

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“DESARROLLO DE UNA HERRAMIENTA DE
APRENDIZAJE PROFUNDO PARA AGILIZAR EL
MONITOREO Y REVISIÓN DE TRAMPAS DE LA MOSCA
DE LA FRUTA CERATITIS CAPITATA EN EL CONTEXTO
CHILENO”

RODRIGO ANTONIO HERNÁNDEZ GONZALES

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Julio Sotelo
Profesor Correferente: Nelson Díaz

Julio - 2026



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Desarrollo de una herramienta de aprendizaje profundo para agilizar el monitoreo y revisión de trampas de la mosca de la fruta *Ceratitis capitata* en el contexto chileno

Nombre del candidato(a): Rodrigo Antonio Hernández Gonzales

Carrera / Grado: Ingeniería Civil Informática

Campus: Casa Central

Departamento: Departamento de Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Julio Sotelo, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (**marcar una opción**):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 28/07/2026

Firma:

Estudiante o Candidato(a):

Fecha: 28/07/2026

Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

DEDICATORIA

A mis padres, por siempre decirme lo que necesito escuchar aunque nunca les he
agradecido debidamente.

AGRADECIMIENTOS

A mi madre, que siempre ha confiado en mí y siempre me está apoyando, incluso si la situación es imposible.

RESUMEN

La mosca de la fruta (*Ceratitis capitata*) es una de las plagas agrícolas más destructivas del mundo. Su control es vital porque daña directamente la producción frutícola, pudriendo los cultivos y generando enormes pérdidas económicas. Además, su presencia provoca el cierre de mercados internacionales debido a cuarentenas, afectando las exportaciones. Identificar un brote tempranamente permite implementar medidas de erradicación focalizadas y menos tóxicas. En esta memoria se implementaron dos modelos de YOLOv11 para la detección de la mosca de la fruta. A ambos modelos se le agregó un aumento de datos y a uno de ellos, además, se aplica un ponderador de pesos. Se creó un dataset con 5 especies de moscas similares para que el modelo pueda diferenciar entre ellas y ayudar al usuario. El mejor modelo obtuvo un mAP@50-95 de 70,1 % mientras que el otro obtuvo un 68.6 %. Finalmente, se usó el modelo con mejor desempeño en una aplicación web para detectar ejemplares de mosca de la fruta y distinguir entre otras especies similares. Esta aplicación permite la carga de imágenes, la gestión de listas de trampas y la visualización del historial de análisis en tiempo real para optimizar la toma de decisiones en el control fitosanitario.

Palabras Clave— Aprendizaje profundo, Control de Plagas, Mosca de la fruta.

ABSTRACT

Abstract— The fruit fly (*Ceratitis capitata*) is one of the world's most destructive agricultural pests. Controlling it is vital because it directly damages fruit production, causing crop rot and massive economic losses. Furthermore, its presence leads to the closure of international markets due to quarantine measures, thereby impacting exports. Early detection of an outbreak enables the implementation of targeted, less toxic eradication measures. In this study, two YOLOv11 models were implemented for fruit fly detection. Data augmentation was applied to both models, while a weight-balancing mechanism was additionally applied to one of them. A dataset comprising five similar fly species was created to enable the model to distinguish between them and assist the user. The best-performing model achieved an mAP@50-95 of 70.1 %, while the other achieved 68.6 %. Finally, the top-performing model was integrated into a web application designed to detect fruit fly specimens and distinguish them from similar species. This application supports image uploading, trap list management, and real-time visualization of analysis history, thereby optimizing decision-making in phytosanitary control.

Keywords— Deep Learning, Pest Control, Fruit fly.

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	IV
ÍNDICE DE FIGURAS	VIII
ÍNDICE DE TABLAS	IX
CAPÍTULO 1: INTRODUCCIÓN	1
1.1 Costos de una zona libre de mosca de la fruta	2
CAPÍTULO 2: DEFINICIÓN DEL PROBLEMA	4
2.1 Objetivo General	7
2.2 Objetivos Específicos	7
2.3 Plan de Trabajo	7
CAPÍTULO 3: MARCO CONCEPTUAL	8
3.1 Etología para el control de plagas	8
3.1.1 Control biológico	8
3.1.2 Control de plagas con químicos	9
3.1.3 Control mediante uso de trampas	10
3.2 Deep Learning (Aprendizaje Profundo)	12
3.2.1 Métodos	15
3.2.2 Métricas de Evaluación	16
3.3 Estado del Arte	19
3.4 Arquitectura del Servidor	25
3.4.1 Herramientas	25
3.4.2 Backend	26
3.4.3 Frontend	27
CAPÍTULO 4: PROPUESTA DE SOLUCIÓN	29
4.1 Metodología	31
4.1.1 Creación del Dataset	32
4.1.2 Balanceo de datos	33
4.1.3 Entrenamiento de YOLOv11	36
4.2 Validación de los modelos	37
4.2.1 Resultados de los Modelos	39
4.3 Validación de la Aplicación	46
CAPÍTULO 5: VALIDACIÓN DE LA SOLUCIÓN	50

5.1	Funcionalidades Requeridas	51
5.1.1	Detección inteligente e Historial y Trazabilidad	51
5.1.2	Geolocalización y Colaboración Técnica	54
5.1.3	Ordenamiento por Clasificación	55
5.2	Pruebas de Humo	56
5.3	Pruebas Unitarias	57
CAPÍTULO 6: CONCLUSIONES		59
ANEXOS		61
6.1	Link video de propuesta	61
REFERENCIAS BIBLIOGRÁFICAS		62

ÍNDICE DE FIGURAS

1	Afiche del Proyecto Nacional para la erradicación de moscas de la fruta Fuente: www.sag.gob.cl	2
2	Árbol del Problema	4
3	Hormigas siendo tratadas con radiación para hacerlas estériles. Fuente: fue-dei.org	9
4	Hombre rociando insecticida para combatir el pulgón amarillo de la avena. Fuente: agronegocios.co	10
5	Trampas utilizadas por SAG. Fuente: sag.cl	11
6	Estructura de una CNN. Adaptado de [O'Shea and Nash 2015]	12
7	Ejemplo de detección de objetos en una intersección. Fuente visionplatform.ai	13
8	Estructura del algoritmo YOLOv11. Adaptado de [Khanam and Hussain 2024]	15
9	Modelo de detección de YOLOv4. Adaptado de [Zhong et al. 2018].	20
10	Estilo de trampas (a) y uso de la grilla (b). Adaptado de [Roosjen et al. 2020].	20
11	Esquema general del algoritmo. Adaptado de [Rustia et al. 2020].	21
12	Uso de las CL dentro del algoritmo. Adaptado de [Rustia et al. 2020].	21
13	Diagrama de Bloques del Servidor.	28
14	Afiche del Proyecto Nacional para la erradicación de moscas de la fruta. Fuente: www.sag.gob.cl	31
15	Ejemplo de un aumento de datos.	34
16	Curva de Training Loss y Validation Loss del Modelo 1.	38
17	Curva de Training Loss y Validation Loss del Modelo 2.	38
18	Matriz de Confusión del Modelo 1.	40
19	Matriz de Confusión del Modelo 2.	41
20	Muestra de predicción de dataset de validación con el Modelo 2.	42
21	Gráfico de Precisión del Modelo 1.	42
22	Gráfico de Precisión del Modelo 2.	43
23	Gráfico de recall del Modelo 1.	43
24	Gráfico de recall del Modelo 2.	44
25	Gráfico de mAP del Modelo 1.	44
26	Gráfico de mAP del Modelo 2.	45
27	Carga y análisis de imágenes.	47
28	Carga y análisis de mapas.	48
29	Historial de análisis de imágenes.	49
30	Página principal de la aplicación	50
31	Análisis de una imagen subida por el usuario.	52
32	Historial de imágenes subidas por el usuario junto con su análisis.	53
33	Historial de imágenes subidas por el usuario a un punto en específico.	54
34	Interfaz para la carga de listas de trampas con usuarios invitados.	55

