enhance learning capability of cnn, 2019a.

C.-Y. Wang, A. Bochkovskiy, y H.-Y. M. Liao. Scaled-yolov4: Scaling cross stage partial network, 2020a.

Y. Wang, X. Zhu, y B. Wu. Automatic detection of individual oil palm trees from uav images using hog features and an svm classifier. *International Journal of Remote Sensing*, 40(19):7356–7370, 2019b.

Y. Wang, Y. Li, Y. Song, y X. Rong. The influence of the activation function in a convolution neural network model of facial expression recognition. *Applied Sciences*, 10(5), 2020b.

M. Wulder, K. Niemann, y D. G. Goodenough. Local maximum filtering for the extraction of tree locations and basal area from high spatial resolution imagery. *Remote Sensing of Environment*, 73(1):103–114, 2000.

Z. Yanling, D. Bimin, y W. Zhanrong. Analysis and study of perceptron to solve xor problem. En *The 2nd International Workshop on Autonomous Decentralized System, 2002*, páginas 168–173, 2002.

K. Yarak, A. Witayangkurn, K. Kritiyutanont, C. Arunplod, y R. Shibasaki. Oil palm tree detection and health classification on high-resolution imagery using deep learning. *Agriculture*, 11:183, 02 2021.

S. Zagoruyko y N. Komodakis. Wide residual networks, 2016.

Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, y D. Ren. Distance-iou loss: Faster and better learning for bounding box regression. volume 34, páginas 12993–13000, 02 2020.

O. Özyeşil, V. Voroninski, R. Basri, y A. Singer. A survey of structure from motion. *Acta Numerica*, 26:305–364, 2017.

Anexo

A.1. Resumen FI

UNIVERSIDAD DE CONCEPCIÓN – FACULTAD DE INGENIERÍA
RESUMEN DE MEMORIA DE TÍTULO

Departamento: Departamento de Ingeniería Industrial

Carrera: Ingeniería Civil Industrial

Nombre del memorista: Pablo Ignacio Meza Sparza

Título de la memoria: Localización y conteo de árboles en áreas de interés forestal mediante imágenes aéreas multiespectrales georreferenciadas

Fecha de la presentación oral:

Profesor(es) Guía: Guillermo Cabrera Vives, Ph.D.

Profesor(es) Revisor(es):

Concepto:

Calificación:

Resumen:

La presente memoria de título aborda la problemática de la gestión de recursos forestales cuando éstos están en la etapa de plantación forestal. Para abordarlo, generalmente implica un gran gasto en recursos de tiempo y dinero al utilizar metodologías manuales. Por ello, este estudio propone implementar, evaluar y comparar distintos algoritmos de aprendizaje profundo para localización y conteo de árboles en plantaciones forestales a travées de imágenes aéreas georreferenciadas. Los resultados evidencian mejoras en comparación al detector del

estudio en el que se basa la presente pesquisa. Además, se observa que de los modelos YOLOv5 y Scaled YOLOv4 propuestos, el modelo que presenta mejores métricas es Scaled YOLOv4-p6. Sin embargo, YOLOv5 presenta resultados levemente inferiores con un tiempo de entrenamiento e inferencia considerablemente menor.