Instituto Tecnológico de Costa Rica

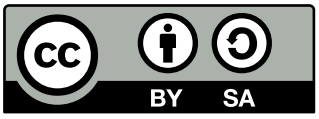
Escuela de Ingeniería en Electrónica

Seguimiento de plantaciones de café a través de fotogrametría UAS y técnicas de aprendizaje profundo

para optar por el título de maestría en electrónica con énfasis en procesamiento digital de señales

Santiago Andrés López Rojas

Cartago Agosto, 2022



Resumen

El documento presenta una estrategia de detección y conteo de cafetos en ortomosaicos RGB tomados con drones a baja altura. El trabajo se realizó en el laboratorio de fotogrametría del Instituto Tecnológico de Costa Rica (ITCR), en colaboración del Instituto del Café de Costa Rica (Icafe). La red neuronal propuesta se basa en la arquitectura YOLO, y los resultados son comparados con el estado del arte de detección YOLOv4. Se describe el proceso de confección del conjunto de datos y el preprocesamiento al que se somete, donde todas las imágenes poseen una resolución espacial de alrededor de 2cm/píxel. Se obtiene como resultado del entrenamiento una detección del 92 %, que hace referencia a la exhaustividad para un IoU de 0.5 y se alcanza un mAP@50 de 83.18 %.

Palabras clave: Red neuronal convolucional, detección, YOLO, mAP, exhaustividad, IoU, fotogrametría.

Abstract

The document presents a strategy for the detection and counting of coffee trees in RGB orthomosaics taken with drones at low altitude. The work was carried out in the photogrammetry laboratory of the Technological Institute of Costa Rica (ITCR), in collaboration with the Coffee Institute of Costa Rica (Icafe). The neural network proposal is based on the YOLO architecture, and the results are compared with the state of the art of YOLOv4 detection. The process of making the data set and the preprocessing to which it is subjected is described, where all the images have a spatial resolution of around 2cm/pixel. As a result of the training, a detection of 92 % is obtained, which refers to the recall for an IoU of 0.5 and a mAP@50 of 83.18 %.

Key words: Convolutional neuronal networks, detection, YOLO, mAP, recall, IoU, photogrammetry.

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Maestría Académica en Electrónica
Trabajo Final de Graduación
Tribunal Evaluador
Acta de Aprobación de Tesis

Defensa del Trabajo Final de Graduación
Requisito para optar por el título de Máster en Ingeniería Electrónica
Grado Académico de Magister Scientiae

El Tribunal Evaluador aprueba la defensa del Trabajo Final de Graduación denominado **"Seguimiento de plantaciones de café a través de fotogrametría UAS y técnicas de aprendizaje profundo",** realizado por **Santiago López Rojas** Carné: **2014098356**, y hace constar que cumple con las normas establecidas por la Unidad Interna de Posgrados de la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal Evaluador

JOSE PABLO ALVARADO MOYA (FIRMA)
PERSONA FISICA, CPF-01-0753-0713.
Fecha declarada: 10/11/2022 06:23:28 PM
Razón: Acta de Aprobación Santiago López
Lugar: Cartago

Dr. Pablo Alvarado Moya
Profesor Lector

SERGIO ALBERTO ARRIOLA VALVERDE (FIRMA)
Firmado digitalmente por SERGIO ALBERTO ARRIOLA VALVERDE (FIRMA)
Fecha: 2022.11.10 11:06:57 -06'00'

M.Sc. Sergio Arriola Valverde
Profesor Lector

M.Sc. Michael Grüner Monzón
Evaluador Independiente

RENATO RIMOLO DONADIO (FIRMA)
Firmado digitalmente por RENATO RIMOLO DONADIO (FIRMA)
Fecha: 2022.11.10 22:31:03 -06'00'

Dr. Renato Rímolo Donadío
Director del Tesis

Cartago, 31 de agosto del 2022

Índice general

Índice de figuras

Índice de tablas

Lista de símbolos y abreviaciones

Abreviaciones

YOLO	*You Only Look Once*
ICAFE	Instituto del Café de Costa Rica
ITCR	Instituto Técnologico de Costa Rica
mAP	*Mean Average Precision*
IoU	*Intersection over Union*
CNN	*Convolutional Neuronal Network*
ANN	*Artifitial Neuronal Network*

Notación General

$\mathbf{A}$	*Matrices*	$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}$
$\underline{x}$	*Vector*	$\underline{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

Capítulo 1

Introducción

Costa Rica es conocido a nivel mundial como un país cafetalero, que exporta café desde 1820 [1], con una producción anual promedio de 2.1 millones de sacos (60kg por saco); en el año 2018 reportó cerca de 84 mil hectáreas dedicadas exclusivamente a la plantación de café [2], mientras que para el 2020 el número aumentó a 93 mil hectáreas, que equivale aproximadamente al 1.8 % del territorio nacional [1] y se registra que más de 40 mil personas son pequeños productores cafetaleros.

Actualmente, en las plantaciones el conteo y monitoreo de cafetos, en búsqueda de enfermedades, se realiza de forma manual. Esto involucra un coste de mano de obra especializada y no especializada, así como la exposición del proceso al error humano; además, la intervención directa en las plantas para recolección de muestras y su posterior envío a laboratorios.

Con esto nace la iniciativa del Laboratorio de Fotogrametría (UASTEC) por realizar análisis en cultivos empleando sistemas UAS (*Unmanned Aerial Systems*) y herramientas fotogramétricas. El objetivo es realizar a partir de imágenes una detección de los cafetos en las plantaciones y obtener información de estos, como el vigor o la presencia de enfermedades.

Para este proyecto se trabajó en conjunto con el Instituto del Café de Costa Rica (*ICAFE*) [3] para realizar investigaciones en algunas de sus instalaciones. Esta institución prestó en favor de este proyecto diferentes parcelas ubicadas en Barva de Heredia. Las demás parcelas provenientes de la Zona de los Santos, Naranjo y Grecia corresponden a producciones independientes. De estas plantaciones se extraen las imágenes usadas para el proceso de entrenamiento, prueba y validación de las redes neuronales artificiales (*ANN*, por sus siglas en inglés).

Algunos trabajos previos relacionados tuvieron como objetivo el conteo de árboles en imágenes aéreas, mediante segmentación y filtros, con lo que se obtuvo cálculos de biomasa en el área; estás investigaciones reportaron resultados de detección superiores al 85 %, aunque se destacó la dificultad de extrapolar a diferentes plantaciones [4]. Posteriormente, los trabajos [5] y [6] se centraron en desarrollar un proceso de detección

que facilite esta portabilidad de una plantación a otra, enfocados en la detección de plantas de café y banano, respectivamente.

De este modo [5] implementó la arquitectura YOLOv3 para la detección de cafetos. Esta detección se realizó en imágenes con una resolución espacial de 1cm/píxel y 4cm/píxel que alcanzó una detección del 35 % y 90 % de los cafetos, respectivamente, para un IoU de 0.35. Mientras que para un IoU de 0.50 la detección bajó a un 21 % y 78 % (para cada una de las resoluciones espaciales antes mencionadas). Se destaca que para las imágenes con resolución espacial de 1cm/píxel la detección no supera el 50 % lo que corresponde con una detección aleatoria.

Los objetivos de este proyecto consisten en solucionar las deficiencias encontradas en [5], planteando un modelo de red neuronal enfocado en detección de cafetos para lograr un mejor porcentaje de detección para un IoU de 0.5 o superior y empleando imágenes con resolución espacial en el orden de 1cm/píxel.

1.1. Planteamiento de la solución

Como solución se plantea confeccionar una arquitectura de red neuronal basada en el sistema *you only look once* (YOLO), aplicado a la detección de los cafetos. Siguiendo el diagrama de entradas/salidas de la figura 1.1, la solución se divide en cuatro bloques. El primero consiste en la generación de la conjunto de datos etiquetados. El segundo divide la conjunto de datos entre set de entrenamiento y prueba. La tercera etapa es el entrenamiento de diferentes redes neuronales. Finalmente, en la cuarta etapa se tiene la propuesta e implementación de una arquitectura orientada a detección de cafetos. Las entradas y salidas de estos bloques son:

- **Mapas:** Hacen referencia a los ortomosaicos generados en base las imágenes aéreas tomadas por los drones. Se emplearán en la investigación únicamente mapas RGB.
- **Bloques de entrenamiento y prueba:** Luego de realizar un etiquetado, preprocesamiento y aumentado de datos de las imágenes, se divide la conjunto de datos entre bloques de entrenamiento y prueba. Este apartado también entrega información de las características imágenes en uso así como de las etiquetas.
- **Resultados comparativos:** Estos son los resultados de diferentes arquitecturas de redes neuronales que son entrenadas para detectar cafetos (gráficas de precisión, exhaustividad, mAP@50, funciones de pérdidas, etc). La intención es encontrar arquitectura con un buen desempeño y mejorarla.
- **Modelo de detección:** Este es el resultado final del proyecto. Consiste en un modelo propuesto, tomando como base la mejor arquitectura de los resultados comparativos a la que se le aplican mejoras considerando los resultados del preprocesamiento de la conjunto de datos.

1.2. Objetivos y estructura del documento

El objetivo general del proyecto es proponer una metodología de detección de cafetos, incluyendo el proceso de toma de imágenes, el etiquetado, preprocesamiento de imágenes, división entre bloques de entrenamiento y validación. Para esto se buscó una arquitectura de red neuronal capaz de realizar una detección de los cafetos en una plantación usando imágenes aéreas a baja altura y mejorar su rendimiento con una arquitectura propuesta.

Los objetivos específicos incluyen: Confeccionar y analizar un conjunto de datos para el entrenamiento y validación, implementar cinco arquitecturas de red neuronal para la detección de cafetos, seleccionar la arquitectura con los mejores resultados de detección y modificarla para mejorar su rendimiento (exhaustividad), y finalmente realizar la experimentación necesaria para verificar que no haya sobreajuste, ni sesgo.

El documento se encuentra organizado en 5 capítulos. El **Capítulo 2** corresponde al *Marco Teórico*, donde se detallan conceptos y fundamentos técnicos, incluyendo la descripción de las métricas para evaluar los resultados. El **Capítulo 3** incorpora la *Solución* del problema, detallando lo propuesto en la figura 1.1. Por su parte, el **Capítulo 4** presenta los resultados obtenidos en cada apartado. Finalmente, el **Capítulo 5** describe las conclusiones y recomendaciones para futuros proyectos.

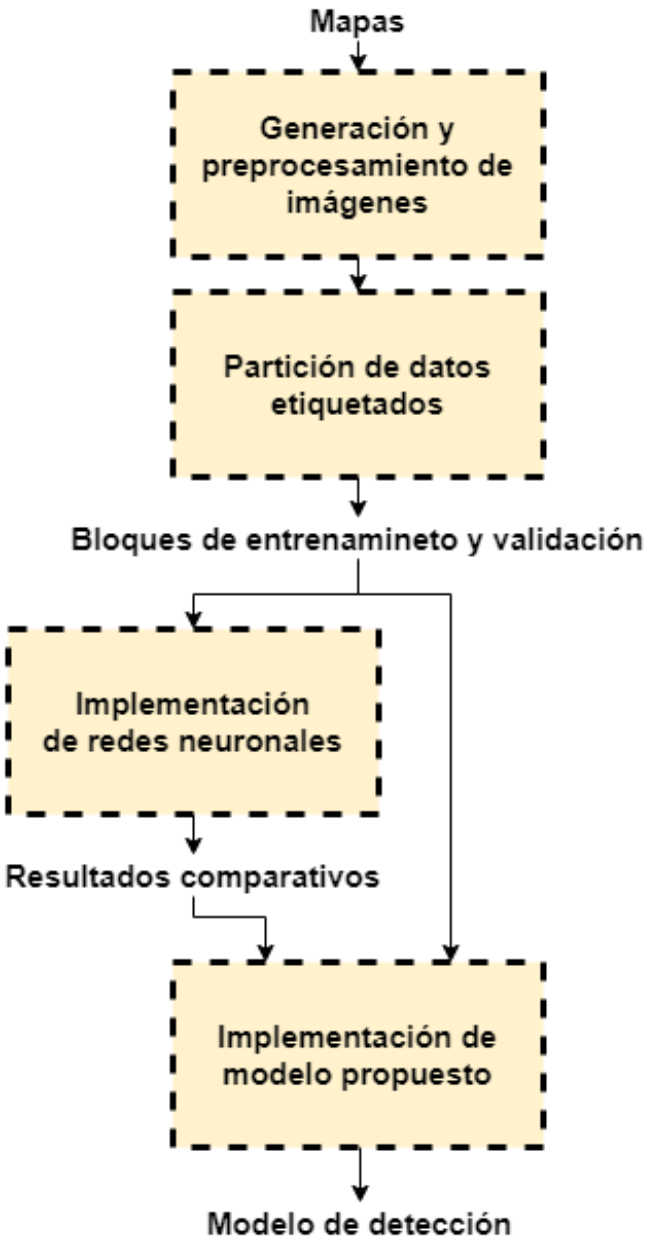


Figura 1.1: Diagrama de entradas/salidas de la metodología seguida para encontrar el modelo óptimo para detección de cafetos.

Capítulo 2

Marco teórico

En esta sección se exponen algunos de los conceptos necesarios para comprender la solución planteada, los resultados y la discusión de los mismos. Se hace una breve introducción a inteligencia artificial, redes neuronales artificiales, redes neuronales convolucionales y detección de objetos. Además, se hace una breve mención de conceptos como fotorgramétría y otros relacionados a la recolección de datos.

2.1. El cultivo del café en Costa Rica

Aunque actualmente el café crece alrededor del mundo, se sabe que estas plantas son originarias de los bosques de café en Etiopía. Una leyenda autóctona cuenta que el primero en descubrir el potencial del café fue un pastor de cabras, que al notar como sus cabras se volvían enérgicas y no conciliaban el sueño por las noches después de comer el fruto de cierto árbol que crecía en los alrededores. Este joven le informó de este descubrimiento al monasterio local y los monjes crearon una bebida a base de estas bayas, que les ayudaba a mantenerse despiertos, dando así origen al consumo del café como lo conocemos hoy [7].

Los cafetos, como su nombre lo sugiere, son los árboles de los que se obtienen las semillas de café, Estas plantas son normalmente pequeñas, aunque dependiendo de la variedad pueden llegar a medir hasta 9 metros. Cada espécimen es cubierto con hojas verdes, que crecen en pares. Los frutos crecen a lo largo de las ramas, iniciando con un color verde y tornándose rojas al madurar. Un cafeto tarda alrededor de un año para alcanzar su estado de maduración y dar frutos por primera vez. Finalmente, la esperanza de vida de estos árboles oscila alrededor de los 100 años, aunque su pico de producción se encuentra entre los 5 y los 25 años [8].

Todo el café comercial del mundo es proveniente de regiones que se encuentran entre las latitudes 25° norte y 30° sur, que incluye 50 países que presentan suelos apropiados, climas con temperaturas tropicales, lluvias frecuentes y sol moderado, siendo condi-

ciones ideales para el apropiado crecimiento de la planta. Este conjunto de lugares es también conocido como *the Coffee Belt* [8].

Para el caso particular de Costa Rica, acorde al artículo "Historia del café en Costa Rica", el grano fue introducido en el país cerca de 1720 con la especie *coffea Arábica*, variedad *Typica*. Esta se modificó casi un siglo más tarde el paradigma agrícola de la época, el cual consistía en agricultura de subsistencia; colocando posteriormente a Costa Rica como el primer país centroamericano en establecer la industria cafetalera [9].

Como se menciona, Costa Rica cuenta con las condiciones ideales para cultivar café y la primera siembra de la que se tiene registro se remonta a 1816, siendo en esa misma década cuando los gobiernos municipales incentivaron el cultivo cafetalero a la población, haciendo entrega de plantas y dando en concesión tierras a agricultores pequeños. La primera exportación de café se efectúa en 1820 y en 1840 se establece como único producto de exportación constante [9]. Actualmente en el país la comercialización del café es totalmente del sector privado, con supervisión del Estado. El Instituto del Café de Costa Rica (ICafe) es quien desempeña esta supervisión [10].

Estos sectores corresponden a los productores, quienes tienen derecho a explotar plantaciones cafetaleras, y entregan las cerezas del café a los beneficiarios quienes además de recibir, también venden el fruto y lo procesan para obtener el grano. Estos entes también son responsables de financiar a los productores. Los Exportadores son el vínculo con el extranjero; suministran los granos del café a compañías importadoras y/o tostadoras que operan en los países consumidores. Finalmente, los Torrefactores son los propietarios de establecimientos de tostado, molido o cualquier otro proceso industrial del grano del café; también dedicándose a comercializar a nivel nacional [10].

El presente trabajo cuenta con el apoyo del Instituto del Café de Costa Rica, teniendo como objetivo la innovación en procesos durante el proceso de cultivo del café. La detección mediante imágenes aéreas de cafetos facilita el conteo en extensiones muy grandes de terreno, y como consecuencia ayuda durante el proceso de planificación de producción. Otro uso directo de este proyecto es el identificar en qué áreas se está realizando siembra de café para efectos del inventario nacional.

En futuros proyectos lo que se pretende es el poder determinar características vitales de los cafetos usando imágenes de otros espectros de luz. Esta tarea se facilita una vez que ya se sepa la ubicación de cada cafeto.

2.2. Introducción a redes neuronales artificiales

Desde años muy tempranos, los investigadores han soñado con la máquina capaz de pensar y razonar, llegando a simular el comportamiento humano. Este ideal fue reforzado con el nacimiento de las computadoras programables, dando nacimiento a lo que hoy conocemos como la *inteligencia artificial*, a la que se le ha dado en últimos años aplicaciones prácticas, con la intención de automatizar labores rutinarias.

La inteligencia artificial ha sido usada en un inicio para la solución de problemas matemáticos de alta complejidad. Aunque el verdadero reto para esta ciencia radica en las tareas sencillas, acciones que los seres humanos efectúan de manera automática, pero que a su vez resultan difíciles de explicar en detalle; como las acciones generadas por intuición, el reconocimiento de rostros o la interpretación de gestos [11].

El desarrollo de redes neuronales artificiales inicia en 1960, no obstante, no fueron usadas en mayor número de tareas debido a la falta de un método de entrenamiento que hiciera que estas superaran a los métodos tradicionales. No es hasta 1986 gracias al trabajo de David Rumelhart, Geoffrey Hinton y Ronald Williams, quienes establecen un método de aprendizaje eficiente [12], y luego en 2006 con el surgimiento de las técnicas de aprendizaje profundo (*deep learning*), que las redes neuronales toman mayor relevancia para la solución de problemas en procesamiento digital de imágenes y visión por computador[11].

El presente proyecto centra su atención en el reconocimiento de objetos en imágenes, específicamente de cafetos. Para realizar esta detección se emplearon algunos de los métodos pertenecientes a los algoritmos de aprendizaje profundo (*deep learning*). Estos algoritmos de modo conceptual, le dan al computador la capacidad de *aprender* y *entender* el problema reduciendo el mismo a una jerarquía de conceptos más sencillos de analizar [13].

Las *redes neuronales artificiales* son un modelo computacional que incorpora algunos de estos métodos de aprendizaje profundo para cumplir tareas como clasificación y localización de objetos en imágenes. Estos modelos son los empleados en el proyecto para automatizar el proceso de detección y conteo de plantas de café en una plantación.

Para dar solución a cualquier problema con el paradigma tradicional de programación, es necesario definirle al computador las acciones específicas a realizar, dividiendo grandes tareas en subtareas que la máquina puede entender y realizar con facilidad. Mientras más compleja sea la tarea mayor cantidad de divisiones serán necesarias.

En contraste, las redes neuronales artificiales rompen este esquema no definiendo explícitamente el contenido o tareas específicas de estas divisiones y delegando esa tarea a la propia red. En este proceso la red modifica de manera iterativa sus valores internos en función de información de problemas resueltos (datos de *entrenamiento*) de modo que la red pueda generalizar de manera autónoma el problema y para el momento en que se le dé una muestra nueva que no haya visto con anterioridad, pueda dar un resultado correcto.

Por ejemplo, si se desea que una red neuronal clasifique imágenes de cafetos se le tendrá que dar una serie de imágenes etiquetadas como cafetos para que esta pueda modificar sus parámetros internos y posteriormente cuando se le de un set de imágenes con diferentes especies de árboles, pueda discernir entre los que corresponden a cafetos y los que no.

2.3. Redes neuronales artificiales convolucionales

En el ámbito de visión por computador, se llegó al punto en que no basta con clasificar diferentes imágenes en categorías, y también se requiere la información de la ubicación relativa de los objetos contenidos en la imagen. Esta tarea se conoce como detección de objetos [14]. En este contexto las redes neuronales convolucionales han tomado popularidad desde llegada de la R-CNN, introducida en la competencia de PASCAL VOC [15].

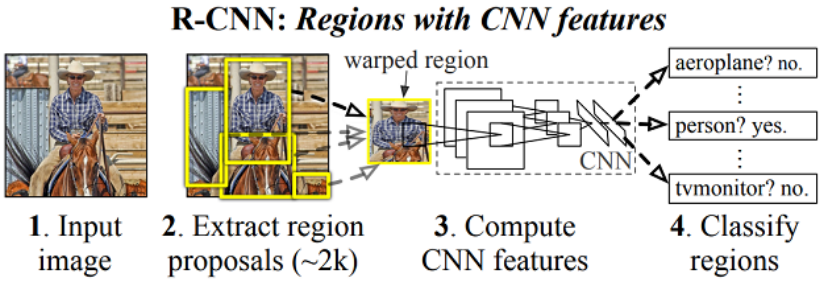


Figura 2.1: Flujo de trabajo de la R-CNN.
(tomados de [16]).

Este suceso cambió el paradigma de la detección, pasando de los métodos tradicionales donde se tenía entendimiento absoluto de todos los procesos a los cuales se sometían las imágenes, a un método donde el algoritmo hace una abstracción sobre la cual no se posee control directo [17]. La arquitectura de R-CNN esta compuesta por una serie de capas convolucionales y capas completamente conectadas, pero de manera conceptual se divide en tres etapas: región informativa, extracción de características y clasificación.

La región informativa es la encargada de escanear toda la imagen en búsqueda de objetos, generando ventanas de diferentes tamaños donde posiblemente se encuentren los objetos. Esta estrategia averigua todas las posiciones posibles de los objetos. Entre los retos en esta parte se encuentran la generación de ventanas redundantes y también el costo computacional que supone.

La extracción de características se emplea para reconocer diferentes objetos. Algunos ejemplos de extractores de características relativas son los algoritmos *Scale-invariant feature transform* (SIFT), *Speeded-Up robust features* (SURF) y *Oriented fast and rotated brief* (ORB). En este apartado, la diversidad de apariencias, condiciones de iluminación y fondos, son los factores que dificultan o entorpecen el proceso de extracción de características.

Finalmente, la sección de clasificación distingue entre un objeto detectado y otro dependiendo de las clases establecidas. Algunos ejemplos de algoritmos de clasificación son el *Support Vector Machine* (SVM), AdaBoost, *The k-nearest neighbors* (kNN) y *Dimension-based Partitioning and Merging* (DPM).