35	Interfaz de los puntos de trampas con sus imágenes subidas, ordenados. . . .	56
----	--	----

ÍNDICE DE TABLAS

1	Condiciones para la exportación de hortalizas por país. Adaptado de [Centro de Información de Recursos Naturales (CIREN) Año de publicación (ej. 2018].	3
2	Revisión y detección de ejemplares en trampas. Fuente: sag.cl	5
3	Matriz de Confusión para la clasificación de dos clases.	16
4	Combinaciones de técnicas descritas por [Mahato et al. 2025].	22
5	Comparativa técnica entre arquitecturas de detección de Faster R-CNN y modelo YOLOv11.	24
6	Especificaciones y Capacidades esperadas del Sistema de Monitoreo.	30
7	Distribución de imágenes por especie de mosca.	32
8	División del Dataset.	33
9	Técnicas de aumento de datos aplicadas.	34
10	Set de entrenamiento aumentado.	35
11	Parametros del entrenamiento de YOLOv11.	37
12	Comparativa de mAP entre modelos.	45
13	Comparativa de mAP entre modelos y clases.	45
14	Resultado de las Pruebas de Humo.	57
15	Resultado de las Pruebas Unitarias.	58

CAPÍTULO 1

INTRODUCCIÓN

La mosca de la fruta es uno de los insectos de mayor importancia económica a nivel mundial, ya que la plaga del mismo ataca a más de 250 especies de árboles frutales y hortalizas, llegando a provocar pérdidas del 80 % en la producción de cultivos hortícolas (dependiendo de la localidad, variedad y temporada). La hembra de la mosca causa daño a las hortalizas depositando sus larvas dentro del fruto, las cuales se alimentan de su pulpa provocando su descomposición y posterior destrucción. Debido a su adaptación a diversas regiones, gran apetito y rápida reproducción, es uno de los insectos más estudiados en los últimos años con tal de combatir el gran daño económico que esta plaga conlleva [Kibira et al. 2015].

En Chile, es considerada la plaga de mayor relevancia económica para la exportación de productos agrícolas debido a las estrictas restricciones de los mercados internacionales. Su control es sumamente difícil principalmente por su capacidad de penetración y adaptación en las condiciones ambientales del país, lo que le ha permitido acceder a nuestro territorio en numerosas oportunidades, todas las cuales ha logrado ser suprimida [Lobos A. 2005]. Específicamente, en el país se cree que se encuentra presente solo 1 género de la mosca, *Ceratitis capitata* [Carillo et al. 2026]. Es por ello que el Servicio Agrícola y Ganadero (SAG) desarrolla desde 1980, un Proyecto Nacional para la erradicación de moscas de la fruta, el cual busca crear una producción de fruta fresca destinada a mercados externos libres de contaminación, aislamiento de áreas libres de moscas de la fruta, la existencia de un sistema de cuarentena vegetal en caso de avistamientos, contar con un sistema nacional de detección y presentar un plan de emergencia para erradicar ejemplares de *Ceratitis capitata* [Lobos A. 1996]. En la Figura 1 se puede evidenciar un afiche de la campaña.



Figura 1: Afiche del Proyecto Nacional para la erradicación de moscas de la fruta
Fuente: www.sag.gob.cl

Desde 1995, el país se declaró como “libre de mosca de la fruta (Familia Tephritidae)”, siendo el único país de Latinoamérica con esta categoría hasta la fecha [Quiroga and Ramírez 2016]. El hecho de que Chile sea el único país del continente libre de la mosca de la fruta es una gran ventaja económica frente a sus vecinos. Esta condición le permite exportar con preferencia a mercados de alta exigencia fitosanitaria, asegurando un acceso privilegiado para sus frutas y verduras, además de atraer mayor cantidad de clientes.

1.1. Costos de una zona libre de mosca de la fruta

Aunque Chile cuenta con barreras naturales que evitan el ingreso de especies desde el continente, pero dado el gran flujo turístico y comercial que existe, deja al país expuesto a una constante presión biológica de ingreso de la plaga mediante fruta contaminada no declarada, pese a las estrictas medidas de vigilancia que existen en los diferentes controles fronterizos a lo largo del territorio.

En caso de que se produzca un brote, existe una serie de regulaciones impuestas por los mercados internacionales para controlar el brote, que van desde un programa de medidas fitosanitarias realizado por SAG hasta el establecimiento de una Área Reglamentada (área en la cual las plantas, productos vegetales y otros artículos están sujetos a medidas fitosanitarias) [Cerezos 2024].

Cada Área Reglamentaria tiene una extensión de un radio diferente dependiendo del país al que se exporte la fruta contaminada, que van desde los 7.2 km hasta los 80 km desde el punto de la trampa donde se detectó el espécimen.

Dentro de esta Área Reglamentada el SAG realiza una serie de actividades preventivas, fitosanitarias y de muestreo de arboles y frutos:

- **Encuestas:** para conocer si en el lugar existen árboles o plantas hospederas que deban ser muestreadas.
- **Toma de muestras de frutas:** para detectar la presencia de larvas de mosca de la fruta en ellas.
- **Descarga de fruta de los árboles:** para evitar que la mosca de la fruta deposite sus huevos en frutos sanos.
- **Tratamiento de asperjado de árboles:** se realiza en casas, calles y plazas para eliminar las moscas que andan volando.
- **Trampeo:** instalación de trampas en árboles hospederos, las que se revisan periódicamente para detectar la presencia de moscas adultas.

Ademas, cada país tiene condiciones diferentes para la exportación de hortalizas de una Área Reglamentada como se evidencia en la Tabla 1:

Medida	Países
Prohibición de exportación	Australia, Corea, Cuba, India (según especie), Nueva Caledonia, Nueva Zelandia, República Dominicana, Taiwan
Tratamiento de frío Tratamiento de fumigación	Argentina, China, Corea, Estados Unidos, Filipinas, Japón, India , Indonesia, México, Sri-Lanka, Tailandia.
Sin restricciones	Unión Europea, Brasil, Medio Oriente

Tabla 1: Condiciones para la exportación de hortalizas por país. Adaptado de [Centro de Información de Recursos Naturales (CIREN) Año de publicación (ej. 2018)].

Con esto se puede apreciar el gran costo y personal requerido que conlleva un brote y la exhaustiva labor que se requiere para mantener el estado de zona libre de la mosca de la fruta, por lo que encontrar formas de agilizar la detección y respuesta ante un eventual brote se traduce en miles de dólares.