Siguiendo con esta división conceptual, se han propuesto modelos como el *Fast R-CNN* [18], el *Faster R-CNN*, entre otros; hasta llegar a la aproximación de *you only look once* [19], que en lugar de verlo como un proceso de localización y clasificación, ve el problema de la detección como una regresión, integrando todas las etapas de detección en una única arquitectura.

El objetivo es tener una única red neuronal que prediga los recuadros delimitadores (*bounding boxes*) y las probabilidades de clase directamente de la imagen completa en una sola evaluación. De este comportamiento deriva el nombre *You Only Look Once*. Uno de los beneficios que se busca es que gracias a que toda la detección ocurre en una sola red neuronal, se puede hacer una optimización de extremo a extremo (*end-to-end*).

El principal beneficio recae en la velocidad de predicción. Desde el primer lanzamiento de YOLO se lograron realizar detecciones a tiempo real con una taza de 45 imágenes por segundo, tasa que ha ido mejorando conforme se han lanzados las siguientes versiones y se han solucionado problemas como la alta tasa de falsos negativos.

A grandes rasgos, el flujo de ejecución de YOLO consiste en primero hacer un escalado de la imagen de entrada a un tamaño específico. Luego se ejecuta una única CNN en la imagen escalada, para finalmente filtrar los resultados basados en un valor de confianza asignado por la misma red a las posibles detecciones (figura 2.2).

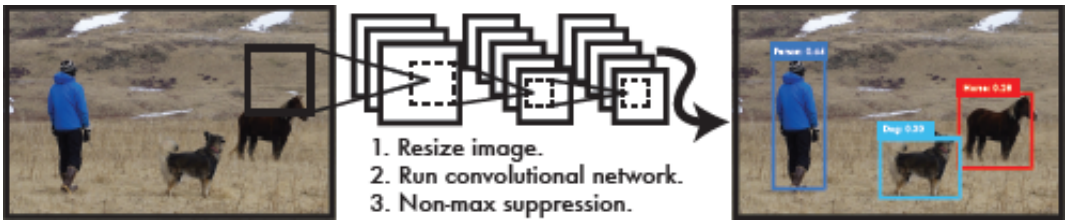


Figura 2.2: Flujo de ejecución básico de YOLO
(tomado de [19]).

Además de la ya mencionada velocidad de entrenamiento y predicción, YOLO no requiere de un pipeline complicado. Para las pruebas de detección simplemente se corre la red neuronal sobre una nueva imagen. Teniendo en cuenta que para el entrenamiento la red ve toda la imagen y no solo fragmentos de ella, la red puede generalizar mejor la información contextual de las imágenes. Además, aprende representaciones generales de objetos. Por lo que en comparación con otra ANN las arquitecturas YOLO se comporta de mejor manera al presentar escenarios o objetos en entornos nuevos.

Tomando esto en cuenta, el presente proyecto esta basado en la aproximación de YOLO y esta sección se encarga de explicar todos los tipos de capas usados en la arquitectura propuesta para la detección de cafetos [20].

2.3.1. Capas Convolucionales

Como su nombre lo indica, las capas convolucionales llevan a cabo convoluciones entre un filtro o *kernel* y la imagen de entrada, dando como salida lo que se denomina como mapa de características. Esta aproximación a diferencia de las capas completamente conectadas no poseen pesos asociados con cada neurona de la capa, sino que se definen un kernel de dimensiones especificas, y en lugar de tomar forma vectorial, posee forma matricial [21].

Otra ventaja que presentan estas capas es que toman en cuenta las proporciones de la imagen, manteniendo así la integridad del dato de entrada no colocándolo en un vector. En la figura 2.3 se presenta la interacción entre el kernel y la imagen de entrada (capa anterior).

A grandes rasgos el proceso consiste en multiplicar los píxeles del kernel con los píxeles del fragmento de imagen de entada. Luego se suman estos resultados y se aplica la función de activación. Este proceso se repite pasando al siguiente fragmento de la imagen. Este nuevo fragmento puede o no estar traslapado con el anterior.

Este proceso se representa de manera matemática siguiendo:

$$Y = X * f + b \tag{2.1}$$

donde $Y \in \mathbb{R}^{h' \times w' \times n}$ son los mapas de características de salida generados con n, h' y w' correspondiendo al número, la altura y el ancho de estos mapas de características, $X \in \mathbb{R}^{c \times h \times w}$ es la entrada de la capa, con c, h y w siendo número de canales, la altura y el ancho de la entrada. El operador $*$ representa la convolución, b es el término de sesgo y finalmente $f \in \mathbb{R}^{c \times k \times k \times n}$ corresponde a los filtros de la capa donde $k \times k$ es el tamaño del kernel y n representa la cantidad de estos [21].

Mientras más capas convolución tenga el modelo, mayor cantidad de características complejas será capaz de determinar en la imagen; sin embargo, esto a su vez incrementa el tiempo de entrenamiento del modelo así como la posibilidad de que el mismo caiga en sobreajuste (*overfitting*).

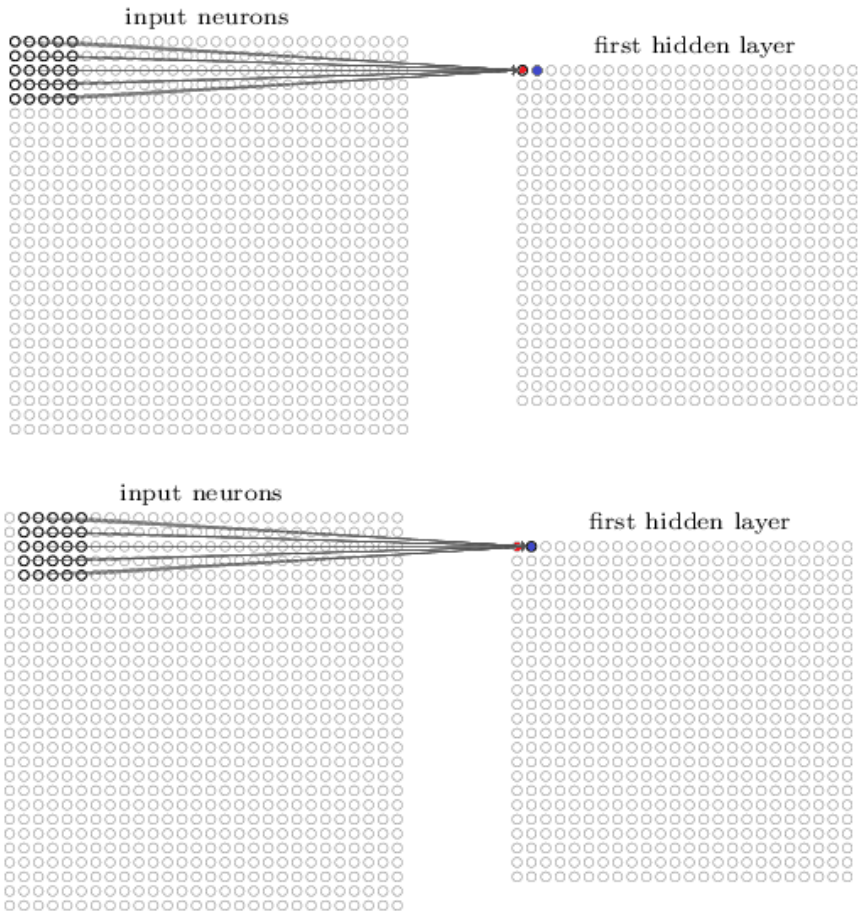


Figura 2.3: Funcionamiento de los campos receptivos locales (tomado de [21]).

2.3.2. Capas de agrupación

Conocidas mayormente por su nombre en inglés *Pooling layers*, estas capas se encargan en reducir el costo computacional de los modelos, además de ayudar a evitar el sobreajuste reduciendo las dimensiones de la capa de entrada. En una capa de agrupación, al igual que en la capa de convolución se tiene un filtro de dimensiones $n \times n$ que realizara una operación sobre las diferentes áreas de la imagen de entrada, moviéndose con un paso (*stride*) de s píxeles [21].

No existe literatura que especifique una frecuencia prudente para colocar capas de agrupamiento por lo que queda a discreción del desarrollador escoger en qué puntos es óptimo colocarlas; no obstante, el modelo de visión por computadora VGG-16 usa entre 2 o 3 capas de convolución entre cada capa de agrupamiento, mientras que VGG-19 usa hasta 4.

Existen varios tipos de capas de agrupación, entre los que se destacan:

- Agrupación máxima (*max pooling*): selecciona el máximo valor de la salida de la convolución y lo usa como salida.
- Agrupación mínima (*min pooling*): selecciona el mínimo valor de la salida de la convolución y lo usa como salida.
- Agrupación promedio (*averge pooling*): saca el promedio valor de la salida de la convolución y lo usa como salida.

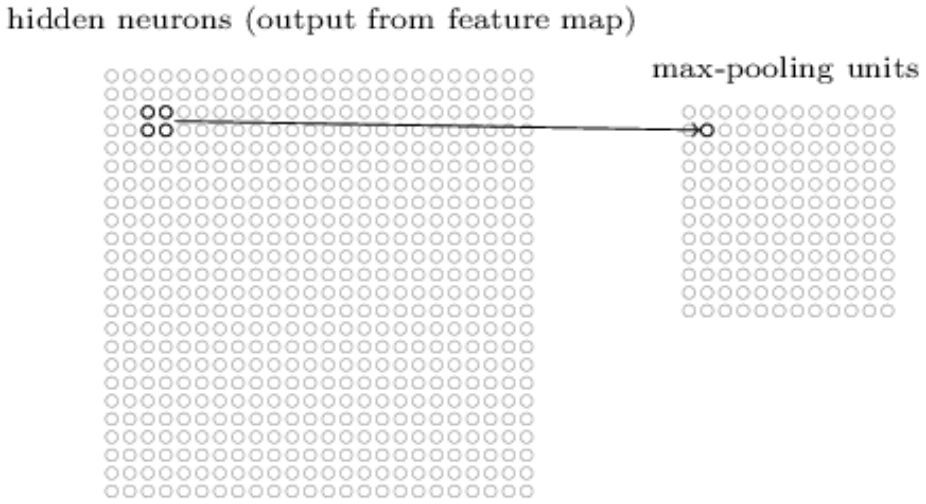


Figura 2.4: Funcionamiento de los *Pooling layers*
(tomado de [21]).

2.3.3. Capas convolucionales fantasmas

La idea de estas capas es reducir la cantidad de memoria que se usa durante el proceso de entrenamiento. Esto permite entrenar una mayor cantidad de capas sin la necesidad de tener tanta capacidad computacional. Otra ventaja de este módulo es que funciona igual que una capa convolucional que se describió anteriormente, por lo que no es necesario hacer mayor modificaciones para implementarlas en una arquitectura sustituyendo a las capas convolucionales de la misma.

Siguiendo con (2.1), el número de parámetros a optimizar está directamente relacionado con las dimensiones de entrada y salida. Como se ejemplifica en la figura 2.5, la salida de mapas de características convolucionales normalmente es muy redundante. Con esto en mente, la idea de las capas convolucionales fantasmas es evitar generar estas características redundantes una a una empleando grandes cantidades de operaciones de punto flotante (FLOP) y parámetros.

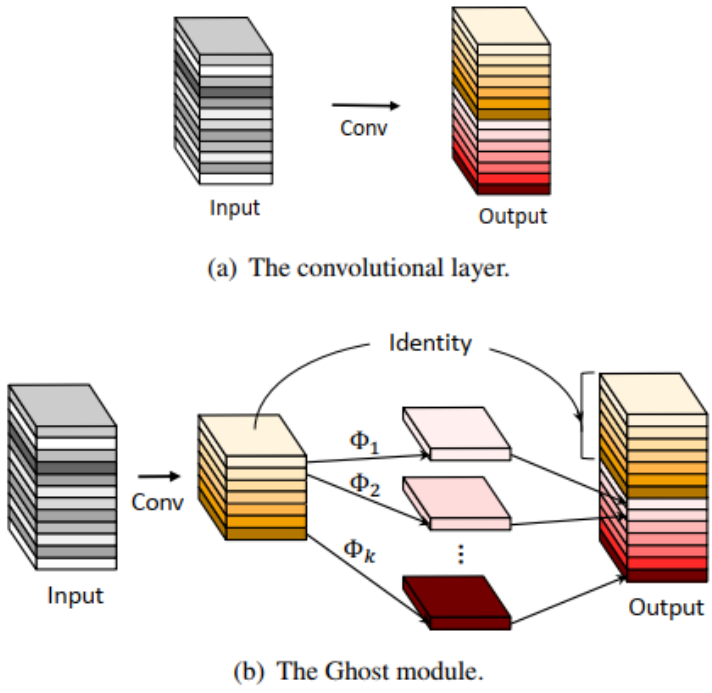


Figura 2.5: Concepto básico de capas convolucionales fantasmas (tomado de [22]).

Se plantea que los mapas de características de salida Y sean "fantasmas" de un grupo de mapas de características intrínsecos Y' a los cuales se les aplica algunas operaciones simples y económicas. Estos mapas de características intrínsecos son de un tamaño más pequeños, pero son producidos por los filtros de convolución ordinarios. Específicamente, m mapas de características intrínsecas $Y' \in \mathbb{R}^{h' \times w' \times m}$ son generados usando la convolución primaria

$$Y' = X * f' \tag{2.2}$$

donde $f' \in \mathbb{R}^{c \times k \times k \times m}$ son los filtros usados, con $m \leq n$ y omitiendo el termino de sesgo por simplicidad. Los hiper-parámetros tales como el tamaño del filtro, el tamaño del *padding* y el *stride* son iguales que en la convolución original (2.1) manteniendo de esta manera sus dimensiones. Entonces, para recuperar estos n mapas de características se propuso la aplicación de una serie de operaciones lineales que se le aplican al Y' para generar s características fantasmas siguiendo con la función:

$$y_{ij} = \phi_{ij}(y'_i), \forall i = 1, \ldots, m, j = 1, \ldots, s \tag{2.3}$$

donde y'_i es el i-ésimo mapa de características de Y' y ϕ_{ij} corresponde al i-ésimo operador lineal para generar el i-ésimo mapa de características fantasma y_{ij}, exceptuando al último $\phi_{i,s}$, que corresponden a el mapeo identidad para preservar las características

intrínsecas. De este modo se dice que $n = m \cdot s$ por lo que el mapa de características corresponde a $Y = [y_{11}, y_{12}, \ldots, y_{ms}]$.

Lo que este módulo propone es dividir el tamaño original de la salida por ejemplo a la mitad y luego a la convolución resultante aplicarle una serie de operaciones lineales y concatenar ambos resultados para al final obtener el tamaño que se deseaba en un inicio, tal y como se ejemplifica en la figura 2.5. Con esto se logra resultados cercanos a las redes neuronales análogas pero logrando reducir a casi la mitad la cantidad de FLOP usadas según los resultados documentados en [22].

2.3.4. Capas de cuello de botella

Las capas de cuello de botella son aquellas que se colocan en las arquitecturas para reducir el número de mapas de características en la red que de otra manera solo irían incrementando. El objetivo de estas capas es eliminar complejidad y peso computacional en la etapa de entrenamiento.

Estas capas consisten en una convolución 1×1 con menos canales de salida que de entrada. Por ejemplo de una capa cuello de botella puede tener como entrada un mapa de características de 512 mapas, y a la salida presentar únicamente 64, eliminando mapas que se consideran innecesarios o demasiado pesados para sostener un crecimiento de red.

2.3.5. Redes neuronales residuales (ResNet)

Las redes neuronales residuales, conocidas como **ResNet** consisten en una arquitectura de red neuronal, que se diferencia de otros modelos de red neuronal en que las conexiones entre capas se realizan tanto en cascada como de manera no lineal, dándose los llamados atajos o "conexiones de salto" para interconectar una capa superior con una inferior saltando capas intermedias. El objetivo de ResNet es dar la posibilidad de entrenar cientos de miles de capas y aun alcanzar un rendimiento aceptable. Usando esta habilidad de representación, el desempeño de aplicaciones de visión de computadora ha alcanzado una mejora considerable en aplicaciones tales como detección de objetos y reconocimiento facial.

Siguiendo con el teorema de la aproximación, dando suficiente capacidad, una red de retroalimentación negativa con una única capa, es suficiente para representar cualquier función; no obstante, esta misma capa puede llegar a ser demasiado grande haciendo que la red caiga en sobre ajuste con los datos de entrada. Para evitar esto, es que se llego a la utilización de las arquitecturas de redes profundas. Desde AlexNet, el estado del arte de las redes neuronales artificiales han seguido la tendencia ir más profundo con AlexNet teniendo 5 capas convolucionales, mientras que VGG y GoogleNet presentan 19 y 22 respectivamente.

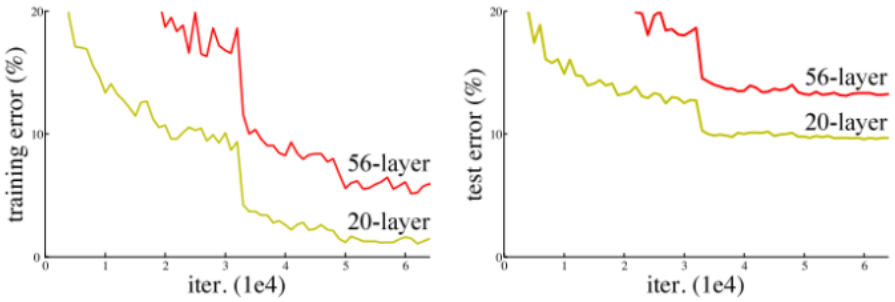


Figura 2.6: Concepto básico de ResNet
(tomado de [23]).

Pero incrementar la profundidad de la red no es simplemente con apilar una capa sobre otra. Redes neuronales con mucha profundidad sufren de la fuga de gradiente (*vanishing gradient problem*) conforme este hace la propagación hacia atrás, la repetición de multiplicaciones hace que el gradiente se haga infinitamente pequeño y en consecuencia los pesos de la red no se actualizan. Como consecuencia, al hacer la red más profunda no se lograba una mejora en la reducción del error, sino que hacia que la red se saturara de manera más rápida (se llega al punto de convergencia, donde el error no sigue disminuyendo).

Con esto presente la idea de ResNet es introducir las *conexiones de atajos identitarios* que conectan la salida de una capa superior con una inferior, saltando una o varias capas intermedias, concatenando la salida de ambas, como se muestra en la figura 2.7, donde la salida X es conectada con la salida $f(x)$, saltando dos capas para realizar la concatenación.

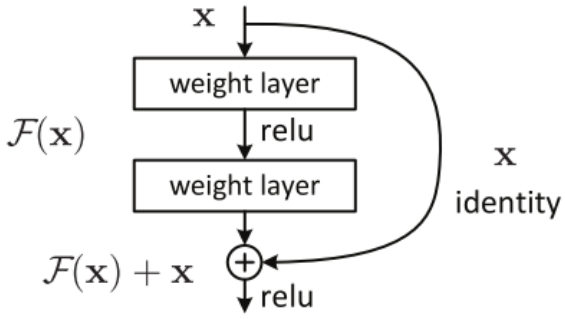


Figura 2.7: Concepto básico de ResNet
(tomado de [23]).

2.3.6. Capa de detección YOLO

You Only Look Once es el estado del arte en lo que respecta a detección de objetos en tiempo real. Esta aproximación se diferencia de trabajos previos en que no es una CNN de clasificación ajustada para realizar detección [14][16]. En cambio, mira la detección de objetos como un problema de regresión para encontrar recuadros delimitadores (*bounding boxes*) y asociar estos a una clase probabilística.

El funcionamiento de YOLO se ilustra en la figura 2.8 donde primeramente se divide la imagen de entrada en una grilla de $S \times S$ píxeles. Si el centro de un objeto cae dentro de una de esas celdas, esa celda es la responsable de determinar el objeto. Cada una de las celdas predice B recuadros delimitadores y un valor de confianza para cada una de las cajas.

La confianza refleja cuánta certeza tiene el modelo de que esa caja contiene un objeto. Si en la caja predicha no hay ningún objeto, la confianza debería ser 0, de lo contrario, se desea que el grado de confianza sea igual a la intersección sobre la unión entre la caja predicha y la caja real (IOU_{pred}^{truth}). Formalmente se define esta confianza como:

$$Confidence = Pr(Object) * IOU_{pred}^{truth} \tag{2.4}$$

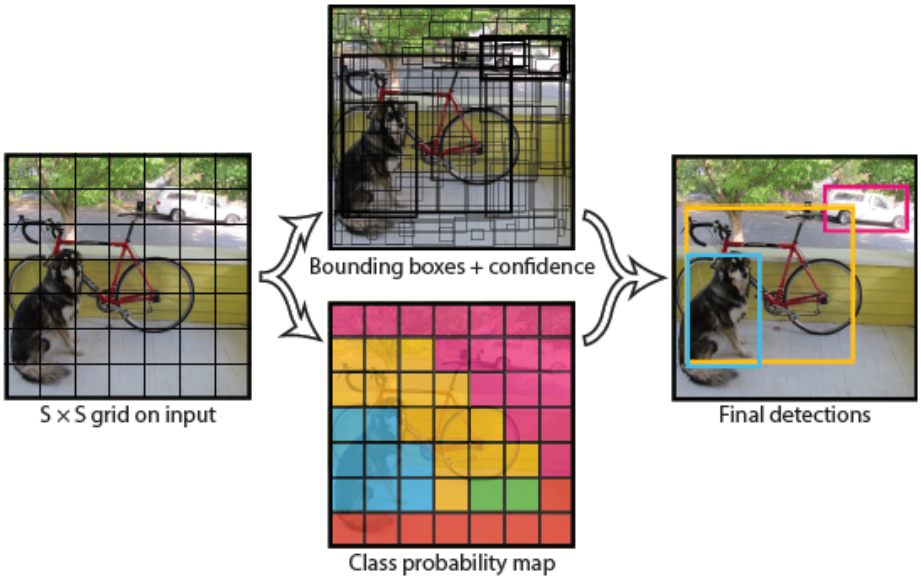


Figura 2.8: Flujo de ejecución de detecciones en YOLO (tomado de [19]).

Cada recuadro delimitador predicho está compuesto por 5 parámetros x, y, w, h junto con la confianza. Las coordenadas (x, y) corresponden al centro del objeto. w es el ancho

del objeto detectado, mientras que h la altura de este. Ambos parámetros son relativos a las dimensiones de la imagen sobre la que se realiza la detección.

Además, cada celda también predice C probabilidades condicionales, $P(Class_i|object)$. Estas probabilidades son condicionales a la celda de la grilla que contiene el objeto. Se calcula solo un set de probabilidades por celda sin importar el número de cajas B. Para el momento de pruebas se multiplica la probabilidad condicional de clase con la confianza predicha de cada caja,

$$P(Class_i|Object) \cdot P(Object) \cdot IOU_{pred}^{truth} = Pr(Class_i) \cdot IOU_{pred}^{truth} \tag{2.5}$$

dando como resultado el grado de confianza específico de cada caja para cada clase individual. Este valor envuelve tanto la probabilidad de la aparición de una clase y qué tan bien la caja predicha encaja con el objeto.

En resumen, el sistema divide la imagen en $S \times S$ celdas, luego para cada celda se predicen B recuadros delimitadores, junto con el grado de confianza para cada caja y además se predicen C probabilidades de clase para cada celda. Esta predicción está codificada en un tensor $S \times S \times (B \times 5 + C)$.

Siguiendo el flujo de la figura 2.2 es necesario especificar a la red los valores de escalamiento de las imágenes de entrada (parámetros *width* y *height*). Este escalamiento repercute directamente en el desempeño de la red, ya que reducir demasiado imágenes muy grandes quita contexto de las mismas, pero dejar tamaños de entrada muy altos hará más lento y computacionalmente pesado el entrenamiento.

La función de activación usada en las capas es la *Leaky Rectified Linear Unit*, también conocida como la función de activación Leaky ReLU, que sigue la ecuación, con un $\alpha = 0{,}1$,

$$\phi = \begin{cases} x & si \quad x > 0 \\ ax & si \quad x < 0 \end{cases} \tag{2.6}$$

Por otra parte, se optimiza el error de suma de cuadrados para la salida del modelo. Se decidió usar este error por su facilidad de optimización, aunque no se alinea con la idea de tener una buena precisión. Para compensar esto lo que se hace es usar dos parámetros λ en la función de pérdida, uno para las cajas que no contienen objetos (λ_{noobj}) y uno para las cajas que sí los contienen (λ_{coord}).

Otro aspecto importante es que se desea que las desviaciones en objetos pequeños sean de mayor importancia que en objetos grandes, por lo que se opta por predecir las raíces cuadradas de los altos y anchos de las cajas en lugar de los altos y anchos directamente. La función de pérdida a optimizar corresponde a,

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{I}_{ij}^{obj}\left[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2\right]$$
$$+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{I}_{ij}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right]$$
$$+\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{I}_{ij}^{obj}\left(C_i-\hat{C}_i\right)^2$$
$$+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{I}_{ij}^{noobj}\left(C_i-\hat{C}_i\right)^2$$
$$+\sum_{i=0}^{S^2}\mathbb{I}_{ij}^{obj}\sum_{C\in classes}\left(p_i(c)-\hat{p}_i(c)\right)^2 \quad (2.7)$$

donde $\mathbb{I}_i^{obj}$ hace referencia a si un objeto aparece en la celda i mientras que $\mathbb{I}_{ij}^{obj}$ se refiere a si la caja j de la celda i tiene una predicción. Para evitar el sobreajuste se colocan capas de *dropout* con un coeficiente de 0.5 después de las capas completamente conectadas y a partir de YOLOv2 se agrega la normalización por lotes (*batch normalization*) a todas las capas convolucionales con la intención de incrementar el mAP. Además, se emplea supresión de no-máximos (*non-maximal suppresion*) para eliminar las múltiples detecciones en los bordes [19].

A partir de YOLOv2 se incorporan los recuadros ancla (*anchor boxes*) que son dimensiones de cajas posibles que la red puede detectar. Esto beneficia las predicciones ya que la red no tiene que generar una caja desde cero sino que predice que tanto la caja predicha se aleja del recuadro ancla. En concreto la red predice 5 coordenadas para cada caja predicha t_x, t_y, t_w, t_h y t_o. Si la celda sobre la que se está haciendo la predicción tiene un corrimiento desde la esquina superior izquierda de la imagen por (c_x, c_y) la caja adapta la función de pérdida para predecir corrimientos a estas posibles cajas ya establecidas. Además se incorpora en el sistema una posibilidad de calcular estos recuadros ancla empleando un algoritmo de k-means [24].

2.3.7. Darknet53

Esta arquitecta de red neuronal es la que se presenta por primera vez en el artículo de YOLOv3. Esta arquitectura es un híbrido entre la expuesta en YOLOv2 (Darknet-19) y una red neuronal residual; implementando sucesiones de capas convolucionales 3×3 y 1×1 pero agregando atajos identitarios, igual que en el ResNet. La arquitectura es la que se muestra en la figura 2.9. La idea de esta arquitectura siguiendo con la filosofía de YOLO, es mantener una eficiencia alta pero teniendo un tamaño reducido, disminuyendo el tiempo de detección y entrenamiento.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figura 2.9: Concepto básico de Darknet53
(tomado de [24]).

2.4. Métricas de detección de objetos

2.4.1. Intersección sobre la unión

En primera instancia para la detección de objetos es necesario tener un criterio para determinar si una detección fue válida (verdadero positivo) o no (falso positivo). La intersección sobre la unión (IOU) da un valor numérico para diferenciar entre dos escenarios en una prueba supervisada. Este valor umbral es arbitrario y seleccionado por el programador. Por ejemplo, un usuario puede definir una detección exitosa para todas las predicciones con un IOU superior al 0.40.

La intersección sobre la unión es una medida basada en el índice de Jaccard que evalúa la superposición entre dos cuadros delimitadores (figura 2.10). Para esta métrica es necesario un cuadro delimitador verdadero (B_{gt}) y un cuadro delimitador predicho (B_p). De este modo el IOU esta dado por la división entre la intersección de estos

cuadros y la unión de los mismos siguiendo [25],

$$IOU = \frac{B_p \cap B_{gt}}{B_p \cup B_{gt}} \tag{2.8}$$

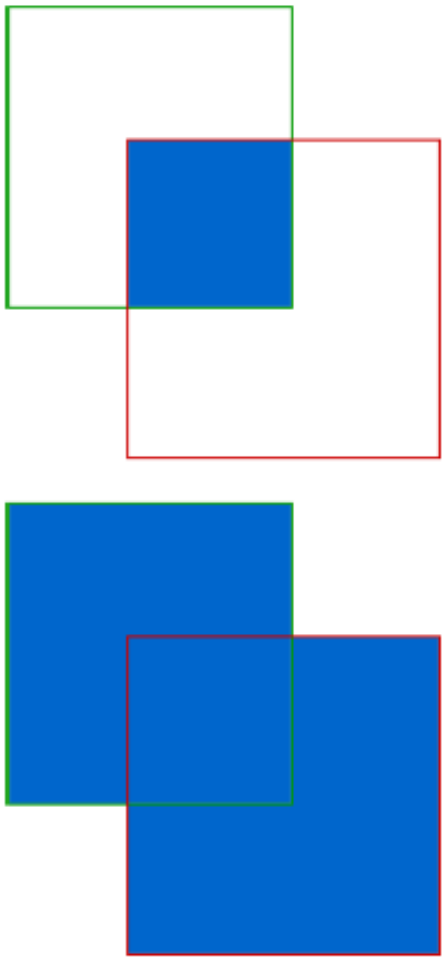


Figura 2.10: Intersección (superior) e Unión (inferior) entre los cuadros delimitadores (tomado de [25]).

2.4.2. Clasificación de detecciones

Algunos conceptos básicos utilizados como métricas incluyen el umbral, que es el valor de IOU determinado para que una detección pueda considerarse como correcta y también se presenta la matriz de confusión mostrada en la figura 2.11.

Figura 2.11: Matriz de Confusión Para Clasificación Binaria.

De esta matriz se extraen los conceptos de verdadero positivo (TP), que denotan una detección que ha sido asignada a la clase correcta, los falsos positivos (FP), que indican detección que no debería ocurrir o que ha sido asignada a una clase incorrecta. Los falsos negativos (FN) que indican una detección que no se realizó de un objeto que sí existe y finalmente los verdaderos negativos (TN) que no se utilizan y representa una omisión correcta; en otras palabras que la red no detectó un objeto en un lugar donde no debería haber un objeto.

A partir de estos conceptos se establecen las métricas como la precisión, que es la probabilidad de que una detección correcta corresponda a un objeto real. En otras palabras, es la proporción de TP que hay en el total de detecciones. El cálculo de esta se realiza siguiendo la ecuación,

$$P = \frac{TP}{TP + FP} \tag{2.9}$$

Por otra parte se tiene la exhaustividad o también conocida en inglés como *recall*, correspondiendo a la probabilidad de que un objeto existente sea detectado correctamente. El cálculo de esta se realiza siguiendo,

$$R = \frac{TP}{TP + FN} \tag{2.10}$$

Finalmente se tiene la métrica F que es simplemente una relación entre entre la precisión y la exhaustividad, siendo calculada con la ecuación,

$$F = \frac{2PR}{P + R} \tag{2.11}$$

2.4.3. Precisión promedio

La precisión promedio, conocida también como *Average Precision* en inglés, es otro método para comparar el rendimiento de la detección de objetos, esto basado en el cálculo del área bajo la curva de la gráfica de precisión interpolada (P_{inter}) contra exhaustividad (r), siguiendo:

$$AP = \frac{1}{11} \sum_{\widetilde{r} \in [0;0,1;\cdots;0,9;1]} P_{inter}(\widetilde{r}) \tag{2.12}$$

Esta gráfica es elaborada con un set de datos etiquetados diferente al usado para el entrenamiento, denominado set de prueba.

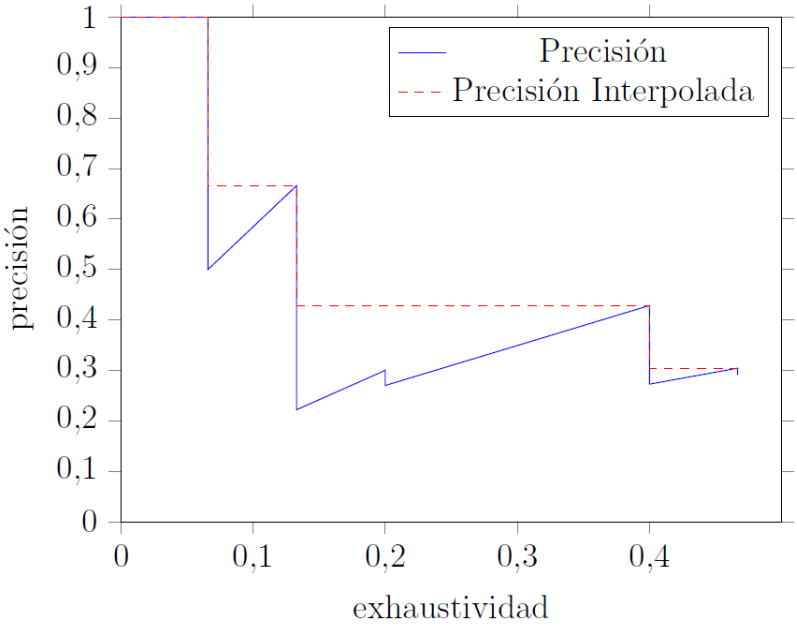


Figura 2.12: Ejemplo de curva P vrs R
(tomado de [25]).

En (2.12) $P_{inter}(\widetilde{r})$ representa la precisión interpolada para cada uno de los once niveles de exhaustividad discretos y equidistantes $\widetilde{r} = [0; 0,1; 0,2 \cdots ; 0,9; 1]$. Para el cálculo de esta variable se realiza una detección sobre todo el set de datos de prueba y se calcula la precisión p y la exhaustividad para cada imagen por separado. Luego se promedia el valor de p de todas las imágenes con un r igual, para finalmente tomar el valor máximo de P en cada nivel $\widetilde{r}$ como el valor final de $P_{inter}(\widetilde{r})$ estableciendo que $P_{inter}(\widetilde{r}) = max(P(r))$. Este método se ilustra con la figura 2.12.

2.4.4. Divergencia Jensen-Shannon

Divergencia Jensen-Shannon, es también conocido como el total de divergencia medio [26] es un método estadístico y probabilístico que realiza una medida de la similitud entre dos distribuciones probabilísticas, siguiendo la ecuación,

$$JSD\left(P||Q\right) = \frac{1}{2}D_{KL}\left(P||\frac{p+q}{2}\right) + \frac{1}{2}D_{KL}\left(Q||\frac{p+q}{2}\right) \tag{2.13}$$

donde D_{KL} corresponde a la divergencia de Kullback-Leibler, dada por la formula,

$$D_{KL}\left(P||Q\right) = \int_{-\infty}^{\infty} p(x)\left[\log\frac{p(x)}{q(x)}\right]dx \tag{2.14}$$

Tanto la divergencia Kullback-Leibler como la de Jensen-Shannon son usadas para determinar el parentesco entre dos conjuntos de datos, pero la Jensen-Shannon posee la ventaja de tener un resultado acotado y ser un cálculo conmutativo. Los resultados pueden ir de 0 a 1, siendo 0 cuando las dos distribuciones en comparación son idénticas, mientras que una divergencia de 1 representa conjuntos mutuamente excluyentes. En este trabajo se usa esta métrica para determinar la similitud entre los datos de las diferentes plantaciones.

2.5. Trabajos relacionados con detección en agricultura

A pesar de su amplio éxito en distintas disciplinas, las CNN aplicadas para la agricultura presentan una aplicación limitada debido a la escasa cantidad de datos etiquetados disponibles [27]. Esta escasez de datos, lleva a las investigaciones a tener que desarrollar sus propios sets de datos, tarea que consume tiempo y recursos [27][28]. A pesar de esto existe una gran variedad de investigaciones que se enfocan en detección temprana de enfermedades, calidad de producto, etc.

Algunas investigaciones como [27] han buscado diseñar CNN robustas de clasificación de café según su calidad usando un set de datos pequeño (cerca de 2100 imágenes) con gran variedad, logrando alcanzar precisiones de hasta el 89 % con un modelo VGG pre entrenado. Los investigadores de [27] recomiendan usar redes neuronales pequeñas cuando se cuenta con un set de entrenamiento reducido, con la intención de no caer en problemas de sobreajuste.

Otras investigaciones como [28][29][30] se enfocan en la clasificación temprana de enfermedades en las plantas, específicamente en el café. En ambas investigaciones lo

que se busca es diferenciar entre hojas sanas y hojas que posean alguna enfermedad. Todas estas investigaciones usan redes neuronales en cascada para cumplir con estas tareas, dichos modelos presentan una primera etapa con capas convolucionales seguidas de una sección de capas completamente conectadas.

Como resultados estas investigaciones logran una clasificación con una precisión del 98 % y 99 %. En el ámbito de detección se encuentran algunas investigaciones como [31][32] donde se usan algoritmos tradicionales de visión por computador como k-means para la detección de granos de cacao y árboles de eucalipto en imágenes satélites. Esta última investigación presenta una aproximación similar a lo que se intenta en el presente trabajo, que es la detección de plantas de café, pero empleando imágenes áreas cercanas tomadas con drones y utilizando redes neuronales artificiales para la detección de los especímenes.

2.6. Trabajos previos para la detección de cafetos

Como antecedentes directos a esta investigación se citan dos tesis de Licenciatura en Ingeniería Electrónica y uno en Ingeniería en Computadores. Los tres proyectos son realizados en el Instituto Tecnológico de Costa Rica. El primer proyecto titulado (*Diseño de un sistema de cuantificación automática de biomasa basado en procesamiento de imágenes y fotogrametría con vehículos aéreos no tripulados*) [4] se centra en el diseño de un sistema software para conteo automatizado de árboles en *productos fotométricos* empleando técnicas de segmentación.

Esta investigación fue realizada por el ingeniero Javier Cordero en el Laboratorio de Fotogrametría. Se emplean filtros y se aplica segmentación para detectar árboles en ortomosaicos. Al final de la investigación se obtiene un error del 3.22 % en mapas con resolución espacial de 2cm/píxel, mientras que para imágenes de resolución espacial de 5cm/píxel el error desciende a 1.39 %. La intención de la detección de los árboles es poder obtener características de la plantación, como por ejemplo cuantificación de biomasa [4].

Por otra parte, el proyecto *Sistema de entrenamiento de redes neuronales artificiales con predicción eficiente en arquitecturas embebidas*, realizado por el ingeniero Emmanuel Madrigal, se describe una estrategia para entrenar redes neuronales convoluciones en arquitecturas embebidas, comprobando el funcionamiento de la misma realizando detección de mariposas en un ambiente confinado. Se emplea el sistema *You Only Look Once* ($YOLO$), obtienen resultados de mAP del 40.26 % para imágenes de 640×352, reportando un tiempo de respuesta de 32ms en el mejor de los casos [25].

Finalmente el tercer proyecto a tomar en cuenta es el titulado *Seguimiento de plantaciones de café a través de fotogrametría UAS* [5] en el cual se toman como base los dos proyectos primeramente mencionados para elaborar un sistema de seguimiento de plantaciones cafetaleras donde se detectan los cafetos y posteriormente se intenta obtener información de las plantas.

Para la tarea de detección se usó del sistema YOLOv3 obteniendo una detección del 90.2 % de los cafetos a un IoU con umbral de 0.35 en imágenes con una resolución espacial de 4 cm/píxel, dando un mAP@50 de 75.50 %. Sin embargo, los resultados de los modelos matemáticos que relacionan el índice NDVI en los mapas y las características vitales de los cafetos mostraron una baja correlación. Un ejemplo de detección es la que se muestra en la figura 2.13.

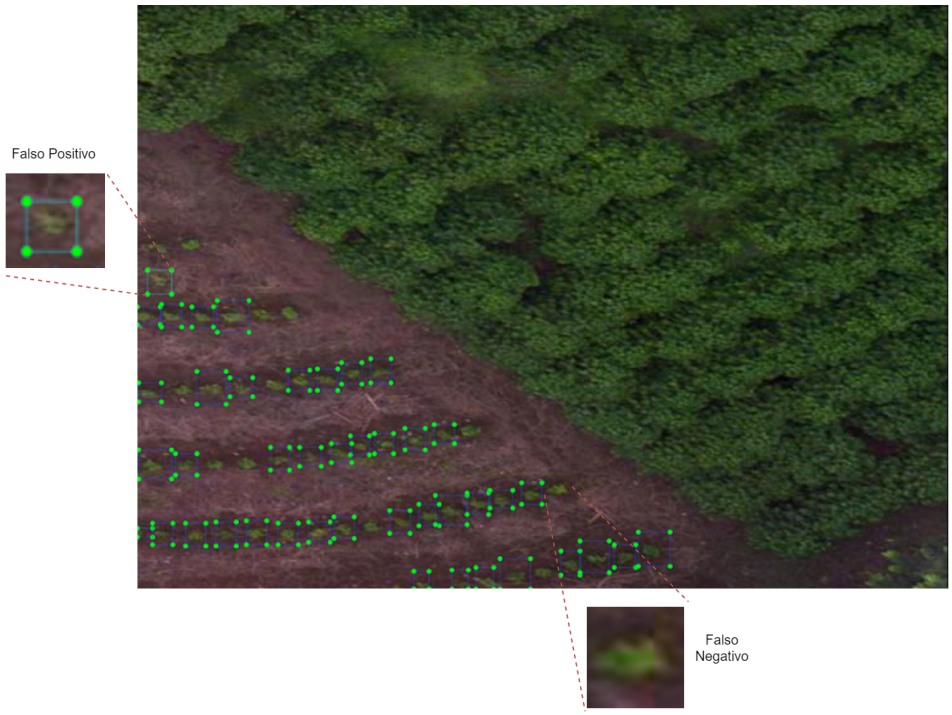


Figura 2.13: Resultado en imagen de prueba con resolución a 4 cm/píxel [5].

La presente investigación es una continuación de este último proyecto, enfocado únicamente en el apartado de detección de cafetos. Empleando la nueva versión de YOLO, agregando nuevas plantaciones, aplicando modificaciones en la forma de entrenamiento y empleando una arquitectura con menor cantidad de parámetros, se tiene como objetivo mejorar las métricas de detección y lograr un modelo más robusto.

Capítulo 3

Metodología de detección de cafetos con redes neuronales convolucionales

En este capítulo se detalla la metodología usada para la detección de cafetos con redes neuronales artificiales. Con base en el diagrama de flujo de la figura 1.1, expandiendo cada uno de los cuatro bloques que aparece en la misma tal como se muestra en el diagrama de la figura 3.1. De esta manera, primero se analiza y discute las características de las imágenes y etiquetas. Finalmente, se indica la forma en que se hace la división entre datos entrenamiento y prueba.

Una vez establecidos los bloques de entrenamiento y prueba, se selecciona una serie de arquitecturas basadas en la detección YOLO, que al entrenarlas da como resultado una base comparativa para finalmente presentar en una arquitectura especializada en detección de cafetos, y verificar su efectividad con los resultados de las arquitecturas del bloque anterior.

3.1. Generación y pre-procesamiento de imágenes

El primer bloque se enfoca en la confección del conjunto de datos de imágenes etiquetadas. Este fue elaborado desde cero como consecuencia de no encontrar ningún conjunto de datos etiquetados que cumpla con características similares para la detección de cafetos en productos fotogramétricos desde una vista aérea tomadas por un dron a baja altura.

Los datos crudos que el proyecto dispone son tomados de cinco plantaciones, con climas y vegetaciones distintas entre si. Estas están concentradas en la parte central del país en alturas de entre los 900 y 1600 msnm. Además, algunas parcelas presentan cafetos jóvenes y ampliamente espaciados, mientras otras contienen cafetos adultos y con altos grados de traslape entre ellos. La figura 3.2 sirve como referencia para visualizar la ubicación de cada zona donde se tomaron muestras.

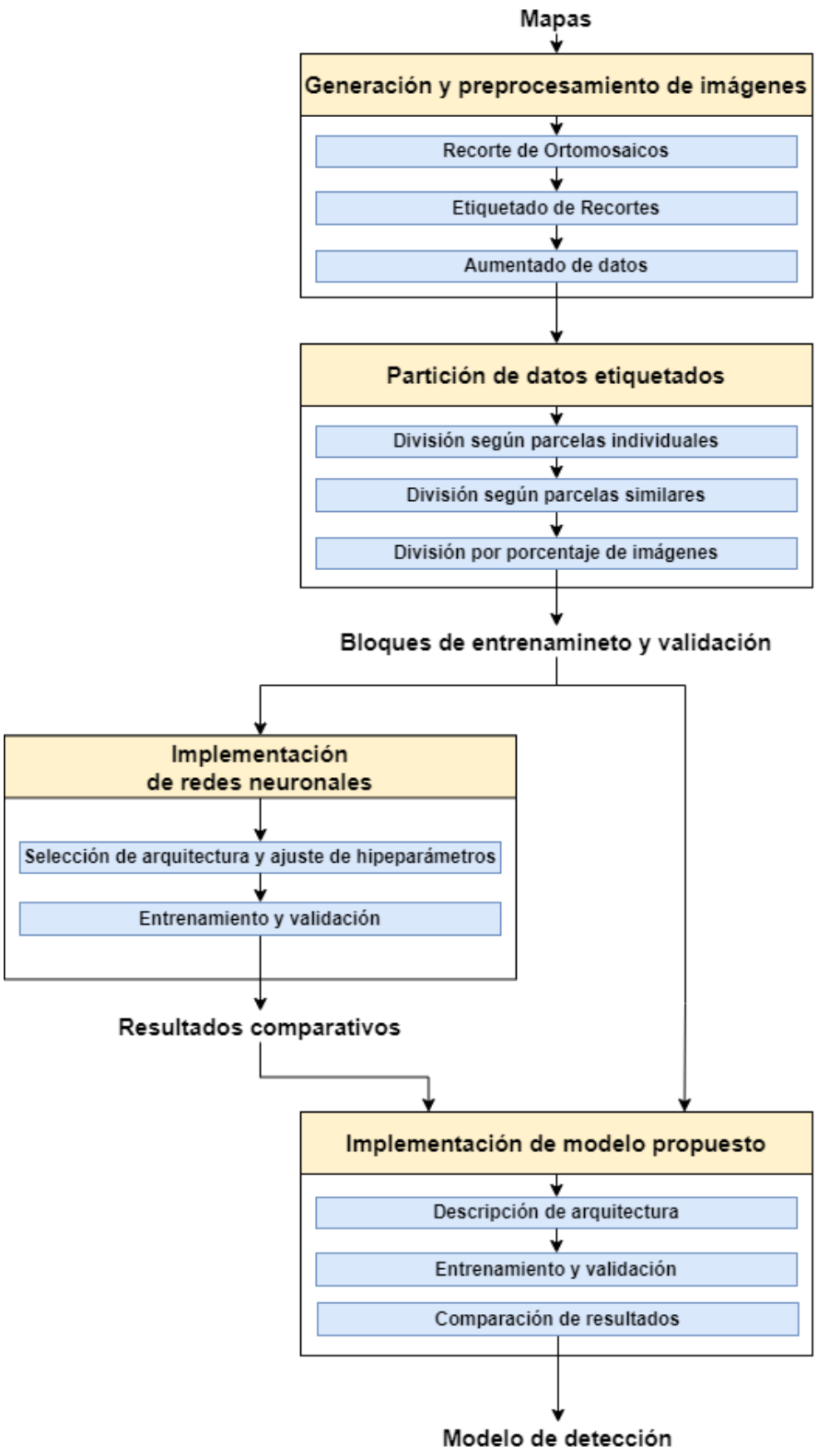


Figura 3.1: Planteamiento de la solución para la identificación de plantas de café (tomado de).