Por ello, en los últimos años se han empleado un sin numero de nuevos enfoques y estrategias con tal de hacer este proceso más expedito que van desde el uso de trampas electrónicas conectadas a internet las cuales detectan ciertos tipos de plagas rápidamente con tal de controlar posibles focos, hasta otras que usan técnicas químicas y estrategias basadas en el comportamiento de la especie a controlar.

CAPÍTULO 2

DEFINICIÓN DEL PROBLEMA



Figura 2: Árbol del Problema

Desde Arica y Parinacota hasta la región de Aysen, el SAG cuenta con un sistema de vigilancia y detección de moscas de la fruta que consta de unas 16.000 trampas que cubren 373.000 hectáreas de producción de fruta. Las medidas de importación de algunos países exigen al país exportador contar con un sistema de detección mediante trampas y revisiones frecuentes. Cada área que cuente con este sistema de trampeo y no ha detectado ningún ejemplar de mosca de la fruta en los últimos 4 meses se considera zona libre de mosca de la fruta [Cerezos 2024]. Para este sistema funcione se necesita una vigilancia constante durante todo el año que se traduce en la revisión sistemática de trampas cada 7 a 10 días (según la temporada) por personal del SAG para luego, en caso de detección de un posible ejemplar, enviar ciertas trampas a una segunda revisión realizada por un profesional con tal de confirmar si efectivamente es un espécimen de *Ceratitis capitata* [Rodríguez Carlos 2012].

Con esto, si tomamos en cuenta que el numero de trampas a revisar se eleva muchísimo cuando se establece una zona de cuarentena (Área Reglamentada) es imperioso buscar una forma de optimizar y agilizar el proceso de recolección, detección y verificación de ejempla-

res de *Ceratitis capitata*.

Por ejemplo, en la campaña de erradicación ocurrida en Ovalle, Monte Patria y Punitanqui, Región de Coquimbo, ocurrida el 30 de Octubre de 2023, podemos observar las siguientes detecciones (entre muchas) de mosca del mediterráneo. Si observamos la Tabla 2, la fecha de recolección de la trampa en el día 29 de Febrero nos damos cuenta que desde que se revisa la trampa hasta que se logra identificar un ejemplar de mosca del mediterráneo pasan 5 días, mientras que en los otros 2 casos la diferencia entre revisión e identificación puede tener una diferencia de hasta 24 días en el 4 de Marzo.

Fecha revisión	Detección nro.	...	Arbol con la trampa	Identificador - Fecha en laboratorio	...
29-02-2024	34	...	Olivo/Olea Europea	05-03-2024	...
04-03-2024	35	...	Higuera/Ficus Carica	02-04-2024	...
01-04-2024	36	...	Olivo/Olea Europea	07-04-2024	...

Tabla 2: Revisión y detección de ejemplares en trampas. Fuente: sag.cl

Aunque el promedio de días entre revisión y detección en gran parte de los casos es de 2 a 3 días, en casi todas las campañas podemos evidenciar casos como los presentados arriba, en los cuales demoran varias semanas en identificar un ejemplar en una trampa, lo que evidencia la sobrecarga de trabajo que llevan los identificadores expertos en zonas de cuarentena.

Es de suma importancia identificar estos ejemplares de forma rauda ya que al momento de detectar efectivamente una mosca, se realizan una serie de medidas en el punto geográfico de la trampa donde se encontró el ejemplar, que van desde la aplicación de insecticidas, trampeo masivo e inspección de frutas [OIEA 2005]. Todas estas medidas pueden no tener efecto si el tiempo entre la revisión y la identificación es demasiado largo, ya que otros ejemplares de la mosca se pueden mover de la zona por el largo periodo, lo que hace las medidas anteriormente mencionadas inútiles, llevando a un gasto ineficiente e innecesario de dinero.

La confusión entre la mosca de la fruta (*Ceratitis capitata*) y otras especies similares, como la mosca del vinagre (*Drosophila melanogaster*) o incluso la mosca doméstica, es un error común que suele derivar de su diminuto tamaño y hábitats compartidos [Cancino and Montoya 2004]. A diferencia de la mosca común, que busca materia en descomposición general, la mosca de la fruta se siente atraída específicamente por los azúcares de la fermentación; sin embargo, el verdadero dolor de cabeza para los inspectores es que estas moscas son tan pequeñas, que a menudo son confundidas con sus parientes inofensivos pero posee la capacidad de atacar fruta sana aún en la planta.

Por ello es importante hacer que las personas puedan distinguir los ejemplares de este insecto y con ello contar con dispositivos tecnológicos que hagan esta distinción mucho más confiable para evitar generar falsos positivos.

2.1. Objetivo General

Desarrollo de una herramienta de deep learning que permita la identificación de ejemplares de mosca de la fruta *Ceratitis capitata* mediante el análisis de imágenes de trampas pegables.

2.2. Objetivos Específicos

1. Creación de un dataset de imágenes de la mosca de la fruta *Ceratitis capitata* y ejemplares similares indicados por SAG.
2. Implementar algoritmos basados en visión por computadora para la detección de plagas en trampas y elegir el mejor evaluado.
3. Creación de una aplicación web con el modelo implementado y un sistema de clasificación de moscas.

2.3. Plan de Trabajo

Actividad	Duración (semanas)
Elaboración de Estado del Arte	5
Creación de un dataset de las diferentes moscas	3
Proponer un marco de evaluación e implementación de algoritmo elegido	6
Desarrollo de aplicación web con el algoritmo mejor evaluado	6
Redacción Informe final	3
Total	23

CAPÍTULO 3

MARCO CONCEPTUAL

3.1. Etología para el control de plagas

La Etología, según la Real Academia Española, se refiere a una rama de la biología que estudia el comportamiento de los animales en medios naturales¹, en este caso insectos. En si, una gran parte de esta rama se refiere al control de plagas de diversas especies que afectan a ganado y cultivos a lo largo del mundo, que es lo que nos interesa en nuestro caso. En esta sección nos centraremos en las metodologías más utilizadas para el control de plagas de insectos (especialmente de las moscas de la fruta) y como se relacionan con las aplicadas por el Servicio Agrícola y Ganadero.

3.1.1. Control biológico

El control biológico se basa en la producción, manipulación y despliegue en masa de depredadores naturales, parásitos o insectos estériles de la plaga que se desea combatir. Los insectos son manipulados a nivel biológico con tal de exterminar la plaga. La manipulación principalmente consiste en que estos insectos sean estériles, y/o que sus crías no vivan lo suficiente como para reproducirse [Jiménez 2009]. Un ejemplo de esto se evidencia en la Figura 3

¹<https://dle.rae.es/etología>

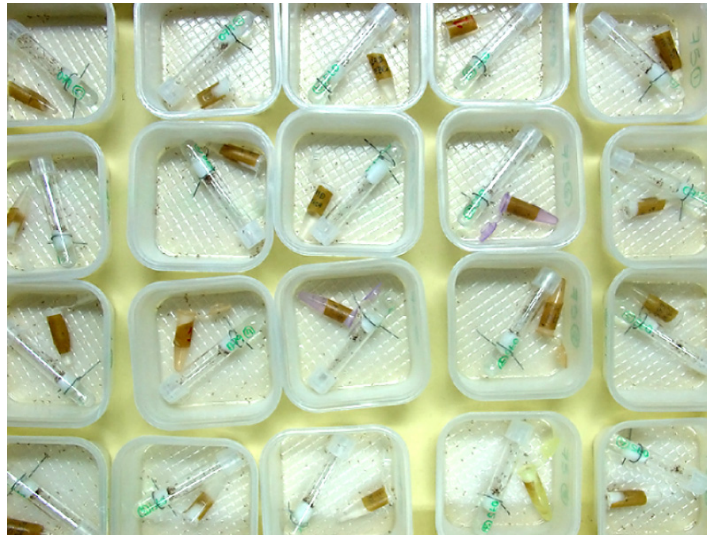


Figura 3: Hormigas siendo tratadas con radiación para hacerlas estériles. Fuente: fuedei.org