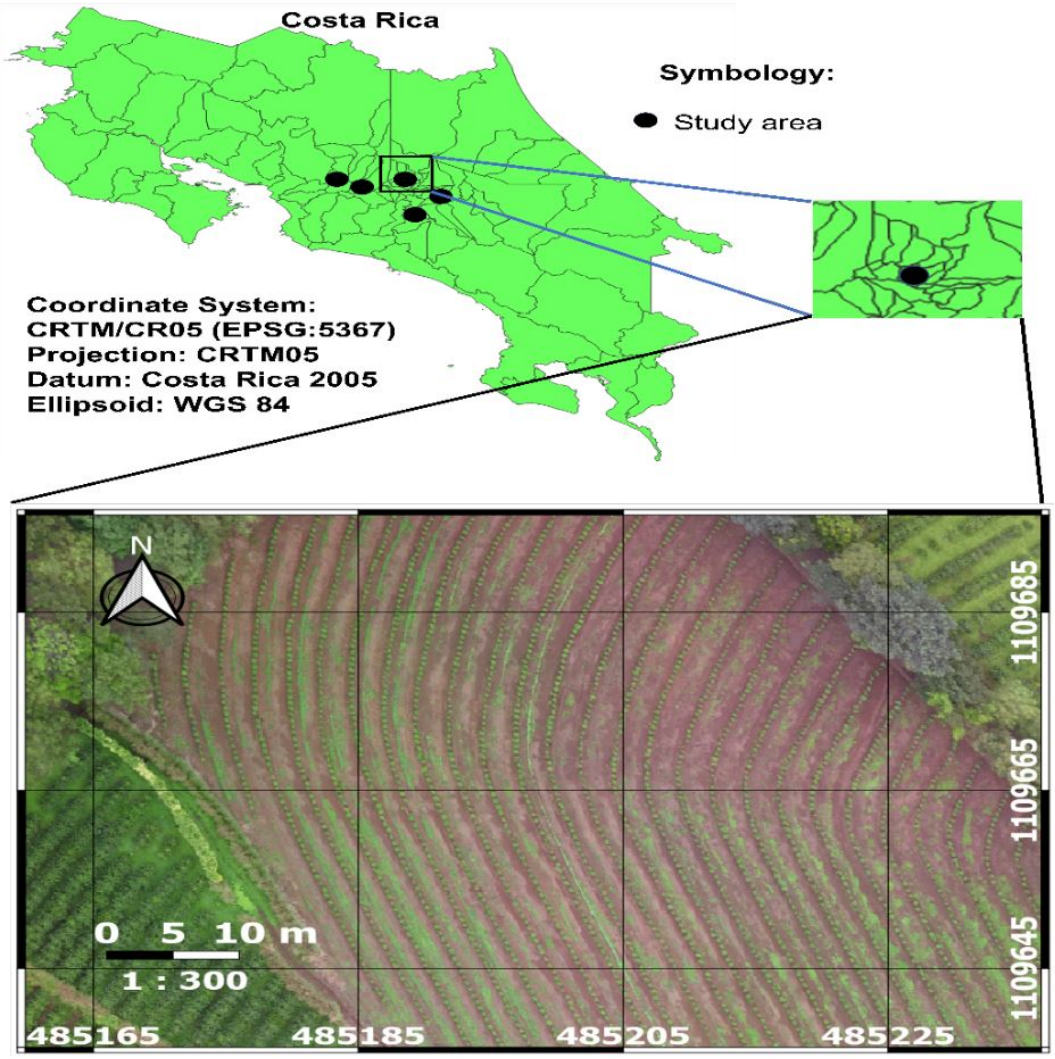


Figura 3.2: Localización de parcelas usadas para la toma de imágenes.

Como indica el diagrama de flujo de la figura 3.1, la generación del conjunto de datos se divide en tres secciones: recorte de ortomosaicos, etiquetado de recortes y aumentado de datos. Todos estos sub-bloques corresponden a la solución de un problema que surge al inicio del desarrollo del proyecto.

Una limitante importante es la escasa cantidad de mapas disponibles. La primera aproximación para aumentar la cantidad de muestras fue aprovechar el hecho de que los productos fotogramétricos de los que se dispone poseen dimensiones que superan los 10000×10000 píxeles. De esta manera, al recortar estos mapas en extractos de 480×640 píxeles se logran generar de los 11 mapas disponibles 1413 imágenes, tal como se reporta en la tabla 3.1.

Una vez hecho el recorte de los mapas se procede a etiquetar estas imágenes indicando con recuadros delimitadores dónde se encuentra cada cafeto. Todos los cuadros delimitadores siguen el formato YOLO indicando el número de clase que identifica el

objeto (c), las coordenadas del centro del recuadro (x, y) y el ancho y largo de este (w, h). Todos estos valores son normalizados con las dimensiones de la imagen. Estas etiquetas se realizaron de forma manual, llevado a cabo por una persona y se le aplicó varias iteraciones de revisión a lo largo de un años.

Nombre de la zona	**Cantidad de mapas por zona**	**Cantidad de cafetos por mapa**	**Cantidad de imágenes por zona**
Acosta	1	1103	71
Barva de Heredia	6	2866	900
Grecia	2	2461	165
Naranjo	1	3611	186
Tres Ríos	1	909	91

Tabla 3.1: Distribución de imágenes del set de datos según localización.

Finalmente, se procede a hacer el aumentado de datos. Cerca de mil muestras sigue siendo una cantidad pequeña de imágenes, por lo que se les aplican transformaciones con el fin de simular imágenes nuevas. Este proceso se realiza después del etiquetado para aplicar el aumentado también a las etiquetas.

Las transformaciones que se efectúan incluyen rotaciones de 30°, efecto espejo, escalado del 5 % y 10 %, cambios en contraste e iluminación y cambios en color predominante. Esta última consiste en bajar o subir la intensidad del color verde en todos los píxeles donde este sea el color predominante; algunos ejemplos de estas transformaciones se muestran en las figuras 3.3 y 3.4. Como resultado se alcanzan más de 50 mil imágenes (tabla 3.2).

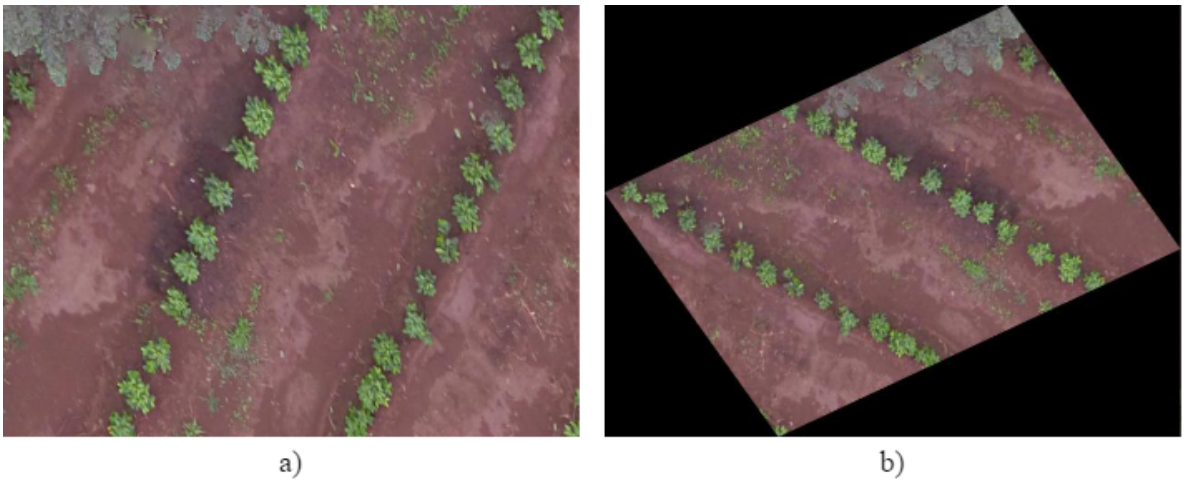
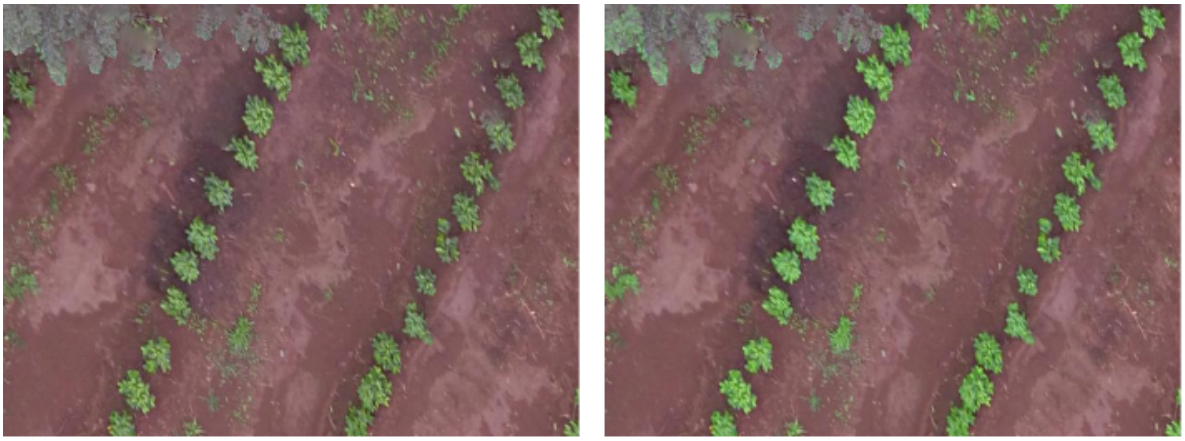


Figura 3.3: a) Imagen original b) Imagen con rotación de 30° y con efecto espejo.



a) b)

Figura 3.4: a) Imagen original b) Imagen con aumento de color verde en los píxeles donde el canal predominante es el verde.

Nombre de la zona	**Cantidad de imágenes por zona**	**Cantidad de imágenes por zona después del aumentado**
Acosta	71	2557
Barva de Heredia	900	32616
Grecia	165	6070
Naranjo	186	6888
Tres Ríos	91	3278

Tabla 3.2: Distribución de imágenes del set de datos según localización.

Como segundo objetivo, esta sección expone las diferencias existentes entre las plantaciones en estudio. De la figura 3.5 a la figura 3.9 se muestra un ejemplo de ortomosaico de cada una de las parcelas usadas. Destacando que la parcela de Barva Heredia (muestra en la figura 3.10), fue la única a la que se le ha dado un seguimiento desde la siembra y como consecuencia es de la que se tiene mayor cantidad de muestras. Esta plantación dispone únicamente de cafetos jóvenes y presenta el mayor control en lo que se refiere a espaciado entre plantas y eliminación de otras vegetaciones en la parcela, así como el cuidado general de los cafetos, abono periódico y podado adecuado de los árboles.

Por otra parte, la plantación de Tres Ríos (figura 3.11) presenta tanto cafetos en etapa adulta como en etapa joven. Además, tiene la diferencia de presentar árboles bananeros junto a los cafetos, con el fin de darle sombra a estos. Esta parcela presenta maleza entre cafetos y estos no están ordenados de forma regular.

La parcela de Acosta por su parte presenta también árboles de banano dentro de la parcela, teniendo una mayor densidad cafetos por unidad de área, pero a diferencia de las dos parcelas anteriores, presenta una gran cantidad de maleza y arboles que

obstaculizan la visualización de los cafetos desde la vista superior que se dispone. En la figura 3.12 se muestra un fragmento de la parcela.

La plantación de Grecia contiene únicamente cafetos adultos y una mayor concentración de cafetos por unidad de área, por este motivo existe traslape entre los cafetos. Además, los ortomosaicos de esta zona posee la particularidad de presentar la mayor cantidad de sombras en comparación con las demás localizaciones. En las figura 3.13 se muestra un fragmento de la parcela de Grecia. Finalmente, la plantación de Naranjo comparte casi todas estas características, exceptuando la presencia de sombras y con la diferencia de que esta parcela presenta árboles en la plantación. En la figura 3.14 se muestra un fragmento de la parcela de Naranjo.

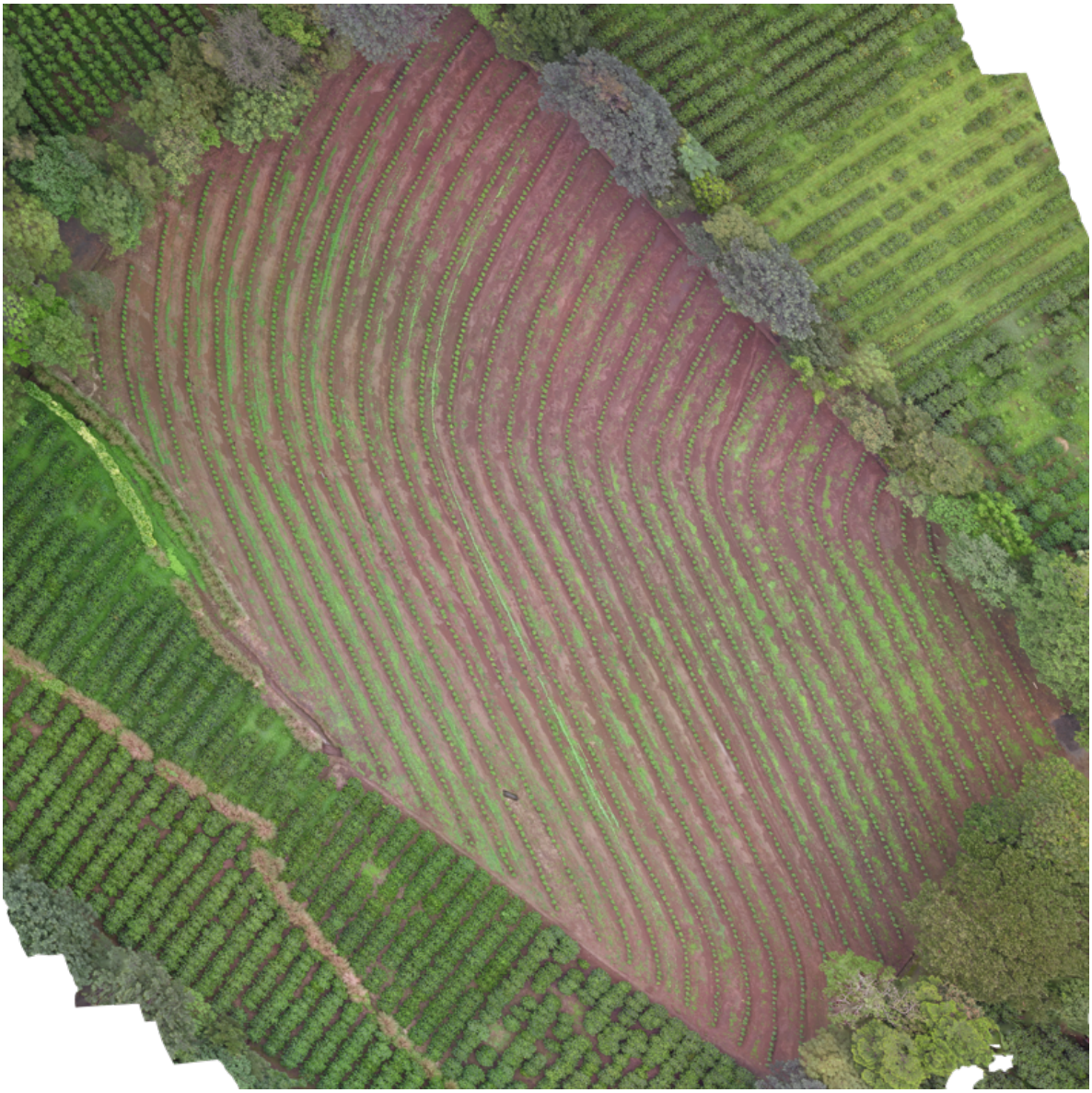


Figura 3.5: Parcela de Barva Heredia.

Figura 3.6: Parcela de Tres Ríos.

Figura 3.7: Parcela de Acosta.

Figura 3.8: Parcela de Grecia.

Figura 3.9: Parcela de Naranjo.

Figura 3.10: Muestra de la parcela de Barva Heredia.

Figura 3.11: Muestra de la parcela de Tres Rios.

Figura 3.12: Muestra de la parcela de Acosta.

Figura 3.13: Muestra de la parcela de Grecia.

Figura 3.14: Muestra de la parcela de Naranjo.

3.2. Partición de datos etiquetados

Con la base de datos ya confeccionada y analizada se procede a dividir las imágenes entre set de entrenamiento y prueba. No obstante, no se realiza una única división, por el contrario se establecen varias permutaciones. El primer criterio de división consiste en dejar una parcela en su totalidad como set de prueba, mientras que el resto se usa para entrenar las redes. Esta aproximación tiene como fin determinar si la red logra generalizar o no que es un cafeto y saber si no se esta cayendo en sesgo.

El segundo criterio de división consiste en tomar un porcentaje de imágenes de cada parcela y validar los modelos con este. El proceso consiste en aislar el 10 % de cada parcela de manera aleatoria antes de efectuar el aumentado de datos y luego de esa división aplicar el aumentado de datos, dejando el 90 % de las imágenes como el set de entrenamiento. Esta aproximación aunque se espera tenga mejores resultados, tiene la desventaja de caer en sesgo, al tener información de cada parcela. Todas las permutaciones se muestran en la tabla 3.3.

En cada una de las permutaciones el set de entrenamiento durante el entrenamiento de las redes neuronales se divide de manera aleatoria en un 90 % para entrenamiento y un 10 % para validación.

Número de permutación	Localización empleada como set de prueba	Localización empleada como set de entrenamiento
1	Acosta	Todas las parcelas restantes
2	Barva de Heredia	Todas las parcelas restantes
3	Grecia	Todas las parcelas restantes
4	Naranjo	Todas las parcelas restantes
5	Tres Ríos	Todas las parcelas restantes
6	10 % de cada localización	90 % de cada localización

Tabla 3.3: Permutación de localizaciones para establecer el set de prueba.

Finalmente se hace una división más tomando en cuenta los resultados del análisis de las imágenes, agrupando las parcelas que presentan más similitudes entre sí y usando una como prueba el resto como entrenamiento. Esta división se muestra en la tabla 3.4. Las parcelas de Grecia, Acosta y Naranjo presentan una paleta de colores similar aunque Acosta presenta dimensiones de recuadros delimitadores muy distintos a las otras dos. Por otra parte la parcela de Barva y de Tres Ríos se agrupan juntas únicamente por que no se parecen a las otras 3, aunque estas no comparten ni paleta de colores ni dimensiones de recuadros delimitadores o cantidad de cafetos por parcela.

Bloque	Parcelas para entrenar	Parcelas para validar
Bloque 1	Grecia y Acosta	Naranjo
Bloque 2	Barva	Tres Ríos

Tabla 3.4: Bloques de parcelas similares.

3.3. Modelos de red neuronal seleccionados

Las arquitecturas propuestas para detección de cafetos se basan en arquitecturas ya propuestas en otros estudios [33][34]. De este modo, siguiendo con la figura 3.1. El tercer paso a seguir es entrenar arquitecturas de red neuronal con las permutaciones especificadas en las tablas 4.7 y 3.4. En esta subsección se describen estas arquitecturas.

Las arquitecturas seleccionadas se muestran en la tabla 3.5. Todas estas están basadas en la metodología de detección propuesta por YOLO, descrita en el capítulo anterior, la tabla describe a nivel general el tamaño de cada arquitectura, indicando la cantidad de capas que posee, el número de parámetros y la cantidad de operaciones que se requieren para hacer una detección (GFLOP).

Arquitectura	Número de parámetros	Número de capas	GFLOPs
YOLOv4	61 523 734	333	154.9
YOLOv4-tiny	8 669 876	59	12.9
YOLOv5s	7 022 326	270	15.8
YOLOV5s-ghost	3 684 542	453	8.1
YOLOv5m	20 871 318	369	48.0
YOLOv5l	46 138 294	468	107.9

Tabla 3.5: Tamaño de las arquitecturas de redes neuronales.

Todas las arquitecturas, se entrenan y validan haciendo uso el servicio dado por **Google Colab Pro+**, el cual presta por periodos de tiempo un servidor con un sistema operativo Linux que cuenta con una GPU Tesla P100-PCIE, 60 GB de memoria RAM DDR4 y 200GB de capacidad de almacenamiento SSD. Se escoge esta opción por el desempeño que brinda en comparación con el costo, teniendo la limitante de que el entorno se borra automáticamente cada 48 horas. Para solucionar este el problema de eliminación del entorno, se liga este con la herramienta en línea **Weight & Biases**[35]; de esta manera se envían las métricas de las redes incluyendo el mAP@0.5, mAP@0.5:0.95, exhaustividad, precisión, resultados de la función de costo, etc. Además, se registra el consumo de los recursos computacionales usados durante cada entrenamiento.

Todas las arquitecturas de los modelos de aprendizaje autónomo son descritas entre la tablas 3.6 y 3.11. En ellas se especifica el número de módulo, tipo de módulo, el número de veces que se repite este módulo antes de pasar al siguiente, número de módulo de entrada y los detalles asociados a cada módulo. Si el valor de la columna de número de entrada es -1, esto significa que la entrada de ese módulo es la salida del módulo anterior. Si esta columna contiene varios números, significa que tiene varias entradas simultáneas.

Por otra parte, la columna de detalles indica la configuración de las características de cada módulo. De este modo los valores de esta columna tienen significados particulares para cada módulo. Por su parte el módulo convolucional indica la integración de una capa convolucional común y sus detalles indican el número de filtros de salida, las dimensiones del kernel, el stride, y el *padding*, en ese orden especifico. En otras palabras, si se tiene $[16, 3, 1, None]$ significa que se tienen 16 filtros de salida, un kernel de 3×3, un *stride* de 1 y sin *padding*.

Entre los módulos que alteran las dimensiones de las capas se tiene el *MaxPoll* siendo una capa de agrupación de valor máximo. En otras palabras, la salida es igual al valor más alto registrado por el kernel de dimensiones $n \times n$ y en el detalle lo único que se especifica es el valor de n. Por el contrario, el módulo *Upsample* expande el valor de un píxel en una matriz $n \times n$ siguiendo un criterio de interpolación específico, ambos parámetros se especifican en el detalle. Por otra parte el *ZeroPad*, se encarga de aplicar *padding* de zero a las capas de entrada, especificando la cantidad de pixeles de padding en el detalle. Finalmente, la capa de concatenación *concat* concatena dos capas no

consecutivas, y no posee detalle.

Para la arquitectura YOLOv4 se incorpora el módulo cuello de botella el cual consiste de dos capas convolucionales con kernels 1×1 y 3×3 seguidos por una concatenación con la entrada, teniendo como única variable el número de n de filtros de salida, como se muestra en la figura 3.15. Al final de las arquitecturas se encuentra la capa de detección, la cual toma los resultados de niveles especificados por el diseñador y los usa para hacer la detección.

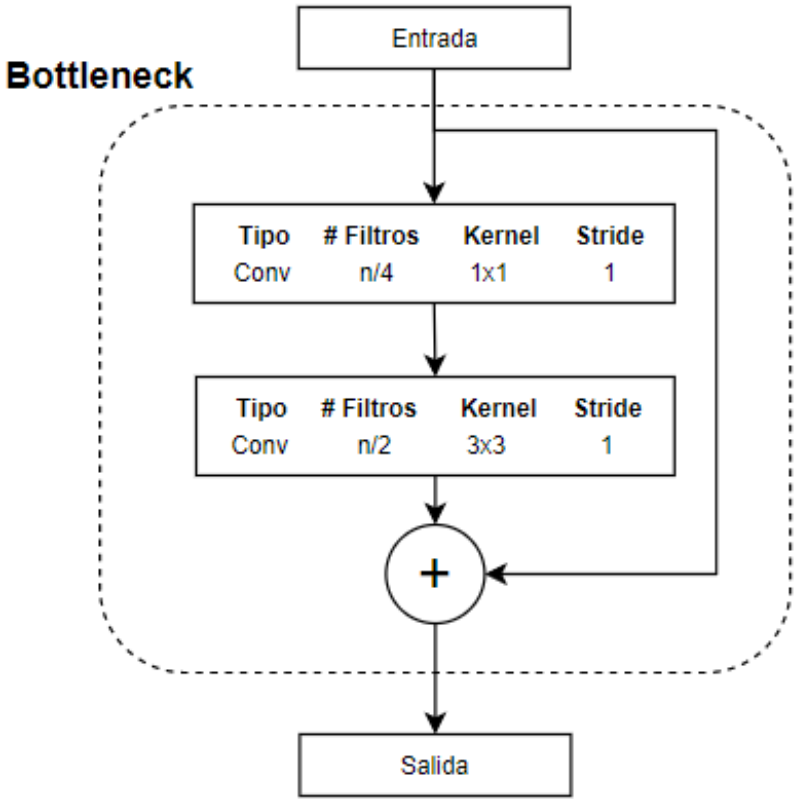


Figura 3.15: Arquitectura módulo de cuello de botella.

Luego tenemos las arquitecturas de la serie YOLOv5 (tablas 3.8, 3.9 y 3.10). Todas estas arquitecturas usan los mismos tipos de módulos, variando únicamente en la cantidad veces que algunos de estos se repiten y el número de filtros en las capas convolucionales. En estas arquitecturas aparece el módulo *Spatial Pyramid Pooling* (SPPF), el cual se describe en el diagrama que se muestra en al figura 3.16.

El módulo *Spatial Pyramid Pooling* (SPPF) consiste en 3 etapas de MaxPoll concatenandas y luego haciéndolas pasar por dos capas convolucionales. Los detalles que se pueden modificar en este módulo son la dimensión del kernel de agrupación y la cantidad de filtros de salida. Otro módulo que se agrega en la serie YOLOv5 es el C3, el cual consiste en una serie de n módulos cuello de botella, seguidos de 3 capas convolucionales con kernel 1×1 con una salida de k filtros. Tanto k como n son especificados en los detalles de este módulo.

Finalmente, la última arquitectura seleccionada es la YOLOv5s-Ghost. La intención de esta arquitectura es probar la efectividad de las capas Ghost, tanto en rendimiento computacional como en rendimiento de detección. Todos los módulos de esta arquitectura son exactamente iguales a los descritos anteriormente pero cambiando todas las capas convolucionales por capas convolucionales fantasma. En todas las arquitecturas la capa final de *Detect* consiste en una la capa de detección YOLO descrita en el capítulo 2.

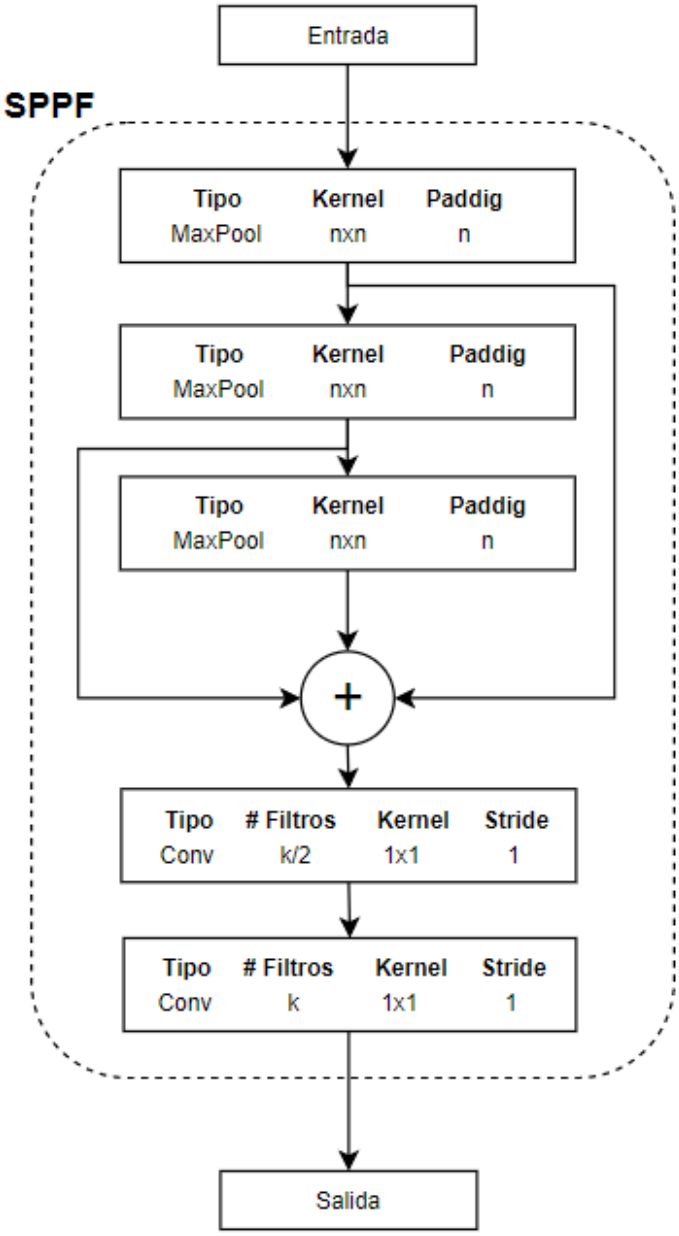


Figura 3.16: Arquitectura módulo SPPF.

Número Módulo	Tipo Módulo	Número Repeticiones	Número entrada	Detalles
0	Convolucional	1	Imágenes base	16, 3 , 1, None
1	MaxPoll	1	-1	2
2	Convolucional	1	-1	32, 3, 1, None
3	MaxPoll	1	-1	2
4	Convolucional	1	-1	64, 3, 1, None
5	MaxPoll	1	-1	2
6	Convolucional	1	-1	128, 3, 1, None
7	MaxPoll	1	-1	2
8	Convolucional	1	-1	256, 3, 1, None
9	MaxPoll	1	-1	2
10	Convolucional	1	-1	512, 3, 1, None
11	ZeroPad	1	-1	2
12	MaxPoll	1	-1	2
13	Convolucional	1	-1	1024, 3, 1, None
14	Convolucional	1	-1	256, 1, 1, None
15	Convolucional	1	-1	512, 3 , 1, None
16	Convolucional	1	-1	256, 1, 1, None
17	Upsample	1	-1	2
18	concat	1	-1,8	-
19	Convolucional	1	-1	256, 3 , 1, None
20	Detect	1	19,15	-

Tabla 3.6: Arquitectura YOLOv4-Tiny.

Número Módulo	Tipo Módulo	Número Repeticiones	Número entrada	Detalles
0	Convolucional	1	Imágenes base	32, 3 , 1, None
1	Convolucional	1	-1	64, 3 , 1, None
2	Bottleneck	1	-1	64
3	Convolucional	1	-1	128, 3, 2, None
4	Bottleneck	2	-1	128
5	Convolucional	1	-1	256, 3 , 2, None
6	Bottleneck	8	-1	256
7	Convolucional	1	-1	512, 3 , 2, None
8	Bottleneck	8	-1	512
9	Convolucional	1	-1	1024, 3 , 2, None
10	Bottleneck	4	-1	1024
11	Bottleneck	1	-1	1024
12	Convolucional	1	-1	512, 1 , 1, None
13	Convolucional	1	-1	1024, 3 , 1, None
14	Convolucional	1	-1	512, 1 , 1, None
15	Convolucional	1	-1	1024, 3 , 1, None
16	Convolucional	1	-2	256, 3 , 1, None
17	Upsample	1	-1	2, nearest
18	Concat	1	-1,8	1
19	Bottleneck	1	-1	512
20	Bottleneck	1	-1	512
21	Convolucional	1	-1	256, 3 , 1, None
22	Convolucional	1	-1	512, 3 , 1, None
23	Convolucional	1	-2	128, 3 , 1, None
24	Upsample	1	-1	2, nearest
25	Concat	1	-1,6	1
26	Bottleneck	1	-1	256
27	Bottleneck	2	-1	256
28	Detect	1	27,22,15	-

Tabla 3.7: Arquitectura YOLOv4.

Número Módulo	Tipo Módulo	Número Repeticiones	Número entrada	Detalles
0	Convolucional	1	Imágenes base	32, 6 , 1, None
1	Convolucional	1	-1	64, 3 , 1, None
2	C3	1	-1	64,1
3	Convolucional	1	-1	128, 3, 2, None
4	C3	2	-1	128,2
5	Convolucional	1	-1	256, 3 , 2, None
6	C3	3	-1	256,3
7	Convolucional	1	-1	512, 3 , 2, None
8	C3	1	-1	512,1
9	SPPF	1	-1	512,5
10	Convolucional	1	-1	256, 3 , 2, None
10	Upsample	1	-1	2,nearest
11	Concat	1	-1,6	1
12	C3	1	-1	256, 1 , 1, None
13	Convolucional	1	-1	128, 3 , 1, None
14	Upsample	1	-1	2,nearest
15	Concat	1	-1,4	1
16	C3	1	-1	128, 1
17	Convolucional	1	-1	128, 3 , 2, None
18	Concat	1	-1,14	1
19	C3	1	-1	256,1
20	Convolucional	1	-1	256, 3 , 2, None
22	Concat	1	-1,10	1
22	C3	1	-1	512, 1
24	Detect	1	17,20,23	-

Tabla 3.8: Arquitectura YOLOv5s.

Número Módulo	Tipo Módulo	Número Repeticiones	Número entrada	Detalles
0	Convolucional	1	Imágenes base	48, 6 , 2, None
1	Convolucional	1	-1	96, 3 , 2, None
2	C3	2	-1	96,2
3	Convolucional	1	-1	192, 3, 2, None
4	C3	4	-1	192,4
5	Convolucional	1	-1	384, 3 , 2, None
6	C3	6	-1	384,6
7	Convolucional	1	-1	768, 3 , 2, None
8	C3	2	-1	768
9	SPPF	1	-1	768,5
10	Convolucional	1	-1	384, 1, 1, None
11	Upsample	1	-1	2,nearest
12	Concat	1	-1,6	1
13	C3	2	-1	384, 2
14	Convolucional	1	-1	192, 1 , 1, None
15	Upsample	1	-1	2,nearest
16	Concat	1	-1,4	1
17	C3	2	-1	192, 2
18	Convolucional	1	-1	192, 3 , 2, None
19	Concat	1	-1,14	1
20	C3	2	-1	384,2
21	Convolucional	1	-1	384, 3 , 2, None
22	Concat	1	-1,10	1
23	C3	2	-1	768, 2
24	Detect	1	17,20,23	-

Tabla 3.9: Arquitectura YOLOv5m.

Número Módulo	Tipo Módulo	Número Repeticiones	Número entrada	Detalles
0	Convolucional	1	Imágenes base	64, 6 , 2, None
1	Convolucional	1	-1	128, 3 , 2, None
2	C3	3	-1	128,3
3	Convolucional	1	-1	256, 3, 2, None
4	C3	6	-1	256,6
5	Convolucional	1	-1	512, 3 , 2, None
6	C3	9	-1	512,6
7	Convolucional	1	-1	1024, 3 , 2, None
8	C3	3	-1	1024,3
9	SPPF	1	-1	768,5
10	Convolucional	1	-1	512, 1, 1, None
11	Upsample	1	-1	2,nearest
12	Concat	1	-1,6	1
13	C3	3	-1	512, 3
14	Convolucional	1	-1	256, 1 , 1, None
15	Upsample	1	-1	2,nearest
16	Concat	1	-1,4	1
17	C3	3	-1	256, 3
18	Convolucional	1	-1	256, 3 , 2, None
19	Concat	1	-1,14	1
20	C3	3	-1	512,3
21	Convolucional	1	-1	512, 3 , 2, None
22	Concat	1	-1,10	1
23	C3	3	-1	1024, 2
24	Detect	1	17,20,23	-

Tabla 3.10: Arquitectura YOLOv5l.

Número Módulo	Tipo Módulo	Número Repeticiones	Número entrada	Detalles
0	Ghost Convolucional	1	Imágenes base	32, 6 , 1, None
1	Ghost Convolucional	1	-1	64, 3 , 1, None
2	Ghost C3	1	-1	64,1
3	Ghost Convolucional	1	-1	128, 3, 2, None
4	Ghost C3	2	-1	128,2
5	Ghost Convolucional	1	-1	256, 3 , 2, None
6	Ghost C3	3	-1	256,3
7	Ghost Convolucional	1	-1	512, 3 , 2, None
8	Ghost C3	1	-1	512,1
9	SPPF	1	-1	512,5
10	Ghost Convolucional	1	-1	256, 3 , 2, None
10	Upsample	1	-1	2,nearest
11	Concat	1	-1,6	1
12	Ghost C3	1	-1	256, 1 , 1, None
13	Ghost Convolucional	1	-1	128, 3 , 1, None
14	Upsample	1	-1	2,nearest
15	Ghost Concat	1	-1,4	1
16	Ghost C3	1	-1	128, 1
17	Ghost Convolucional	1	-1	128, 3 , 2, None
18	Concat	1	-1,14	1
19	Ghost C3	1	-1	256,1
20	Ghost Convolucional	1	-1	256, 3 , 2, None
22	Concat	1	-1,10	1
22	Ghost C3	1	-1	512, 1
24	Detect	1	17,20,23	-

Tabla 3.11: Arquitectura YOLOv5s-Ghost.

3.4. Modelos de red neuronal propuestos

Tomando en cuenta los resultados de los modelos seleccionados, se proponen dos arquitecturas basadas en YOLOv4. Estas se describen en las tablas 3.12 y 3.13. En la primera arquitectura se toma YOLOv4 manteniendo el Darknet53 sin modificar como la base y extendiendo la cabeza con 5 capas. Con esto se hace otro sobremuestreo y se concatena la salida del tercer cuello de botella del Darknet53, con lo que se logra obtener imágenes de 120×120 píxeles. El otro cambio que se hace es en la capa de detección. En lugar de tomar la salida de las capa 27, 22 y 15, se elimina la conexión de la capa 15 y se coloca la salida de la capa 32.

Ambos cambios se hacen tomando en cuenta las dimensiones de los objetos a detectar en función del tamaño de la imagen. Todas las muestras son de 480×640 píxeles y los objetos en promedio están al rededor de los 50×50 píxeles. Esto hace que al tener una grilla tan pequeña como la de la capa 15 varios cafetos puedan estar contenidos en una misma celda, imposibilitando la diferenciación entre estos. De este modo agregando una capa con una grilla mayor como la de la capa 32 y quitando la de la capa 15 se tiene mayor granularidad en todas las secciones de detección. A esta arquitectura se le llama *yolov4ext*

La segunda arquitectura también esta basada en la YOLOv4 y aplica los mismos cambios que en la arquitectura *yolov4ext*, pero cambiando todas las operaciones convolucionales por convoluciones fantasma, además de remplazar todos los módulos de cuello de botella por módulos C3 fantasmas, esto con la intención de mitigar los efectos de sobreajuste. A esta arquitectura se le llama *yolov4ext-ghost*.

Número Módulo	Tipo Módulo	Número Repeticiones	Número entrada	Detalles
0	Convolucional	1	-1	32, 3 , 1, None
1	Convolucional	1	-1	64, 3 , 1, None
2	Bottleneck	1	-1	64
3	Convolucional	1	-1	128, 3, 2, None
4	Bottleneck	2	-1	128
5	Convolucional	1	-1	256, 3 , 2, None
6	Bottleneck	8	-1	256
7	Convolucional	1	-1	512, 3 , 2, None
8	Bottleneck	8	-1	512
9	Convolucional	1	-1	1024, 3 , 2, None
10	Bottleneck	4	-1	1024
11	Bottleneck	1	-1	1024
12	Convolucional	1	-1	512, 1 , 1, None
13	Convolucional	1	-1	1024, 3 , 1, None
14	Convolucional	1	-1	512, 1 , 1, None
15	Convolucional	1	-1	1024, 3 , 1, None
16	Convolucional	1	-2	256, 3 , 1, None
17	Upsample	1	-1	2, nearest
18	Concat	1	-1,8	1
19	Bottleneck	1	-1	512
20	Bottleneck	1	-1	512
21	Convolucional	1	-1	256, 3 , 1, None
22	Convolucional	1	-1	512, 3 , 1, None
23	Convolucional	1	-2	128, 3 , 1, None
24	Upsample	1	-1	2, nearest
25	Concat	1	-1,6	1
26	Bottleneck	1	-1	256
27	Bottleneck	2	-1	256
28	Convolucional	1	-2	64, 3 , 1, None
29	Upsample	1	-1	2, nearest
30	Concat	1	-1,4	1
31	Bottleneck	2	-1	128
32	Bottleneck	1	-1	128
33	Detect	1	32,27,22	-

Tabla 3.12: Arquitectura yolov4ext.

Número Módulo	Tipo Módulo	Número Repeticiones	Número entrada	Detalles
0	Ghost Convolucional	1	-1	32, 3 , 1, None
1	Ghost Convolucional	1	-1	64, 3 , 1, None
2	Ghost C3	1	-1	64
3	Ghost Convolucional	1	-1	128, 3, 2, None
4	Ghost C3	2	-1	128
5	Ghost Convolucional	1	-1	256, 3 , 2, None
6	Ghost C3	8	-1	256
7	Ghost Convolucional	1	-1	512, 3 , 2, None
8	Ghost C3	8	-1	512
9	Ghost Convolucional	1	-1	1024, 3 , 2, None
10	Ghost C3	4	-1	1024
11	Ghost C3	1	-1	1024
12	Ghost Convolucional	1	-1	512, 1 , 1, None
13	Ghost Convolucional	1	-1	1024, 3 , 1, None
14	Ghost Convolucional	1	-1	512, 1 , 1, None
15	Ghost Convolucional	1	-1	1024, 3 , 1, None
16	Ghost Convolucional	1	-2	256, 3 , 1, None
17	Upsample	1	-1	2, nearest
18	Concat	1	-1,8	1
19	Ghost C3	1	-1	512
20	Ghost C3	1	-1	512
21	Ghost Convolucional	1	-1	256, 3 , 1, None
22	Ghost Convolucional	1	-1	512, 3 , 1, None
23	Ghost Convolucional	1	-2	128, 3 , 1, None
24	Upsample	1	-1	2, nearest
25	Concat	1	-1,6	1
26	Ghost C3	1	-1	256
27	Ghost C3	2	-1	256
28	Ghost Convolucional	1	-2	64, 3 , 1, None
29	Upsample	1	-1	2, nearest
30	Concat	1	-1,4	1
31	Ghost C3	2	-1	128
32	Ghost C3	1	-1	128
33	Detect	1	32,27,22	

Tabla 3.13: Arquitectura yolov4ext-ghost.

Capítulo 4

Resultados para la detección de cafetos con redes neuronales convolucionales

En este capítulo se detallan los resultados de la metodología descrita en el capítulo, iniciando por los resultados del análisis del conjunto de datos. Luego se presentan los resultados de los entrenamientos de las redes neuronales seleccionadas y finalmente los resultados de los modelos propuestos. En cada apartado se realiza una breve discusión de los resultados.

4.1. Análisis del conjunto de datos

El primer análisis se realiza sobre las etiquetas. Se usó del algoritmo de k-means para determinar las 3 combinaciones de la tupla anchos y largos más representativas de cada parcela de manera separada. La tabla 4.1 muestra los resultados de este experimento, indicando así los 3 grupos y los porcentajes de etiquetas que caen en ese grupo específico.

Las dimensiones de las etiquetas van desde los 29×51 píxeles, hasta los 79×162 píxeles. Las parcelas que presentan una semejanza entre ellas corresponden la de Acosta con la de Barva de Heredia, así como la de Grecia con Naranjo. Tres Ríos es la única que no se asemeja a las anteriores, debido a que sus etiquetas poseen dimensiones de mayor tamaño, en otras palabras, presenta los cafetos que abarcan la mayor cantidad de espacio en la imagen.

Con este experimento se da una primera impresión de las dimensiones las etiquetas y por ende, la cantidad de espacio que cada cafeto ocupa en las imágenes. No obstante, con este experimento no es posible cuantificar la diferencia entre las parcelas, por lo que se procede a hacer uso de la divergencia de Jensen-Shannon para determinar esta similitud. Esta divergencia cuantifica la diferencia entre dos distribuciones probabilísticas: un resultado de 0 indica distribuciones idénticas y un 1 corresponde a distribuciones mutuamente excluyentes.