3.1.2. Control de plagas con químicos

Mediante este método, se utilizan sustancias químicas o biológicas que resultan letales para la plaga que se quiere combatir. Dentro de esta categoría existen un sinnúmero de sustancias disponibles en el mercado que resultan letales para varias plagas de insectos a la vez, pero también son tóxicas para el ganado, la flora, fauna y en algunos casos para los seres humanos por lo que es el método de control de plagas más cuestionado hoy en día y su aplicación se realiza con numerosas medidas de seguridad [Martínez 2010]. Aun así, es el método por preferencia utilizado hoy en día debido a su efectividad, aunque este conlleva un gran costo monetario para hacerlo eficaz en el control de una plaga a gran escala. En la Figura 4 se encuentra un ejemplo de esta práctica que es ampliamente utilizada en el mundo.



Figura 4: Hombre rociando insecticida para combatir el pulgón amarillo de la avena. Fuente: agronegocios.co

3.1.3. Control mediante uso de trampas

El método de mayor relevancia en este trabajo es el uso de trampas. El trampeo masivo es una buena forma de combatir una plaga cuando hay escases de comida, ya que se pueden ubicar en puntos estratégicos de importancia para lo que se quiere atrapar y es la principal forma que tiene SAG para controlar la plaga de mosca de la fruta.

En las trampas para insectos, la táctica mas común es utilizar trampas de papel pegajoso junto con colores atrayentes y/o feromonas de hembra centradas específicamente el insecto que se quiera atrapar, lo que ha demostrado ser una metodología con excelentes resultados [Andrade S. 2007]. Por ejemplo, un gran numero de especies de moscas se ven atraídas por el **color amarillo**, dentro de las cuales se encuentran la mosca domestica, la mosca blanca y la mosca de la fruta *Ceratitis capitata*.

Específicamente, en esta sección nos centraremos en describir los 2 tipos de trampas para mosca de la fruta que utiliza el Servicio Agrícola y Ganadero referenciadas en la Figura 5.



Trampa tipo Jackson



Trampa tipo McPhail

Figura 5: Trampas utilizadas por SAG. Fuente: sag.cl

- Trampa tipo Jackson: Trampa de tipo delta construida de cartón blanco cubierto con una delgada capa de plástico por los costados. El atrayente se sitúa en la parte superior al interior de la trampa en una rejilla de plástico. En la parte inferior se encuentra una lamina de cartón amarilla cubierta con un pegamento especializado para atrapar insectos. Esta lamina es extraible y es la que recolecta el personal del SAG en caso de notar la presencia de insectos con características de propias de la mosca de la fruta (moscas pequeñas, de distintos colores y características definidas).
- Trampa tipo McPhail: Trampa de tipo líquida, que consta de una botella con una abertura en su parte inferior en forma de dona por la cual entran las moscas. La trampa se llena de líquido atrayente junto con otros ingredientes los cuales hacen que las moscas caigan al líquido y no puedan volver a volar. Si el líquido contiene la presencia de insectos pequeños con forma de moscas de la fruta, el personal del SAG recolecta este líquido y lo guarda en un recipiente de cristal para su posterior análisis.

3.2. Deep Learning (Aprendizaje Profundo)

Es un subcampo del *Machine Learning* que se fundamenta en el uso de redes neuronales artificiales multicapa para modelar y resolver problemas complejos de alta dimensionalidad [Sarker 2021]. A diferencia de los enfoques estadísticos tradicionales, los algoritmos de Deep Learning poseen la capacidad de extraer y aprender de características de forma completamente automatizada directamente desde los datos crudos (como los píxeles de una imagen), eliminando la necesidad de una guía de aprendizajes manual. En el contexto de esta memoria, estas arquitecturas permiten procesar variaciones de color, condiciones de iluminación cambiantes y la presencia de oclusiones en las capturas de las trampas agrícolas.

Redes Neuronales Convolucionales (CNN):

Una red neuronal convolucional (CNN) es un tipo de red neuronal artificial diseñada específicamente para analizar datos visuales. Una CNN se entrena para ver el mundo identificando patrones en datos visuales. Pero su principal ventaja no es solo ver imágenes, sino reconocer patrones en cualquier dato con una estructura similar a una cuadrícula, desde espectrogramas de audio hasta escáneres médicos. Este proceso genera mapas de características que capturan patrones en las imágenes (bordes, texturas y contrastes) en las primeras capas, hasta estructuras complejas de mayor relevancia en las capas más profundas [O'Shea and Nash 2015]. Su estructura se muestra en la Figura 6

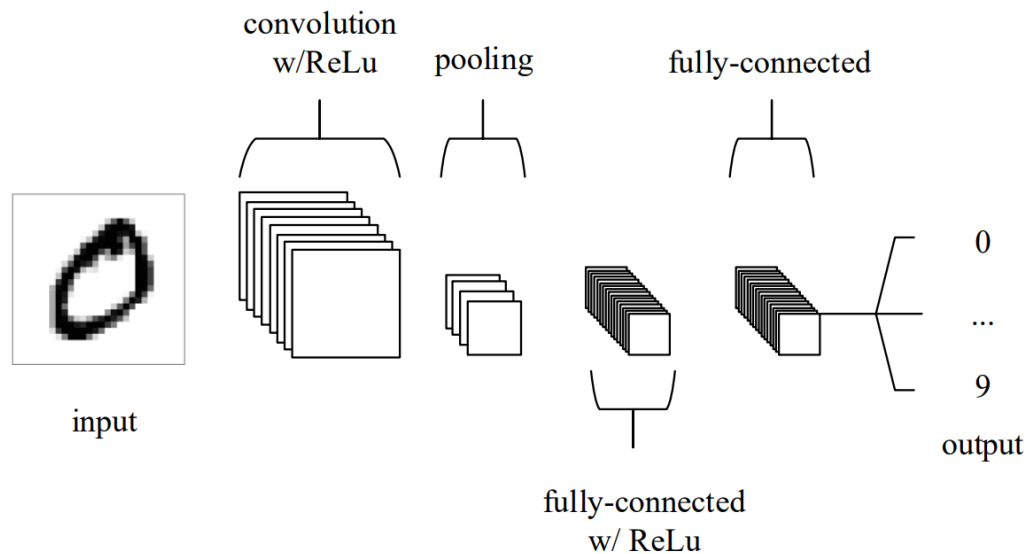


Figura 6: Estructura de una CNN. Adaptado de [O'Shea and Nash 2015]

Detección de Objetos:

A diferencia de la clasificación de imágenes tradicional, que asigna una única etiqueta a una imagen completa, la detección de objetos aborda la localización y clasificación simultánea de múltiples elementos dentro de una misma escena. Se realiza mediante dos pasos: la regresión de coordenadas para delimitar los cuadros delimitadores (bounding boxes) que encierran a los objetos de interés, y la clasificación probabilística de la clase contenida en dicho cuadro. En el monitoreo de plagas, esta técnica es fundamental, ya que una sola trampa pegable contiene decenas de insectos capturados simultáneamente que deben ser analizados de forma individual. Un ejemplo de detección de objetos se muestra en la Figura 7.