Localización	**Ancho de etiqueta (Píxeles)**	**Largo de etiqueta (Píxeles)**	**Porcentaje** (%)
Acosta	29	51	41.7 %
	34	71	35.5 %
	54	92	22.7 %
Barva	29	50	41.4 %
	34	71	35.6 %
	54	90	22.9 %
Grecia	44	74	47.2 %
	36	54	29.6 %
	49	97	23.3 %
Naranjo	44	84	36.9 %
	32	54	36.4 %
	56	117	26.7 %
Tres Ríos	69	125	45.8 %
	60	88	26.5 %
	79	162	27.7 %

Tabla 4.1: Distribución de imágenes del set de datos según localización.

Por lo tanto, se procede a usar un modelo Gaussiano Mixto [36] para convertir las tuplas de largo y ancho de cada etiqueta en una distribución probabilística con 3 centros, siguiendo con lo establecido en el experimento anterior. Con las tuplas convertidas en distribuciones probabilísticas se proceden a calcular los valores de la divergencia Jensen-Shannon al comparar una parcela con otra. Los resultados de esta comparativa se muestran en la tabla 4.2.

Parcelas	**Acosta**	**Barva**	**Grecia**	**Naranjo**	**Tres Ríos**
Acosta	0.000	0.114	0.190	0.216	0.633
Barva	0.114	0.000	0.102	0.104	0.493
Grecia	0.190	0.102	0.000	0.081	0.471
Naranjo	0.215	0.104	0.081	0.000	0.357
Tres Ríos	0.633	0.493	0.471	0.357	0.000

Tabla 4.2: Comparación de Parcelas usando el método Jesen-Shannon usando las etiquetas como distribución probabilística.

Esto confirma que las parcelas de Barva, Acosta, Grecia y Naranjo presentan similitud entre ellas. Se muestra que las parcelas de Grecia y Naranjo son casi idénticas. Mientras que las etiquetas de Tres Ríos no comparten similitud con ninguna otra parcela, lo cual confirma todas las similitudes discutidas en el experimento anterior.

Otro dato que debe considerarse es la resolución espacial de las imágenes de cada parcela. Con esta característica se trata de establecer una correlación con los resultados del análisis de etiquetas que se ha realizado. No obstante, como se muestra en la tabla 4.3, las parcelas de Barva y Tres Ríos poseen casi la misma resolución, mientras que en la tabla 4.2 se muestra que estas dos parcelas no comparten similitud en lo que respecta al tamaño de los cafetos en las imágenes. Además, la resolución espacial de la parcela de Acosta presenta la mitad de resolución espacial que Barva, y en consecuencia, esta parcela cuenta con las imágenes con menor definición. A pesar de eso, las dimensiones de etiqueta no se separan a las de las otras parcelas.

Parcelas	**Acosta**	**Barva**	**Grecia**	**Naranjo**	**Tres Ríos**
Resolución (cm/píxel)	2.83	1.20	2.11	1.92	0.99

Tabla 4.3: Resolución espacial de las imágenes extraídas según la parcela.

El último experimento en el análisis de etiquetas consiste en establecer la densidad de cafetos por imagen. Para esta tarea se genera un histograma por localización de la cantidad de etiquetas que hay por imagen, presentando estos resultados desde la figura 4.1 a la 4.5. De este modo, las parcelas de Tres Ríos y de Barva de Heredia (figura 4.1 y figura 4.2, respectivamente) tienen las distribuciones de cafetos por imagen más estables. Se muestra que en Tres Ríos el 50 % de las imágenes poseen entre 10 y 15 cafetos por imagen, mientras que en Barva de Heredia la media está al rededor de los 20 y 25 cafetos por imagen.

Por otra parte, la figura 4.3 muestra el histograma de la cantidad de cafetos por imagen para la parcela de Grecia, que presenta mayor diferencia de cantidad de cafetos por imagen, entre 10 y 45. Finalmente, los histogramas de las parcelas Naranjo y Acosta (figuras 4.4 y 4.5) tienen un comportamiento similar, presentando un mayor porcentaje de imágenes entre 5 y 25 cafetos por imagen. Otra característica de estas 3 últimas parcelas, es que son las únicas que poseen imágenes con más de 50 cafetos por imagen.

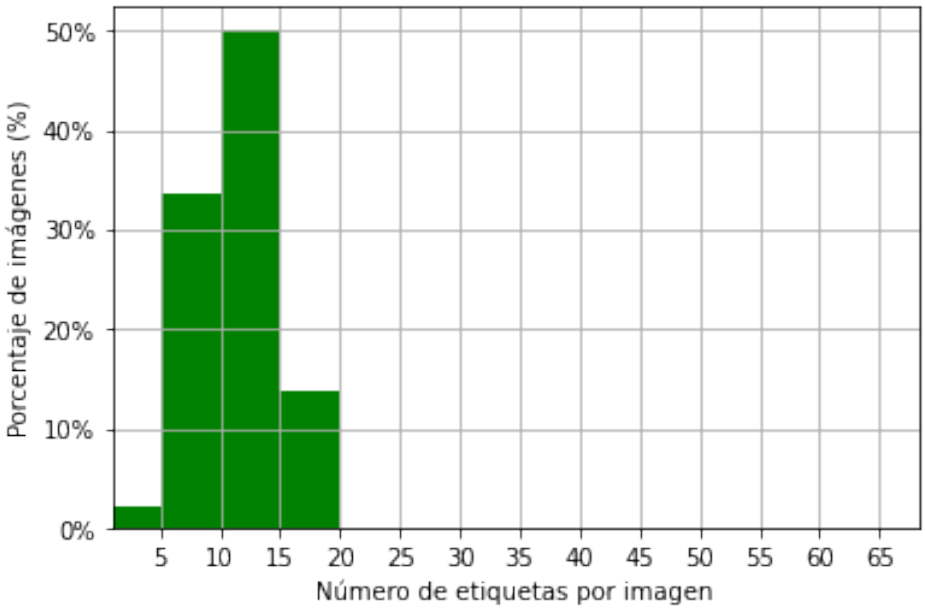


Figura 4.1: Histograma de cantidad de etiquetas por imagen en la parcela de Tres Ríos.

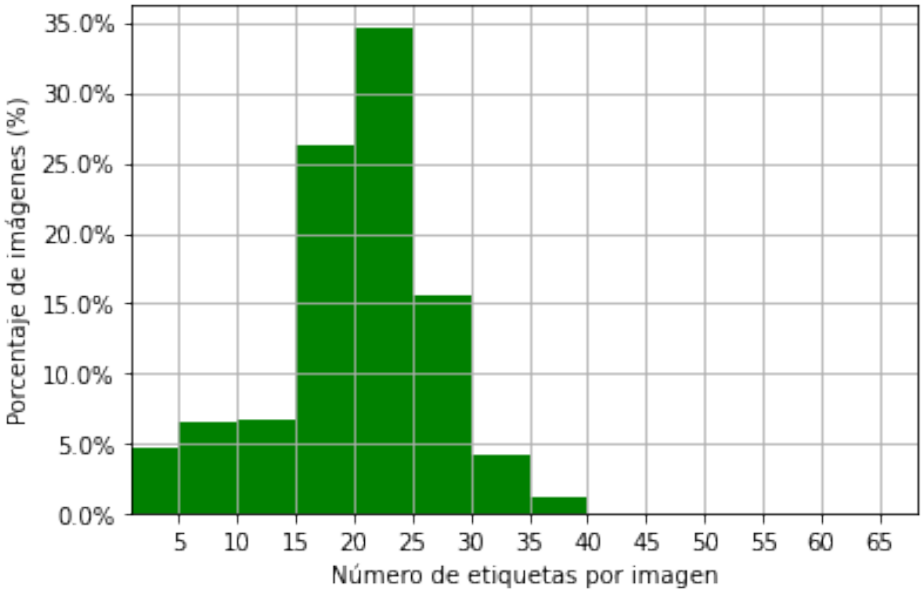


Figura 4.2: Histograma de cantidad de etiquetas por imagen en la parcela de Barva de Heredia.

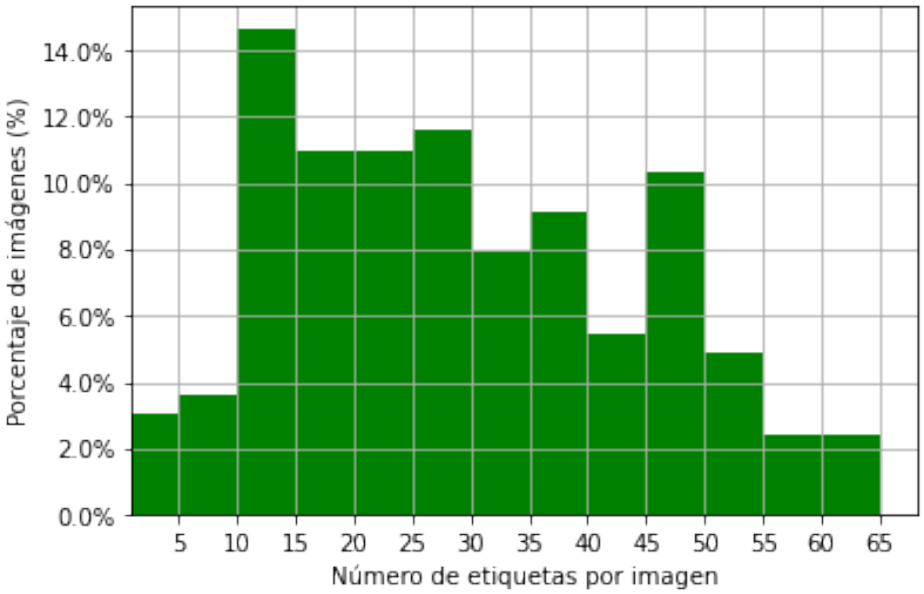


Figura 4.3: Histograma de cantidad de etiquetas por imagen en la parcela de Grecia.

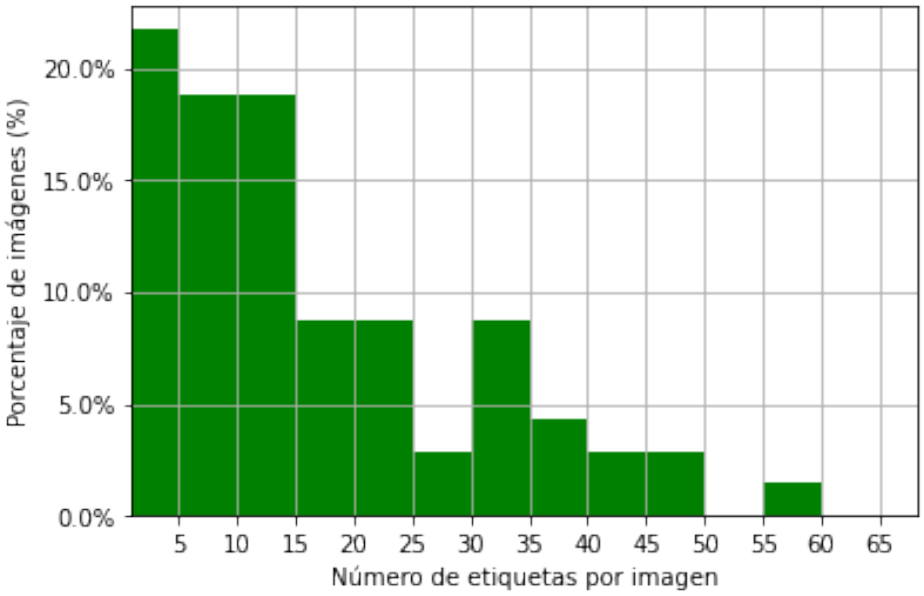


Figura 4.4: Histograma de cantidad de etiquetas por imagen en la parcela de Acosta.

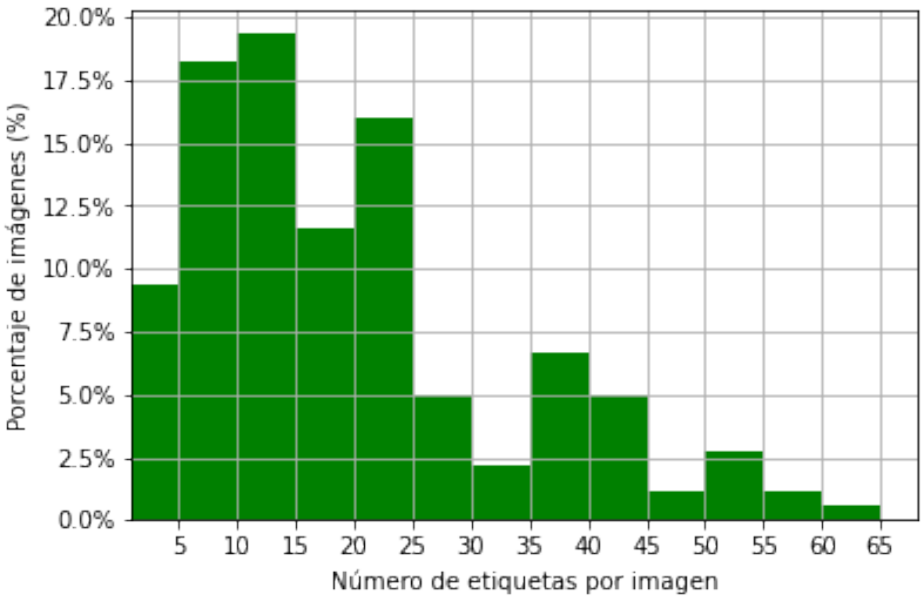


Figura 4.5: Histograma de cantidad de etiquetas por imagen en la parcela de Naranjo.

Una vez terminado el análisis de las etiquetas, se realiza un análisis de los colores de las imágenes. Para esto, se elaboraron diferentes histogramas RGB mostrando la cantidad de píxeles según la intensidad del color, dividiéndolo por canal. Esto da una idea de los colores dominantes, pero es una información incompleta, ya que no muestra la relación entre los tres canales. De esta manera, se establece esta relación entre los canales RGB y obtienen los colores predominantes de cada grupo de imágenes según localización empleando el algoritmo de k-means. Se usa este algoritmo para encontrar los 10 colores dominantes, y el porcentaje de píxeles que se acercan más a este. Los resultados de estos dos experimentos se muestran de la figura 4.6 a la 4.10.

Los resultados en las figuras 4.6, corresponden a la gama de colores verde provenientes de los cafetos y demás vegetación. Existe cerca de un 20 % de colores de la gama de colores marrón provenientes de la tierra que rodea a los cafetos. Esta característica la comparte la parcela de Tres Ríos, con un 29 % de colores de esa gama de colores, como se muestra en la figura 4.7.

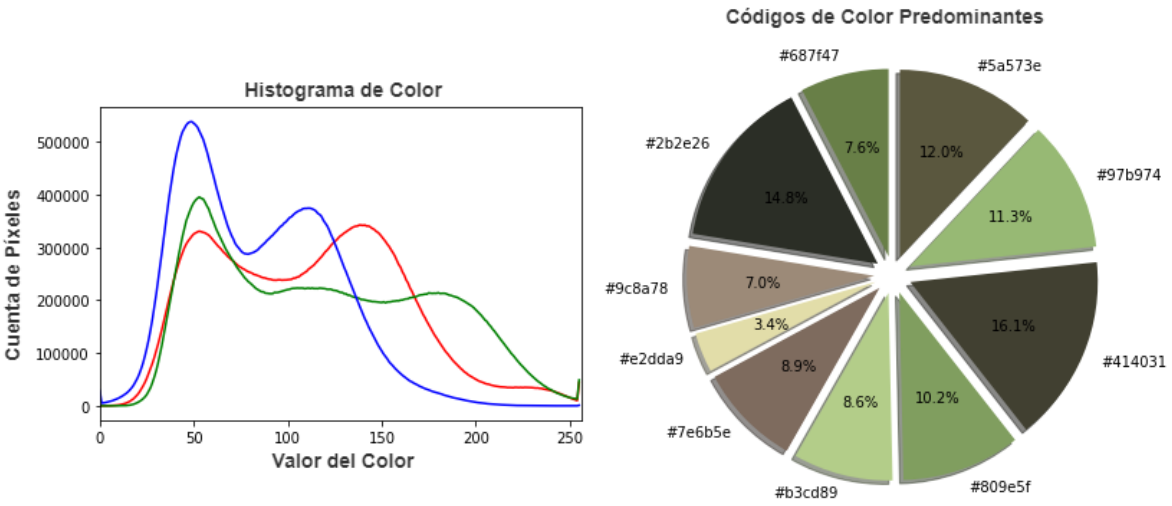


Figura 4.6: Histograma de color de las etiquetas de la parcela de Barva de Heredia.

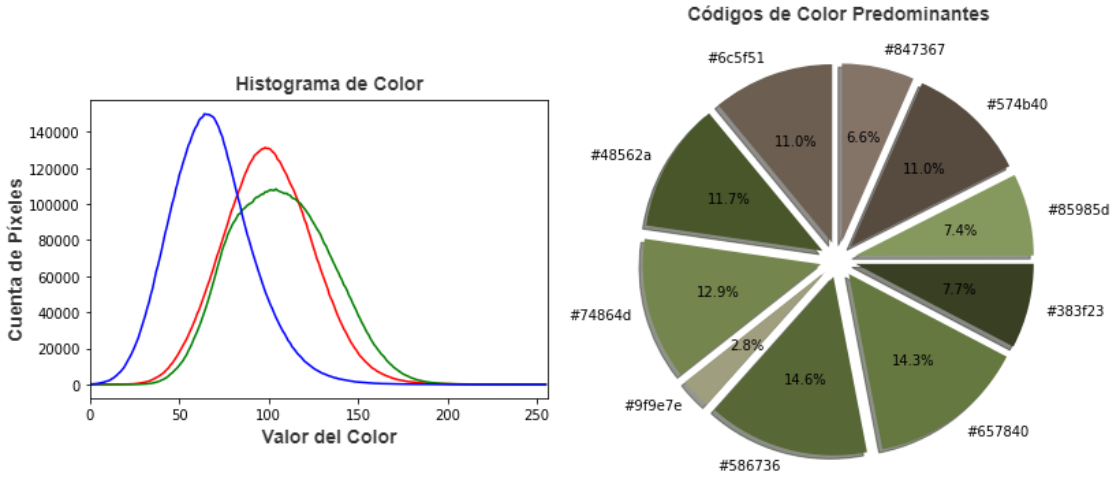


Figura 4.7: Histograma de color de las etiquetas de la parcela de Tres Ríos.

Por otra parte, las figuras 4.8, 4.9 y 4.10 corresponden a los resultados de las imágenes de las parcelas de Acosta, Grecia y Naranjo, respectivamente. En estas parcelas todos los análisis de color usando k-means presenta únicamente gama de color verde y en tonalidades similares entre sí. Esta similitud se muestra también en los histogramas donde se mostró una distribución semejante y se presentó la media de cada color en una intensidad similar. Las etiquetas de Grecia son las que presentan el mayor porcentaje de colores cercanos al negro, esto debido a que los ortomosaicos de esta parcela contienen una cantidad superior de sombras.

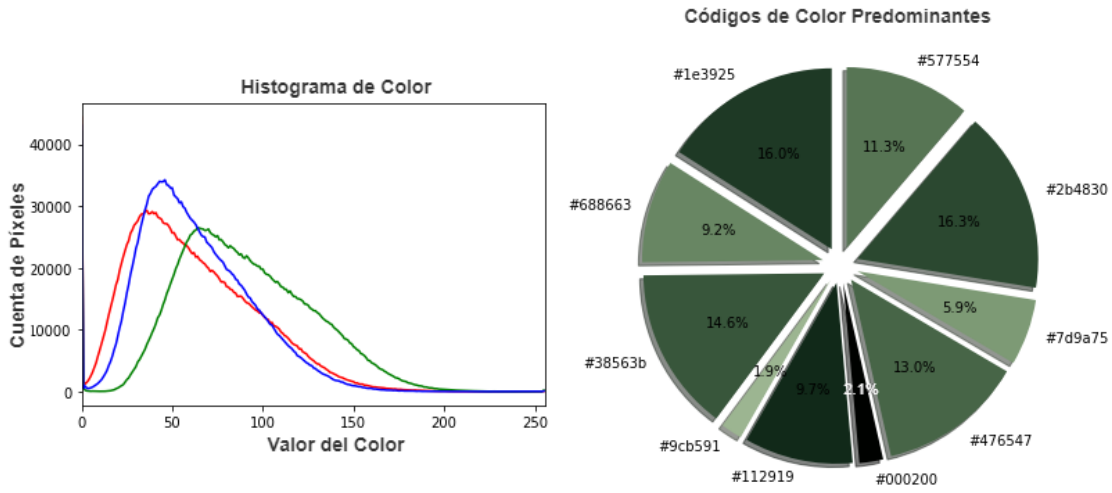


Figura 4.8: Histograma de color de las etiquetas de la parcela de Acosta.

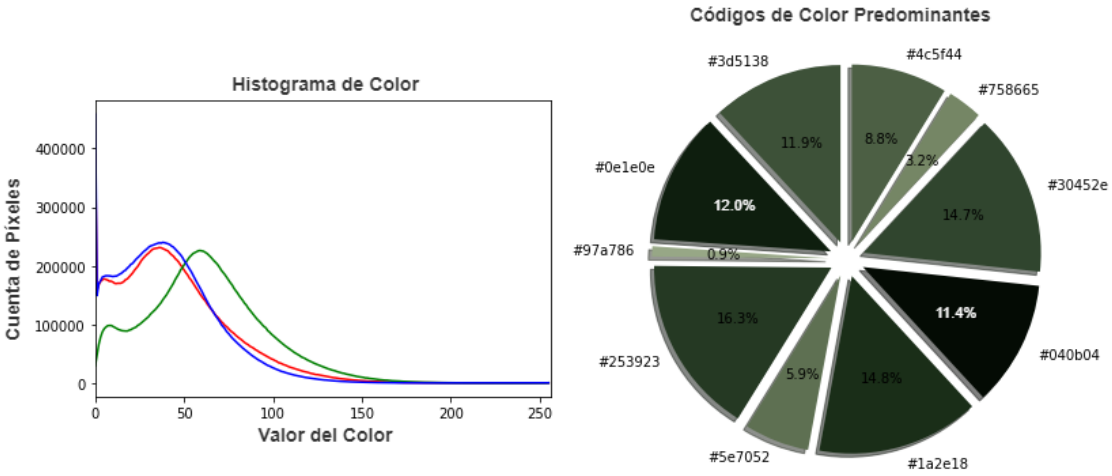


Figura 4.9: Histograma de color de las etiquetas de la parcela de Grecia.

Las paletas de color en conjunto con los histogramas por canal RGB permiten evaluar diferencias y semejanzas entre parcelas de forma visual, sin embargo, al igual que en el caso de las etiquetas, es necesario cuantificar esta similitud y se opta por usar nuevamente la divergencia Jensen-Shannon. De esta manera, se hace uso de un modelo Gaussiano Mixto [36] con 10 centros para convertir las imágenes de cada parcela en una distribución probabilística siendo las variables aleatorias cada píxel con sus tres canales RGB.