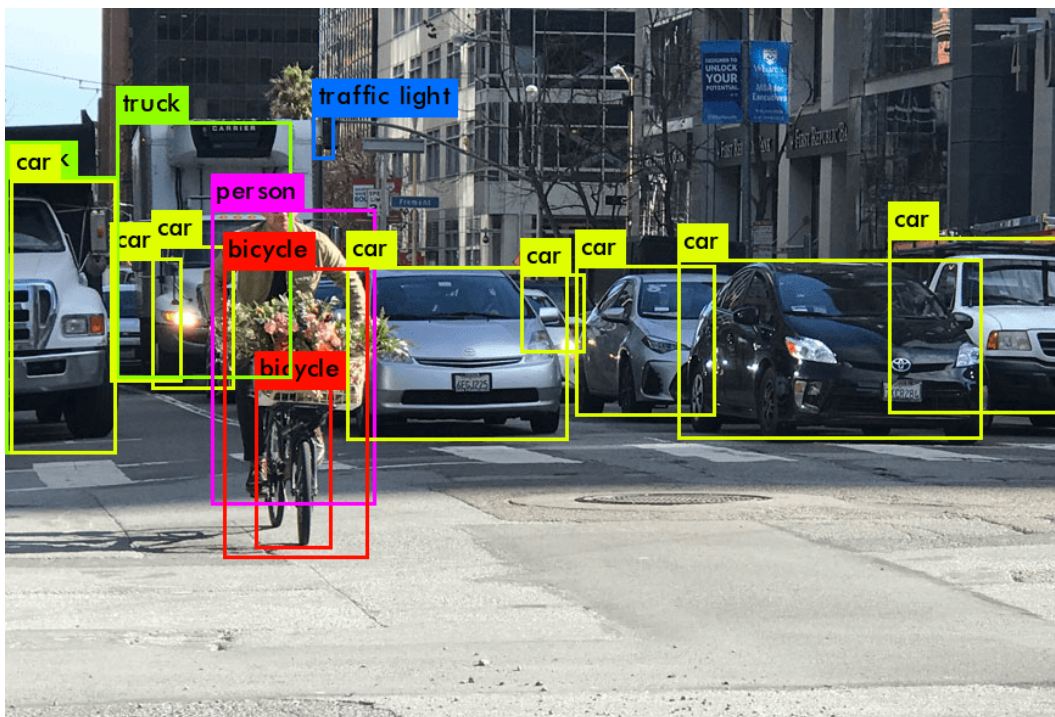


Figura 7: Ejemplo de detección de objetos en una intersección. Fuente visionplatform.ai

Función de Pérdida

Es el componente matemático que cuantifica la discrepancia entre las predicciones del modelo y las clase real durante la fase de entrenamiento. En arquitecturas de detección de objetos como YOLOv11, es una función multiobjetivo que penaliza de forma combinada tres aspectos:

- Pérdida por localización (box Loss)
Evalúa el error en las coordenadas y dimensiones de los cuadros predichos.

- Pérdida por clasificación (cls loss)
Evalúa la capacidad del modelo para asignar correctamente la clase a cada objeto detectado
- Distribution Focal Loss (dfl loss):
Optimiza de forma continua la regresión de los bordes de las cajas delimitadoras tratándolos como una distribución de probabilidad generalizada

Durante el entrenamiento, el optimizador busca minimizar globalmente este valor ajustando los pesos de la red.

YOLOv11:

YOLO es una serie de algoritmos de visión por computadora desarrollados por Ultralytics. Presentado en la conferencia YOLO Vision 2024, YOLOv11 representa un avance significativo en la tecnología de detección de objetos en tiempo real. Esta versión introduce mejoras sustanciales tanto en la arquitectura como en las metodologías de entrenamiento, superando los límites de la precisión, la velocidad y la eficiencia a comparación a sus anteriores versiones. Esta nueva versión incorpora técnicas avanzadas de extracción de características, lo que permite una captura de detalles más especializada, manteniendo una extracción de parámetros precisa, especializada en el reconocimiento de objetivos. Un gran número de artículos basados en el reconocimiento de objetos utilizan versiones de este algoritmo por su eficacia para la clasificación de objetos.

Su arquitectura se compone de tres bloques fundamentales. El primero, conocido como backbone, está basado en CSPDarknet53 y preentrenado en el dataset COCO; cumple la función de extraer características visuales de manera eficiente y a múltiples escalas. El segundo bloque, denominado neck, empleó una combinación de módulos Spatial Pyramid Pooling Fast (SPPF) y Path Aggregation Network (PANet), que permitieron fusionar eficazmente las características extraídas y conservar detalles relevantes a diferentes resoluciones. El último bloque es el head, que se caracteriza por ser desacoplado y libre de anclas (anchor-free). Este diseño permitió predecir directamente el centro de los objetos y separa las tareas de clasificación y regresión de las cajas delimitadoras, lo que favorece una convergencia más rápida y mayor precisión en la detección [Khanam and Hussain 2024]. En la Figura 8 se puede apreciar un resumen de las etapas de YOLOv11.

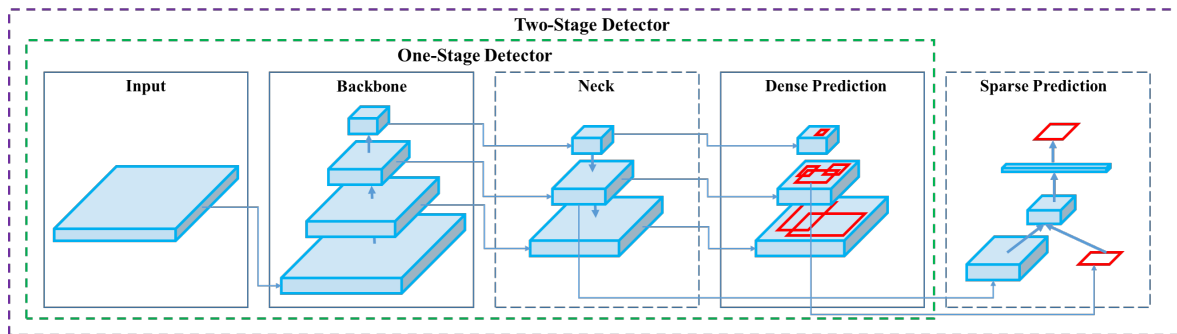


Figura 8: Estructura del algoritmo YOLOv11. Adaptado de [Khanam and Hussain 2024]

3.2.1. Métodos

Data augmentation:

Son métodos que se usan para aprovechar al máximo los pocos ejemplos de datos de prueba y aumentar la precisión de modelos algorítmicos. Estas técnicas pueden ser escalar, rotar, reflejar horizontal o verticalmente, desplazar horizontal o verticalmente, hacer zoom, entre otros. Se espera que, con estas técnicas, el aumento de datos ayude a prevenir el sobreajuste (problema común cuando se tiene un conjunto de datos pequeño) y así el modelo mejore su capacidad para generalizar.