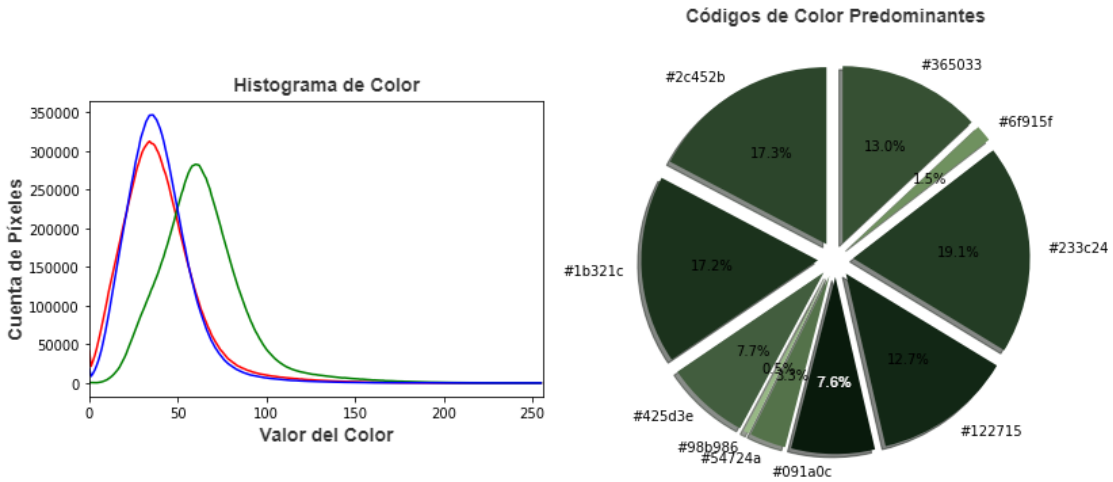


Figura 4.10: Histograma de color de las etiquetas de la parcela de Naranjo.

Con las imágenes convertidas en distribuciones probabilísticas se calculan todos los valores de divergencia Jensen-Shannon entre parcelas. Todos los valores de divergencia se muestran en la tabla 4.4. Estos resultados confirman la similitud entre las parcelas Acosta, Grecia y Naranjo, e indican una semejanza entre las parcelas de Barva de Heredia y Tres Ríos. Esta última parcela presenta la mayor divergencia con las demás.

Parcelas	**Acosta**	**Barva**	**Grecia**	**Naranjo**	**Tres Ríos**
Acosta	0.000	0.445	0.207	0.210	0.618
Barva	0.508	0.000	0.446	0.539	0.255
Grecia	0.207	0.446	0.000	0.170	0.532
Naranjo	0.210	0.539	0.170	0.000	0.639
Tres Ríos	0.618	0.255	0.532	0.639	0.000

Tabla 4.4: Comparación de Parcelas usando el método Jesen-Shannon usando los colores como criterio de comparación

De este modo, los aspectos más destacables del análisis recaen en que la parcela de Tres Ríos es la única parcela que no presenta similitud en lo que respecta a dimensiones de etiqueta y además presenta la menor densidad de cafetos por imagen. Se identifica que la parcela de Acosta es la que presenta la menor resolución espacial y ,finalmente, que las parcelas de Acosta, Grecia y Naranjo poseen una distribución de color semejante.

4.2. Entrenamiento de redes neuronales

Con las arquitecturas ya descritas y siguiendo con la tabla de permutaciones (tabla 4.7), el primer experimento que se llevó a cabo fue entrenar durante 28 épocas todas las

arquitecturas, usando el 10 % de cada parcela como conjunto de prueba (permutación número 6). Los resultados de consumo durante el entrenamiento se detallan en la tabla 4.5, siendo la arquitectura YOLOv4 la que presenta el mayor uso de memoria en GPU, mientras que la YOLOv5-Ghost es la que se ejecuta de manera más rápida y utilizando el menor porcentaje de GPU. Esto concuerda con lo mostrado en la tabla 3.5 en lo que respecta a mayor y menor número de parámetros y GFLOPs.

Arquitectura Arquitectura	**Tiempo de entrenamiento (min)**	**Uso medio de GPU(%)**	**Uso medio de CPU (%)**
YOLOv4	1435	51.65	14.33
YOLOv4-tiny	219	18.72	25.81
YOLOv5s	303	19.99	20.05
YOLOv5m	721	30.01	16.92
YOLOv5l	1080	44.23	15.15
YOLOV5s-ghost	305	18	21.47

Tabla 4.5: Características Hardware.

Los modelos en cada época fueron evaluados con el conjunto de prueba y las métricas obtenidas en este proceso se muestran en las figuras 4.11, 4.12 y 4.13. De estos resultados, la tabla 4.6 recopila los mejores resultados de cada arquitectura, donde se reporta el menor error en la función de costo. Los resultados de todas las arquitecturas, exceptuando los de YOLOv4-Tiny, son similares entre si, pero es el modelo YOLOv4 el que sobresale, llegando a un 91.05 % de exhaustividad.

Durante el entrenamiento, para poder determinar si se cae o no en sobreajuste, en cada época se registra el valor de la función de pérdida evaluada con el set de prueba. Siguiendo con la función de costos de YOLO. Se recopila el valor de cada uno de los tres grupos de sumatorias: pérdida por detección de objeto, pérdida por caja delimitadora y pérdida por clase detectada. No obstante, esta última siempre es 0 debido a que solo se tiene una clase.

Los resultados de la función de costo evaluada con el set de prueba se muestran en la figura 4.14 y 4.15. En estas gráficas no se incluye la arquitectura YOLOv4-Tiny debido a que su error era mucho mayor al resto y no permitía ver el comportamiento de los demás modelos. Todas las arquitecturas de red neuronal inician con una tendencia de error decreciente y al llegar a la época 15 el error empiezan con una tendencia ascendente, lo que indica el inicio de un sobreajuste.

La única arquitectura que no sufre de este comportamiento es la YOLOv5-Ghost, cuyo error decrece a una velocidad menor que sus contra partes, pero nunca entra en sobreajuste. Este análisis se utiliza más adelante en este capítulo para justificar los modelos de red neuronal que se usaron como base para las arquitecturas propuestas.

Arquitectura	Exhaustividad (%)	Precisión (%)	mAP50 (%)
YOLOv4	91.05	90.80	93.86
YOLOv5l	90.09	90.48	91.90
YOLOv5m	90.74	90.11	92.59
YOLOV5s-ghost	90.72	90.60	93.87
YOLOv5s	90.07	90.48	93.61
YOLOv4-tiny	88.43	90.13	92.89

Tabla 4.6: Mejores resultados de cada arquitectura para la permutación 6.

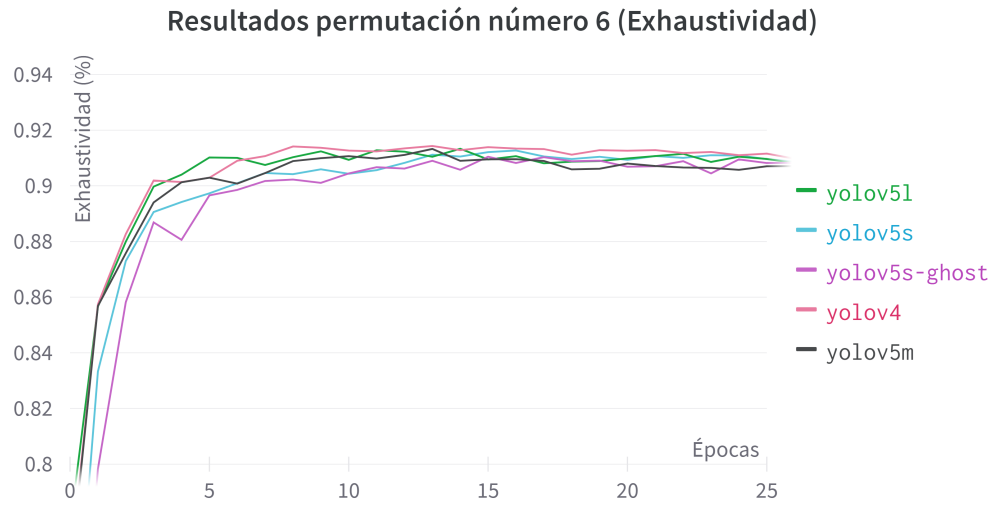


Figura 4.11: Resultados de exhaustividad para todas las arquitecturas entrenadas con la permutación 6, evaluando modelos con el conjunto de prueba.

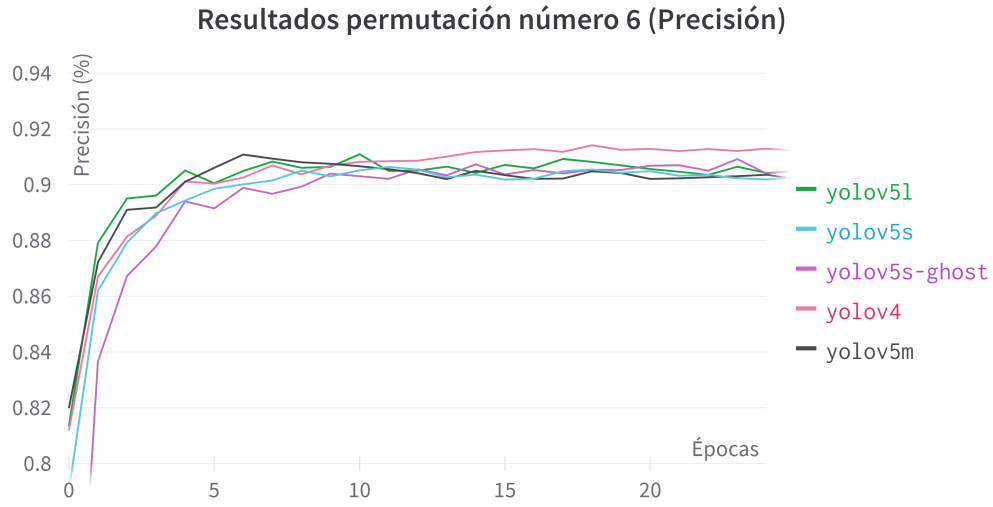


Figura 4.12: Resultados de precisión para todas las arquitecturas entrenadas con la permutación 6, evaluando modelos con el conjunto de prueba.

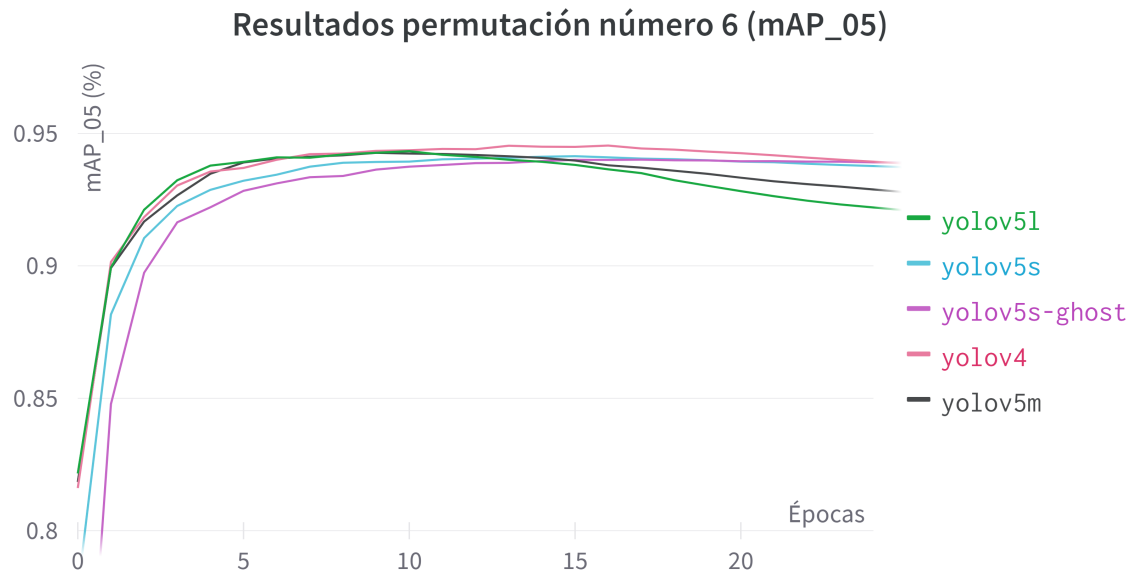


Figura 4.13: Resultados de mAP0.5 para todas las arquitecturas entrenadas con la permutación 6, evaluando modelos con el conjunto de prueba.

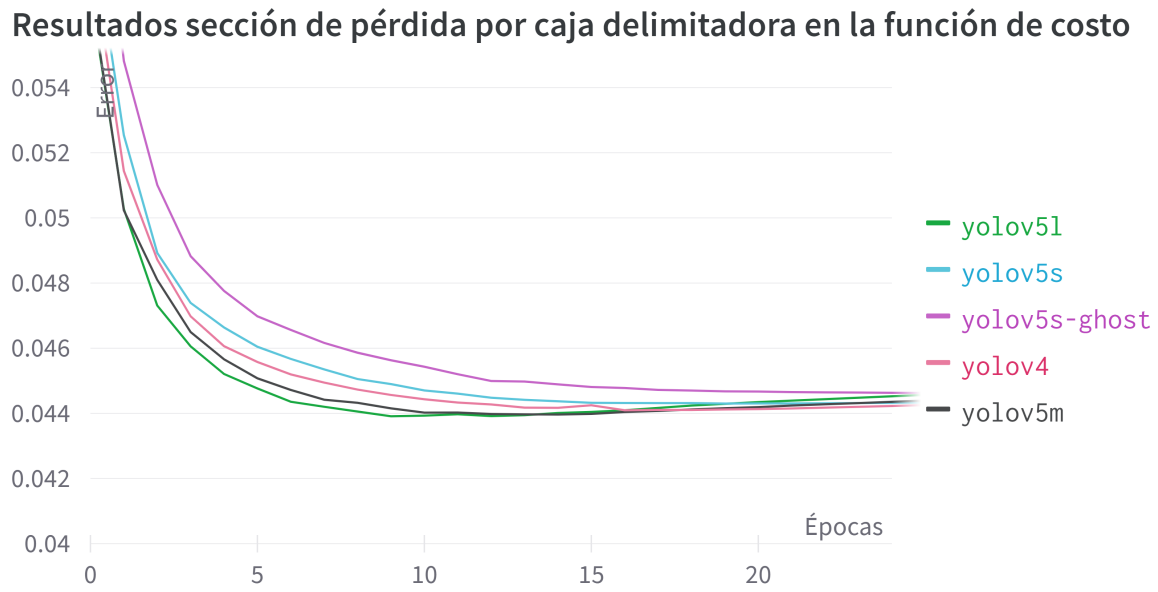


Figura 4.14: Valor de error por cajas delimitadoras evaluando el set de prueba.

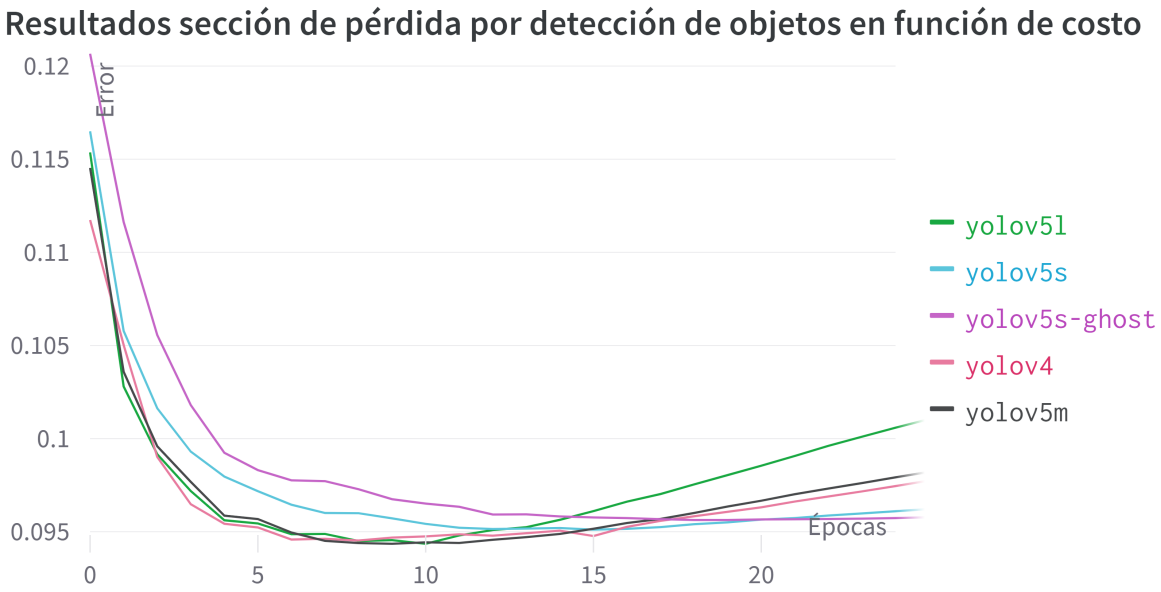


Figura 4.15: Valor de error por detección de objetos evaluando el set de prueba.

4.3. Evaluación de modelo propuesto

Tomando en cuenta estos resultados de los modelos seleccionados se proponen dos arquitecturas basadas en la YOLOv4 llamadas yolov4ext y yolov4ext-ghost. Ambas se entrenan usando la permutación 6 (10 % de cada localización) y al igual que en los entrenamientos anteriores, se registra el consumo de recursos durante el entrenamiento.

Ambas redes presentan un consumo del 60 % de la memoria de la GPU a disposición y tardan 23 y 20 horas en alcanzar las 30 épocas, respectivamente. Los resultados durante los entrenamientos se muestran en las figuras 4.16, 4.17 y 4.18. Ambas arquitecturas superan a la YOLOv4 en todas las métricas exceptuando en la precisión. Al igual que en los entrenamientos anteriores se registran los valores de las métricas cuando se alcanza el menor error de la función de costo y se reportan estos en la tabla 4.8.

Finalmente, se hace la misma revisión que con las arquitecturas anteriores para determinar si hay o no sobreajuste y si se da, registrar a partir de que época ocurre. Los resultados se muestran en las figuras 4.19 y 4.20, obteniendo un menor error en ambos casos. En la figura 4.20 no aparecen los resultados de la arquitectura YOLOv4 para que la escala permita apreciar el punto en el que la arquitectura yolov4-ext entra en sobreajuste, fenómeno que aparece a partir de la época 15 al igual que los primeros modelos entrenados. Por otra parte, la arquitectura yolov4-est-ghost nunca entró en sobreajuste.

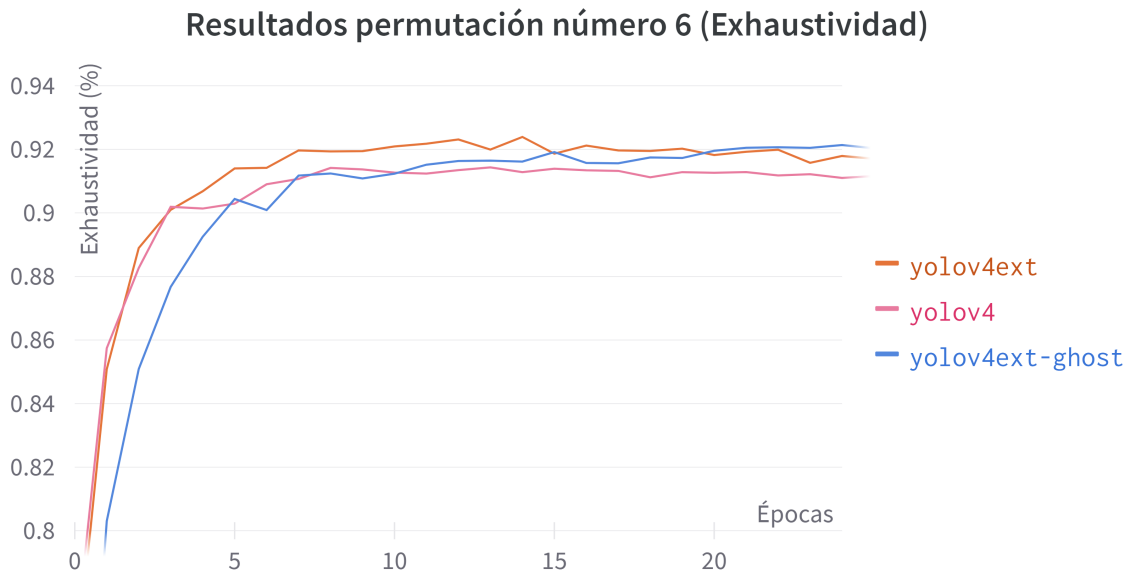


Figura 4.16: Resultados de exhaustividad para todas las arquitecturas entrenadas con la permutación 6, evaluando modelos con el conjunto de prueba.

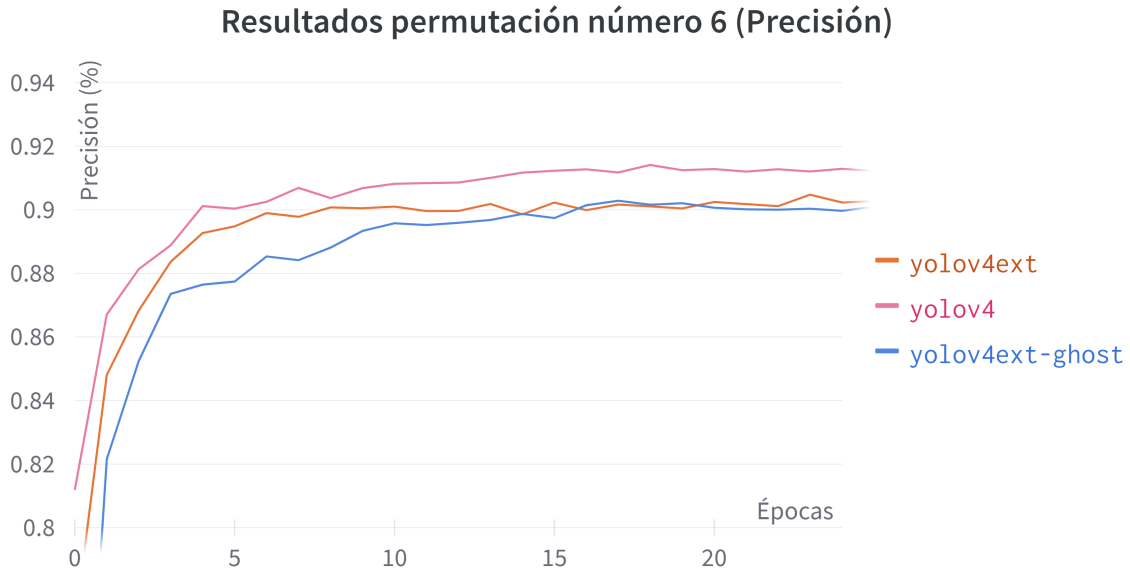


Figura 4.17: Resultados de precisión para todas las arquitecturas entrenadas con la permutación 6, evaluando modelos con el conjunto de prueba.