Transferencia de aprendizaje (Transfer Learning):

Es una técnica que se emplea en modelos CNN pre-entrenados. Estos fueron entrenados con data masiva y lo que se quiere es ajustar las características aprendidas en los últimos niveles y añadir data objetivo para obtener el resultado deseado.

Weighted dataloader:

Técnica que modifica la forma en que se muestrean los lotes de entrenamiento en el aprendizaje automático, asignando probabilidades mayores a las clases con menos datos. Se utiliza principalmente para corregir el desequilibrio de clases cuando un conjunto de datos contiene muchos más ejemplos de una categoría que de otra, asegurando que los ejemplos poco frecuentes o minoritarios se suministren al modelo con mayor asiduidad.

Optimizador Adam:

Abreviatura de Adaptive Moment Estimation, es un algoritmo de optimización sofisticado ampliamente utilizado para entrenar modelos de aprendizaje profundo. Al calcular tasas de aprendizaje adaptativas individuales para diferentes parámetros a partir de estimaciones de los primeros y segundos momentos de los gradientes, Adam permite que las redes neuronales converjan significativamente más rápido que los métodos tradicionales.

3.2.2. Métricas de Evaluación

En el ámbito de la clasificación de imágenes, se recurre a diversas métricas para evaluar el desempeño de un modelo, siendo las más comunes la exactitud (*Accuracy*), la precisión y el *recall* [Rainio et al. 2024].

El cálculo de estos indicadores se fundamenta en la matriz de confusión, una herramienta que clasifica los resultados en cuatro grupos según la concordancia entre la predicción y la realidad: los Verdaderos Positivos (TP), correspondientes a los datos de la clase objetivo correctamente identificados; los Falsos Positivos (FP), que representan elementos de otras categorías asignados erróneamente a la clase de interés; los Falsos Negativos (FN), que constituyen omisiones de la clase objetivo; y los Verdaderos Negativos (TN), que reflejan los descartes correctos de otras clases. En la Tabla 3 se muestra un ejemplo de la Matriz de Confusión.

		Clase verdadera	
		Positivo	Negativo
Clase predicha	Positivo	True Positive (TP)	False Positive (FP)
	Negativo	False Negative (FN)	True Negative (TN)

Tabla 3: Matriz de Confusión para la clasificación de dos clases.

Exactitud:

Es la proporción de predicciones correctas (tanto positivas como negativas) sobre el total de casos. Evalúa que tan cerca del rango estuvo el modelo original. En la ecuación (1) se muestra como obtenerla:

$$\text{Exactitud} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Esta fórmula entrega un número entre 0 y 1. Un valor cercano a 1 indica un muy buen modelo, mientras que un valor cercano a 0 indica un mal modelo. Esta métrica es bastante intuitiva y fácil de comprender. Sin embargo, hay que tener en cuenta que evalúa mal el rendimiento

de un modelo basado en datos desequilibrados, o datos en los que los errores de predicción no tienen el mismo impacto.

La exactitud no permite hacer matices entre las diferentes predicciones y no toma en cuenta los datos desequilibrados. Es por esta razón que existen métricas para paliar este problema:

Precisión:

Se utiliza para determinar la calidad de las predicciones positivas. Responde a: "De todas las veces que el modelo dijo 'Positivo', ¿cuántas veces acertó realmente?". En (2) se muestra como obtenerla

$$\text{Precisión} = \frac{TP}{TP + FP} \quad (2)$$

Recall:

Se enfoca en la capacidad del modelo para encontrar todos los casos positivos reales (evitar falsos negativos). Se calcula como se muestra en (3)

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

Intersección sobre Unión (IoU):

Se utiliza para evaluar el rendimiento de un modelo comparando el cuadro delimitador real con el cuadro delimitador predicho. En (4) se muestra su cálculo:

$$\text{IoU} = \frac{\text{Área de la Intersección}}{\text{Área de la Unión}} \quad (4)$$

Average Precision (AP):

Es una popular herramienta de comparación entre modelos que toma en consideración la precisión y el recall de los algoritmos de detección. Resume la curva de precisión-recall en un valor entre 0 y 1 mediante una integral de la precisión en función del recall. A diferencia de la precisión simple, que se calcula en un punto fijo, el AP busca promediar la precisión sobre todos los niveles de recall posibles como se muestra en (5).

$$\text{AP} = \int_0^1 p(r) dr \quad (5)$$

Mean Average Precision (mAP):

Es simplemente el promedio del AP calculado para cada una de las clases (K) del conjunto de datos. Si tu modelo detecta 10 tipos de objetos diferentes, calculas el AP para cada uno y luego se promedia. Una forma de obtenerla es como se muestra en (6):

$$mAP = \frac{1}{K} \sum_{i=1}^K AP_i \quad (6)$$

3.3. Estado del Arte

En la última década, con el desarrollo de múltiples algoritmos y técnicas de visión por computadora, no es de extrañar de que muchos autores se hayan centrado en utilizarlos para reconocer insectos y combatir plagas debido a las ventajas y facilidades que ofrecen para observar puntos estáticos, como lo son las trampas. Así, simplemente realizando una búsqueda en bases de datos científicas (por ejemplo Google Scholar) con palabras clave como reconocimiento de plagas con visión por computadora o detección de plagas usando redes neuronales, un gran número de artículos científicos que utilizan aquellas técnicas aparecen como resultado. La mayoría de estos artículos se han publicado en los últimos 15 años, lo que evidencia a la detección y monitoreo automático de plagas como una disciplina reciente que esta en auge.

En esta sección nos concentraremos específicamente en hacer una revisión de aquellos artículos relacionados a las moscas de la fruta y como los autores han llevado a cabo sus acercamientos al monitoreo y detección de esta plaga usando visión por computadora junto con diferentes técnicas relacionadas con ella.

El artículo desarrollado por Zhong et al. (2018) llama especialmente la atención. Los autores desarrollan un sistema de conteo y reconocimiento de insectos atrapados en trampas plegables utilizando métodos basados en visión por computadora. El enfoque utilizado para monitorear las trampas fue acoplar una Raspberry Pi junto con una cámara que esta grabando en tiempo real la trampa. El algoritmo utilizado para la detección de objetos (insectos) fue YOLOv4 mientras que la clasificación y el conteo se realizó mediante un algoritmo de SVM (*support vector machine*). La cámara toma una foto de la trampa cada cierto tiempo, la divide como una grilla en imágenes de 30x30 píxeles y las pasa a YOLOv4 para la detección de insectos. Con ello, mediante métodos matemáticos se extraen características de las detecciones, se unen y se pasan al algoritmo basado en SVM para la clasificación y el conteo. Cabe recalcar que el autor señala a YOLOv4 como el mejor algoritmo para la detección de objetos. El proceso se muestra en la Figura 9.

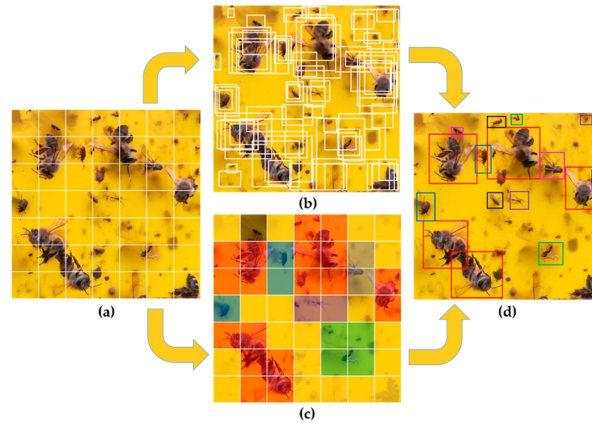


Figura 9: Modelo de detección de YOLOv4. Adaptado de [Zhong et al. 2018].

Otro enfoque interesante fue propuesto por Roosjen et al. (2020) donde los autores desarrollan un sistema de monitoreo de trampas utilizando aprendizaje profundo para la detección de la mosca del vinagre de alas manchadas. Lo interesante de esta propuesta es que se utiliza un dron montado con una cámara que revisa sistemáticamente trampas. El modelo de reconocimiento propuesto está basado en un algoritmo de aprendizaje profundo llamado ResNet-18, que se especializa en el reconocimiento de solo una clase en particular. Debido a que las trampas para este tipo de mosca de la fruta son de color rojo, los autores utilizan esto como bordes para la creación de una grilla en la imagen obtenida por el dron y con ello aplican capas convolucionales con tal de mejorar el reconocimiento. El set de entrenamiento fue hecho con 249 imágenes y se aplican métodos de aumento como rotaciones o blur. El proceso se evidencia en la Figura 10.



Figura 10: Estilo de trampas (a) y uso de la grilla (b). Adaptado de [Roosjen et al. 2020].

Un enfoque interesante que usa IoT es propuesto en Rustia et al. (2020), el cual consiste en un algoritmo para la detección y el reconocimiento de insectos mediante aprendizaje profundo. El algoritmo utiliza YOLOv3 y una CNN para la detección de insectos dentro de las imágenes y descartar los que no lo eran. El sistema de monitoreo emplea un sistema in-

alámbrico con múltiples nodos de cámaras conectadas por Wi-Fi. El objetivo del algoritmo es clasificar los objetos dentro de las imágenes como insectos o no insectos, con especial atención a la identificación de cuatro especies de plagas que afectan a las frutas de invernadero. Estas especies incluyen tisanópteros, mosquitos, moscas blancas y moscas. El algoritmo de detección y reconocimiento procesa cada imagen utilizando un detector de objetos basado en YOLOv3 para obtener las coordenadas delimitadoras dentro de cada objeto presente en la trampa. Luego cada coordenada de los objetos se envía a un clasificador de imágenes que utiliza 3 CNN con tal de extraer las características de cada objeto y clasificarlos en las 5 clases antes mencionadas. El proceso se muestra en en las Figuras 11 y 12.

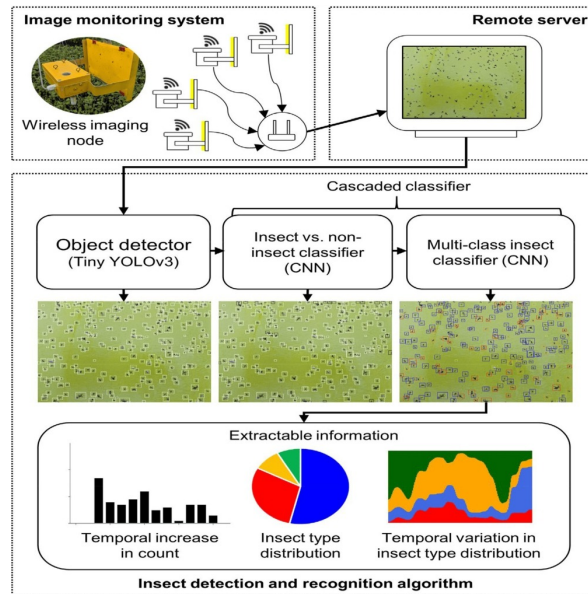


Figura 11: Esquema general del algoritmo. Adaptado de [Rustia et al. 2020].

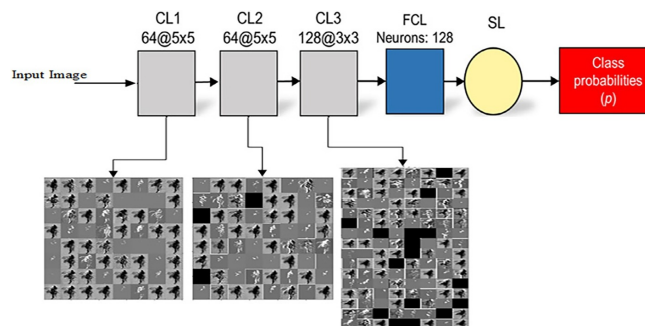


Figura 12: Uso de las CL dentro del algoritmo. Adaptado de [Rustia et al. 2020].

Otros trabajos recientes no solamente se basan en el reconocimiento de imágenes si no que proponen usar comportamientos típicos de las plagas a la luz, lo que se denomina Fotofobia. Esta habilidad que poseen muchas células para realizar movimientos orientados en respuesta a la intensidad de la luz lo que permite añadir luz a los dispositivos de control de plagas para incluso mejorar el rendimiento. Mahato et al. (2025) integra la Inteligencia Artificial con diferentes métodos existentes para revolucionar el control de plagas. Profundiza en los fundamentos biológicos de la respuesta de los insectos a los estímulos luminosos y examina las aplicaciones, tanto establecidas como emergentes, de los métodos de control de plagas basados en la luz. Así se pueden entrenar modelos que sean especialmente entrenados basandose, por ejemplo cuando una luz se enciende y la respuesta del objetivo. Además entregan varias estrategias y beneficios sobre el enfoque descritos en la Tabla 4.

Estrategia	Descripción	Beneficios
Integración con el Control del Hábitat	Colocación estratégica de trampas de luz en áreas modificadas para mejorar su efectividad, como cerca de barreras de vegetación o sitios de cría.	Combina el control mecánico con la gestión del hábitat para un enfoque holístico y sostenible.
Complemento con Agentes Biológicos	Atracción de plagas a áreas donde se introducen depredadores naturales o parasitoides, aprovechando la fototaxia para el control biológico.	Mejora la eficacia de los agentes de control biológico al concentrar las poblaciones de plagas en áreas específicas.
Longitudes de Onda de Luz Personalizadas	Diseño de trampas de luz con longitudes de onda optimizadas para especies de plagas específicas basadas en sus preferencias fototácticas.	Reduce los efectos en especies no deseadas, mejora las tasas de captura y minimiza el impacto ambiental de las estrategias basadas en luz.
Monitoreo Basado en Fototaxia	Seguimiento del comportamiento y movimiento de las plagas mediante su respuesta a estímulos luminosos.	Permite el seguimiento de la población de plagas en tiempo real, ayuda en la detección temprana y facilita la confianza.

Tabla 4: Combinaciones de técnicas descritas por [Mahato et al. 2025].