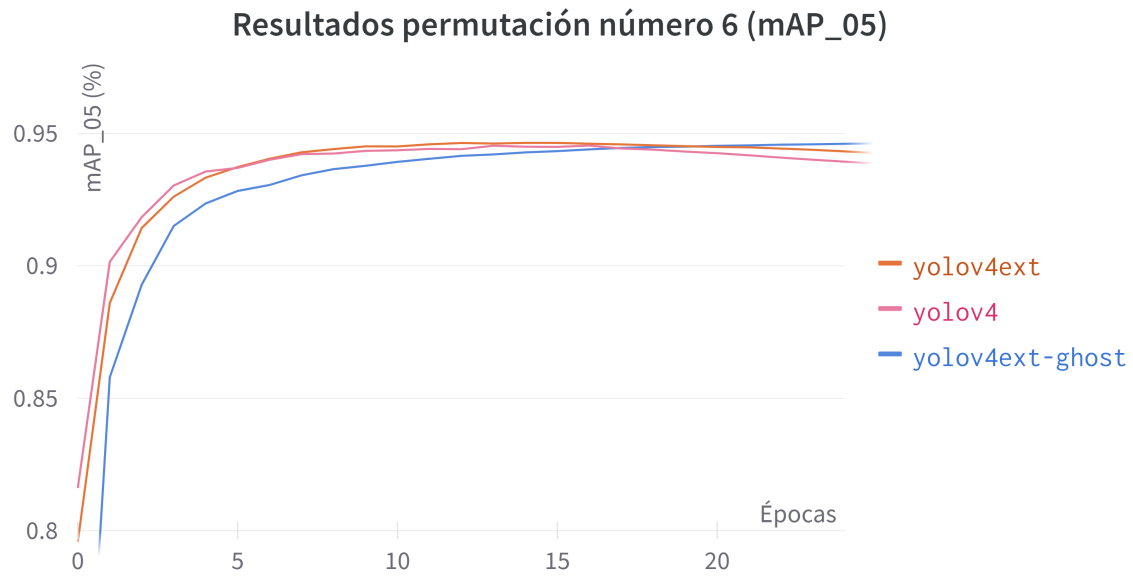


Figura 4.18: Resultados de mAP0.5 para todas las arquitecturas entrenadas con la permutación 6, evaluando modelos con el conjunto de prueba.

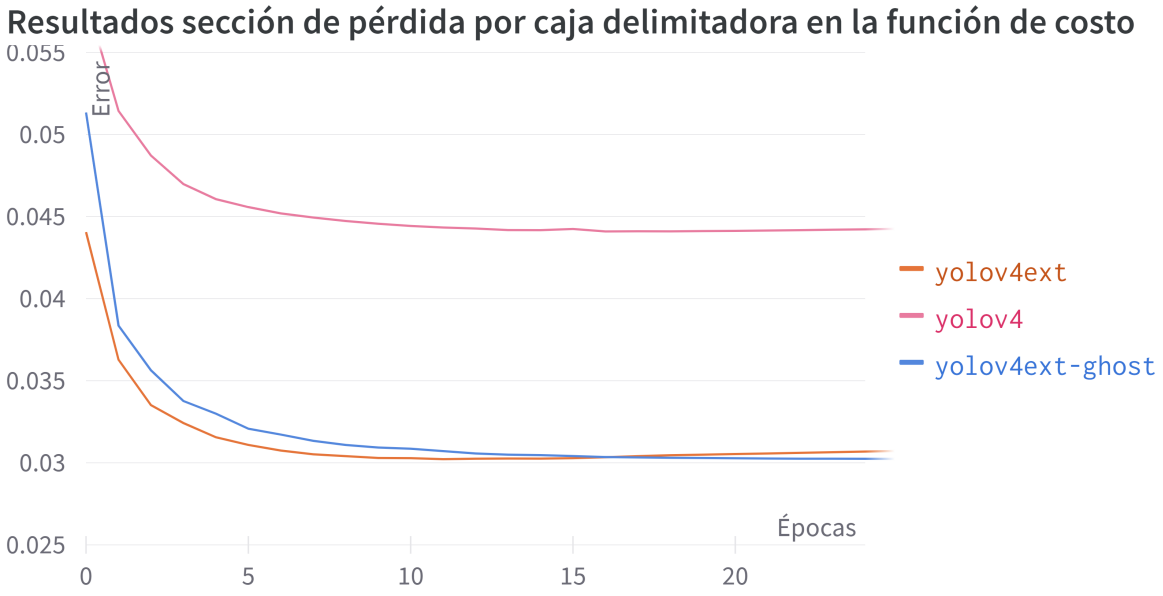


Figura 4.19: Valor de error por cajas delimitadoras evaluando el conjunto de prueba.

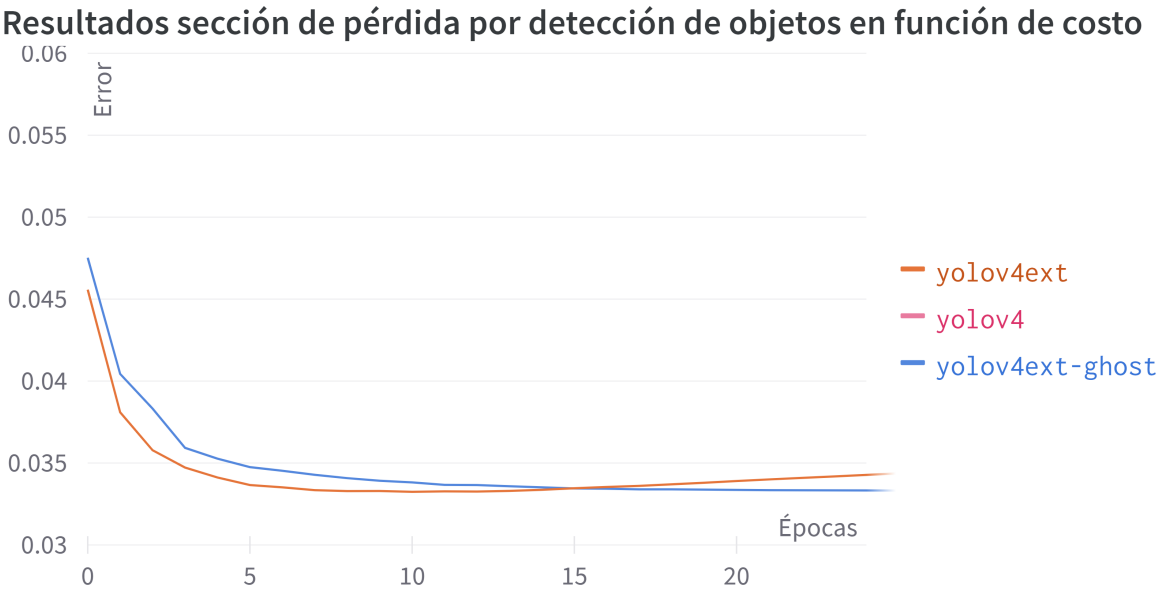


Figura 4.20: Valor de error por detección de objetos evaluando el conjunto de prueba.

Las arquitecturas alcanzan una detección de más del 90 % y se logra un menor error en las funciones de pérdida que los modelos entrenados en el primer experimento. Además, se encuentra una arquitectura que posee estos resultados y además, no cae en sobreajuste en todo el entrenamiento. El siguiente paso es determinar si se está cayendo en sesgo. Para esto, en los siguientes experimentos se entrenan desde cero nuevamente las 2 arquitecturas propuestas pero con las otras permutaciones explicadas en la tabla 4.7. El objetivo es determinar si la red puede generalizar e identificar cafetos en plantaciones nunca antes vistas, cuáles parcelas se le facilitan más a la red según las parcelas con las que se le alimenta durante el entrenamiento.

El resumen de los resultados de todos los entrenamientos del experimento de sesgo se muestra en la tabla 4.8. La permutación 1 usa las imágenes de la parcela de Acosta como set de prueba. Se destaca como la arquitectura Yolov4ext supera los resultados de su predecesora la yolov4 aunque no logra superar el 62 % de detección, con el cual se infiere que las parcelas restantes no logran generalizar la plantación de Acosta.

Este comportamiento es similar en la permutación 2 donde se usa la parcela de Barva como set de prueba, resaltando que la arquitectura Yolov4ext-ghost tiene resultados mejores llegando a casi el 70 % de detección. Se observa que en este experimento en particular el set de prueba es de mayor tamaño que el de entrenamiento y este comportamiento se puede deber a la reducida cantidad de imágenes a disposición para alimentar las redes durante el entrenamiento.

Por el contrario, las permutaciones 3 y 4 que usan las parcelas de Grecia y Naranjo como set de prueba logran una detección por encima del 80 %. Por tanto, se puede afirmar que estas dos parcelas si logran ser generalizadas con los datos a disposición. Estas dos parcelas son parecidas entre ellas tanto a nivel de color, densidad de cafetos por imagen, dimensiones de cajas delimitadoras, nivel de ruido en la plantación y traslape

entre cafetos.

La permutación 5 por su parte tiene resultados casi iguales a los obtenidos en la permutación 1. En esta prueba, las imágenes de la parcela de Tres Ríos son empleadas como set de prueba. La arquitectura propuestas Yolov4ext es la que logra mejores resultados de detección (exhaustividad) superando a su predecesora la Yolov4. Mientras que Yolov4ext-ghost apenas logra superar el 50 % de detecciones. Todos los resultados son bajos, demostrando que la generalización de la parcela de Tres Ríos no es posible usando las otras para alimentar la red.

Número de permutación	**Localización empleada como set de prueba**	**Localización empleada como set de entrenamiento**
1	Acosta	Todas las parcelas restantes
2	Barva de Heredia	Todas las parcelas restantes
3	Grecia	Todas las parcelas restantes
4	Naranjo	Todas las parcelas restantes
5	Tres Ríos	Todas las parcelas restantes
6	10 % de cada localización	90 % de cada localización

Tabla 4.7: Permutación de localizaciones para establecer el set de prueba.

Permutación	**Arquitectura**	**exhaustividad**	**Precisión**	**mAP@50**
Permutación 1	Yolov4ext	60.97	65.90	63.76
	Yolov4ext-ghost	58.94	66.32	62.52
	Yolov4	48.65	71.32	55.81
Permutación 2	Yolov4ext	61.57	70.41	65.60
	Yolov4ext-ghost	67.01	69.84	70.26
	Yolov4	61.47	75.49	67.50
Permutación 3	Yolov4ext	77.11	81.82	81.74
	Yolov4ext-ghost	78.51	81.00	83.99
	Yolov4	75.90	85.91	81.86
Permutación 4	Yolov4ext	80.10	74.41	79.59
	Yolov4ext-ghost	80.68	73.82	78.76
	Yolov4	80.79	73.82	78.76
Permutación 5	Yolov4ext	61.66	63.15	64.38
	Yolov4ext-ghost	53.95	67.74	59.83
	Yolov4	59.10	66.83	64.69
Permutación 6	Yolov4ext	91.93	90.28	94.41
	Yolov4ext-ghost	92.05	90.10	94.59
	Yolov4	91.05	90.80	93.86

Tabla 4.8: Mejores Resultados entrenamiento con todas las permutaciones.

Condensando los resultados anteriormente mencionados, se determina que las parcelas de Grecia, Naranjo y Barva tienden a presentar mejores resultados usando otras parcelas como alimentación, mientras que las parcelas de de Acosta y Tres Ríos son las que presentan la peor detección, con valores cercanos al 50 %. Con esto surgen dos preguntas: ¿Es posible generalizar la parcela de Naranjo usando únicamente la parcela de Grecia? ¿De alguna manera las parcelas de Grecia, Acosta y Naranjo son tan distintas a la parcela de Tres Ríos que hacen que la generalización que la red realiza no permita la detección de los cafetos en estas imágenes?

Para verificar estas premisas se confeccionan los bloques de entrenamiento descritos en la tabla 3.4. El primer bloque usa las parcelas de Acosta y Grecia para alimentar la red y emplea la parcela de Naranjo como prueba. El bloque 2 usa la parcela de Barva para entrenar y la de Tres Ríos como prueba.

Se destaca que la parcela de Tres Ríos realmente no comparte características con la de Barva. Tienen una paleta de colores, densidad de cafetos por imagen y un tamaño promedio de cajas delimitadoras distintos. La razón por la que se empareja la parcela de Barva con la de Tres Ríos es únicamente porque el nivel de diferencia no es tan grande en comparación con las otras tres parcelas.

Los resultados de estos experimentos se muestra en la tabla 4.9. Con esto vemos que las parcelas de Grecia y Acosta por sí solas, aún con la baja cantidad de imágenes que estas presentan, siguen alcanzando una detección de alrededor del 80 % en la parcela de Naranjo. Este resultado en conjunto con lo encontrado en el cálculo de la divergencia Jensen-Shannon descrita en las tablas 4.4 y 4.2, da la inferencia que la similitud de las parcelas de entrenamiento cobra mayor relevancia que el número de imágenes.

Esto concuerda con los resultados del experimento entrenando con la parcela de Barva y validando con la de Tres Ríos. La primera mencionada posee más imágenes que todas las parcelas juntas y aún así no logra generalizar la parcela de Tres Ríos, con la que comparte mayor semejanza en lo que respecta a distribución de colores según los resultados de la divergencia Jensen-Shannon de la tabla 4.4; no obstante, no comparte semejanza en lo que respecta a las dimensiones de etiqueta, como se muestra en la tabla 4.2.

Con esto surge otra duda más: ¿Que tanto se mejorarían las métricas si se elimina la parcela de Tres Ríos? por lo que se opta por modificar la permutación 6 y eliminar esta parcela tanto del entrenamiento como de la prueba. En consecuencia se obtiene una mejora en la precisión de ambas redes propuestas subiendo de un 90 % a un 92 % en ambos casos y las demás métricas se mantienen sin cambios.

De este modo se llega a la conclusión de que es necesario agregar más parcelas que presenten distribuciones de color distintas. Esto puede ser mitigado mediante un aumentado de datos si se tiene conocimiento de la distribución de color que se tiene en otras plantaciones. Además de esto, es necesario en tener datos que presenten cafetos que abarquen mayor parte de las imágenes, para conseguir diferentes tamaños de caja delimitadora al momento del etiquetado.

Entren.	Prueba	Arquitectura	exhaustividad	Precisión	mAP50
Grecia Acosta	Naranjo	Yolov4ext	77.16	81.68	83.35
		Yolov4ext-ghost	77.17	79.51	83.30
Barva	Tres Ríos	Yolov4ext	44.98	52.10	44.42
		Yolov4ext-ghost	55.65	56.47	55.79

Tabla 4.9: Resultados de entrenamiento solo con parcelas similares.

Capítulo 5

Conclusiones

En este trabajo se logró proponer una metodología a seguir para determinar la arquitectura de red neuronal óptima para detección de cafetos en plantaciones usando imágenes tomadas con drones desde una vista superior. Dicha metodología incluye la recolección, etiquetado y análisis de las imágenes usadas para alimentar las redes neuronales, así como una propuesta de arquitectura basada en YOLO.

El análisis de las imágenes incluye determinación de paleta de colores según localización, densidad de cafetos por imagen y dimensiones de los mismos. Además, se propuso el uso de la divergencia de Jensen-Shannon para cuantificar la similitud entre los datos de las parcelas, convirtiendo las imágenes y etiquetas en distribuciones probabilísticas usando un modelo Gaussiano mixto.

Mediante la experimentación se determina que la arquitectura YOLOv4 es la que presenta las mejores métricas de detección de cafetos, con un 91.05 % de exhaustividad. A partir de este resultado se propone la arquitectura Yolov4ext basada en la anteriormente mencionada alcanzando el 92 % de exhaustividad y un mAP con IOU de umbral de 0.5 (mAP@0.5) del 94.41 %, valor por encima de la media de mAP del 80 % [37][38].

También se propone la arquitectura Yolov4ext-ghost, que consiste en cambiar las capas convolucionales por capas convolucionales fantasma. Esta arquitectura mitiga los efectos del sobreajuste y mantiene los valores de métricas de su predecesora, la Yolov4ext.

Además de esto se realiza la experimentación necesaria para determinar el peso de cada parcela en el entrenamiento de las redes propuestas, y se encuentra que algunas parcelas no pueden ser generalizadas al entrenar los modelos con las demás.

Este comportamiento se atribuye a conjuntos de datos con menores resoluciones espaciales, en conjunto a una densidad de arboles y maleza que envuelve la plantación y dificulta la diferenciación de los cafetos. Otras razones recaen en diferencias en densidad de cafetos por imagen, distribución de colores y dimensiones por etiquetas. Todas estas diferencias pueden ser demostradas cuantitativamente con el análisis con la divergencia de Jensen-Shannon.

De modo contrario, se estable que para las parcelas que comparten características de tamaño de etiqueta, distribución de color, resolución espacial y densidad de cafetos por imagen se alcanza un 80 % de exhaustividad a pesar de una reducida cantidad de muestras disponibles.

Aunque los resultados obtenidos son prometedores, en síntesis, para futuros trabajos lo que se necesita primordialmente es expandir el set de entrenamiento hasta alcanzar una cantidad de parcelas que abarque la mayor cantidad de posibilidades escenarios que tenga una plantación cafetalera. En otras palabras se sugiere agregar a la base de datos plantaciones que no sean semejantes a las ya existentes, determinando este parentesco con la divergencia Jensen-Shannon tanto para color como para dimensiones de etiqueta.

De igual manera se sugiere tener datos a distintas resoluciones espaciales. En este proyecto no se logra encontrar una correlación pero se espera que esta característica tenga un rol importante en un set de datos más extenso. Además, se sugiere evitar utilizar plantaciones con entornos ruidosos ya que los resultados indican que este tipo de ambientes no se logran generalizar con las arquitecturas YOLO.

Se espera que aumentar el set de datos y dando un seguimiento a diferentes parcelas, similar a como se hizo con la parcela de Barba de Heredia, los resultados de los experimentos de sesgo mejoren las métricas para las arquitecturas propuestas.

Bibliografía

[1] C. Geiler, "El aroma del café costarricense despierta al mundo," Delfino CR, 2021. https://delfino.cr/2021/01/el-aroma-del-cafe-costarricense-despierta-al-mundo.

[2] M. Barquero, "Cosecha nacional de café sube un 14 %," La Nación, 2017. https://www.nacion.com/economia/agro/cosecha-nacional-de-cafe-sube-un-14/OXGRI2Q2CNAF7IB55P4GXBDSPM/story/.

[3] Icafe, "Acerca del icafe," Instituto del Café de Costa Rica, 2018. http://www.icafe.cr/icafe/acerca-del-icafe/.

[4] J. Cordero, "Diseño de un sistema de cuantificación automática de biomasa basado en procesamiento de imágenes y fotogrametría con vehículos aéreos no tripulados," Escuela de Ingeniería Electrónica. Instituto Tecnológico de Costa Rica, 12 2018.

[5] S. López, "Seguimiento de plantaciones de café a través de fotogrametría UAS," Escuela de Ingeniería Electrónica. Instituto Tecnológico de Costa Rica, 9 2019.

[6] S. Quesada, "Detección automática sobre ortomosaicos RGB de plantas de banano," Escuela de Ingeniería Electrónica. Instituto Tecnológico de Costa Rica, 11 2019.

[7] B. Weinberg and B. Bealer, El mundo de la cafeína. Tezontle, 2 ed., 2012. http://www.fondodeculturaeconomica.com/subdirectorios_site/libros_electronicos/desde_la_imprenta/017555R/files/publication.pdf.

[8] I. C. Council, "World coffee trade (1963 – 2013): A review of the markets, challenges and opportunities facing the sector," in International Coffee Council. London, UK, pp. 1–7, Internarional Coffee Organization, Feb. 2014. http://www.ico.org/news/icc-111-5-r1e-world-coffee-outlook.pdf.

[9] Icafe, "Historia del café de costa rica," Icafe, 2018. http://www.icafe.cr/nuestro-cafe/historia/.

[10] ND, "Informe sobre la actividad cafetalera de costa rica," in XLVI Congreso Nacional Cafetalero Ordinario, Heredia, Costa Rica, p. 24, 12 2017. http://www.icafe.cr/wp-content/uploads/informacion_mercado/informes_actividad/anteriores/2017.pdf.

[11] M. Nielsen, Neural Networks and Deep Learning. Determination Press, 2015. http://neuralnetworksanddeeplearning.com/about.html.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," Nature, vol. 323, pp. 533–, Oct. 1986.

[13] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.

[14] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 9, pp. 1627–1645, 2010.

[15] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," International Journal of Computer Vision, vol. 88, pp. 303–338, 06 2010.

[16] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587, 2014.

[17] H. G. LeCun Y, Bengio Y, "Deep learning," Nature, vol. 521(7553), pp. 436–444, 05 2015.

[18] R. Girshick, "Fast r-cnn," in 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448, 2015.

[19] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," CoRR, vol. abs/1506.02640, 2015. https://pjreddie.com/media/files/papers/yolo_1.pdf.

[20] B. A. .C and E. Chandra, "Deep learning architectures, algorithms for speech recognition: An overview," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 7, 1 2017. http://www.ijarcsse.com/docs/papers/Volume_7/1_January2017/V7I1-0107.pdf.

[21] M. Nielsen, Neural Networks and Deep Learning, ch. 6: Deep Learning. Determination Press, 2015. http://neuralnetworksanddeeplearning.com/chap6.html.

[22] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "Ghostnet: More features from cheap operations," CoRR, 2019.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[24] A. Kathuria, "What's new in yolo v3?," Towards Data Science, 4 2017. https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b.

[25] E. Madrigal, "Sistema de entrenamiento de redes neuronales artificiales con predicción eficiente en arquitecturas embebidas," Área Académica de Ingeniería en Computadores. Instituto Tecnológico de Costa Rica, 10 2018.

[26] F. Nielsen, "On a generalization of the jensen–shannon divergence and the jensen–shannon centroid," Entropy, vol. 22, p. 221, feb 2020.

[27] J. T. Wallelign S., Polceanu M. and C. B., "Coffee grading with convolutional neural networks using small datasets with high variance," Journal of WSCG, 2019.

[28] S. Jadhav, V. Udupi, and S. Patil, "Identification of plant diseases using convolutional neural networks," International Journal of Information Technology, 2020.

[29] L. Ximenes, C. Ferraz, F. Fambrini, R. Goulart, and J. Saito, "Coffee leaf disease recognition based on deep learning and texture attributes," Procedia Computer Science, 2019.

[30] J. Esgario, R. Krohling, and J. Ventura, "Deep learning for classification and severity estimation of coffee leaf biotic stress," Computers and Electronics in Agriculture, 2020.

[31] I. Y. Ekawaty and I. S. Areni, "Automatic cacao pod detection under outdoor condition using computer vision," 4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE), 2019.

[32] J. Z. et al., "Tree crown detection in high resolution optical images during the early growth stages of eucalyptus plantations in brazil," The First Asian Conference on Pattern Recognition, 2011.

[33] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," CoRR, 2020.

[34] J. Glenn, "Yolov5 release v6.1.," ND, 7 2022. https://github.com/ultralytics/yolov5/releases/tag/v6.1.

[35] L. Biewald, "Experiment tracking with weights and biases," 2020.

[36] L. Scrucca, M. Fop, T. Murphy, and A. Raftery, "mclust 5: Clustering, classification and density estimation using gaussian finite mixture models," The R Journal, vol. 8, pp. 205–233, 08 2016.

[37] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022.

[38] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," CoRR, 2014.