A partir de los documentos descritos anteriormente, los modelos basados en YOLO se han consolidado como el algoritmo estándar con mayor confianza en mucha de la literatura aca-

démica para el reconocimiento y clasificación de insectos en entornos no controlados. La prevalencia de este modelo sobre otras arquitecturas de detección de objetos se fundamenta en los siguientes beneficios:

- Compromiso entre precisión y bajo costo

A diferencia de los modelos basados en regiones (Region-based Convolutional Neural Networks o R-CNN), que operan mediante una segmentación de dos etapas, YOLOv11 formula la detección de objetos como un único problema de regresión. Mediante una sola pasada hacia adelante (forward pass) a través de la red, el sistema predice simultáneamente las coordenadas de los cuadros delimitadores (bounding boxes) y las probabilidades de una clase. En el contexto de la detección de insectos, esta capacidad de procesamiento de bajo costo computacional es crítica para la implementación de sistemas de alerta temprana de bajos recursos, donde la demora y/o falla en la identificación de una especie invasora puede comprometer la viabilidad económica de una zona agrícola completa

- Viabilidad en Hardware de bajos recursos

Una de las mayores limitantes de la investigación académica aplicada es la transición del laboratorio al campo. Los trabajos de tesis frecuentemente buscan la implementación en dispositivos de computación de bajos recursos, tales como Arduino o Raspberry Pi. YOLOv11 ofrece variantes de arquitectura Nano o Small que mantienen una alta eficiencia de inferencia con una carga computacional reducida, permitiendo la autonomía energética de las trampas inteligentes de bajos recursos y no muy potentes sin depender de conectividad constante a servidores en la nube.

Las investigaciones en torno al uso de deep learning para la segmentación de imágenes de insectos demuestran la viabilidad de emplear tanto algoritmos basados en segmentación por color como redes neuronales convolucionales (CNN). Bajo este enfoque, es posible tomar un modelo de detección preentrenado e implementar transferencia de aprendizaje (transfer learning), adaptándolo con precisión a las necesidades específicas de identificación y clasificación de plagas, específicamente de mosca de la fruta.

Con esto, en la Tabla 5 se muestra una pequeña comparación entre los 2 algoritmos estudiados a lo largo de la memoria.

Criterio	Faster R-CNN	Arquitectura YOLOv11
Velocidad	Baja a moderada	Alta (Tiempo Real)
Complejidad Computacional	Elevada	Optimizada para dispositivos móviles
Precisión en Objetos Pequeños	Alta (Históricamente superior)	Alta (Equiparada en versiones recientes)
Aplicabilidad en Campo	Limitada por hardware	Óptima para monitoreo autónomo

Tabla 5: Comparativa técnica entre arquitecturas de detección de Faster R-CNN y modelo YOLOv11.

3.4. Arquitectura del Servidor

El desarrollo del sistema de detección se articuló mediante la integración de un modelo de deep learning y una aplicación web. El modelo de visión computacional que será utilizado en esta aplicación fue YOLOv11, seleccionado por su precisión y velocidad en tiempo real para identificar múltiples clases en imágenes. Esta arquitectura se integró con una aplicación web dinámica desarrollada con el uso de Flask para el backend y en Bootstrap para la interfaz de usuario (frontend).

3.4.1. Herramientas

Python:

Lenguaje de programación orientado a objetos de alto nivel con semántica dinámica. Es el lenguaje de programación mas avanzado y con mayor soporte para el desarrollo de aplicaciones con base científica, las que incluyen Ciencia de Datos, Procesamiento de Imágenes, Aprendizaje Profundo, Visualización de Datos, *Machine Learning*, etc. [Dhruv et al. 2020]

Keras:

API que orienta al usuario a seguir buenas practicas en la programación para reducir la carga cognitiva al leer el código. A su vez proporciona consejos y comentarios para posibles errores del usuario ².

Tensorflow:

Biblioteca de código abierto desarrollada por Google para el aprendizaje automático. Ofrece una amplia gama de herramientas para cualquier nivel de abstracción que permiten desarrollar aplicaciones de aprendizaje automático que pueden ejecutarse en cualquier entorno ³.

Scikit-learn:

Herramienta simple y eficiente de código abierto para el análisis de datos. Especialmente útil para analizar los datos que se extraen a través de como se entrenó el modelo de visión por computadora ⁴.

LabelImg:

Herramienta de anotación gráfica en imágenes especializada en el etiquetado de regiones de interés sobre imágenes para su posterior uso en el entrenamiento de modelos de visión

²keras.io

³tensorflow.org

⁴scikit-learn.org

por computadora ⁵.

Flask:

Micro-Framework basado en Python que tiene como objetivo crear aplicaciones web simples, de bajos requerimientos pero extensibles y compatibles con la mayoría de las ventajas que Python ofrece ⁶.

Bootstrap:

Popular framework de diseño web de código abierto para desarrollo frontend. Proporciona una colección de herramientas basadas en HTML, CSS y JavaScript para crear sitios y aplicaciones web adaptables (responsive), ahorrando tiempo al ofrecer componentes preconstruidos como botones, menús y formularios listos para usar. ⁷.

Google Colab:

Servicio en la nube basado en los Jupyter Notebooks que permite escribir y ejecutar código Python desde el navegador. No requiere instalación ni configuración previa, y destaca por ofrecer acceso gratuito a recursos de hardware avanzado, como GPUs y TPUs. Está diseñado principalmente para proyectos de inteligencia artificial, aprendizaje automático (machine learning) y análisis de datos. Su interfaz de cuaderno digital permite combinar bloques de código ejecutable con texto explicativo, fórmulas matemáticas y gráficos ⁸

MySQL:

Sistema de gestión de bases de datos relacionales (RDBMS) de código abierto. Funciona utilizando el lenguaje SQL (Structured Query Language) para almacenar, organizar, modificar y consultar información de manera estructurada en tablas vinculadas entre sí ⁹.

Jinja2:

Motor de plantillas para Python. Permite generar documentos dinámicos (como HTML) separando la lógica de programación (backend) de la presentación visual. Es ampliamente utilizado en frameworks web como Flask, Django y FastAPI para construir interfaces web modulares y reutilizables

3.4.2. Backend

El backend de la aplicación web fue desarrollado utilizando el micro-framework Flask en lenguaje Python, seleccionado por su ligereza y compatibilidad con librerías de deep learning. El

⁵pypi.org/project/labelImg

⁶flask.palletsprojects.com

⁷<https://getbootstrap.com/>

⁸colab.research.google.com

⁹<https://www.mysql.com/>

objetivo de este entorno es actuar como el motor lógico que orquesta el procesamiento de los datos y la seguridad, desde la recepción de la imagen hasta la entrega de los resultados analíticos.

Se implementaron servicios REST encargados de gestionar usuarios, registros de trampas, imágenes y resultados de detección, mediante formato JSON. Estos servicios reciben, almacenan y procesan las peticiones del usuario, para guardarlos en la base de datos MySQL y dejarlos disponibles para su posterior consulta. Se implementaron clases para lista de trampas, imágenes y usuarios con tal de usar los servicios de integración API que ofrece Flask y facilitar el manejo de datos.

3.4.3. Frontend

El frontend de la aplicación fue diseñado bajo un enfoque simplificado, utilizando el motor de plantillas Jinja2 integrado en Flask, en conjunto con HTML nativo y Bootstrap. Para las peticiones API REST utilizamos el motor por defecto de HTML junto con Javascript. Esto nos ahorra escribir consultas manuales SQL y usar las clases definidas en nuestro entorno de Flask. El objetivo principal de esta capa es ofrecer una interfaz web para autenticación, visualización de imágenes procesadas y gestión de resultados.

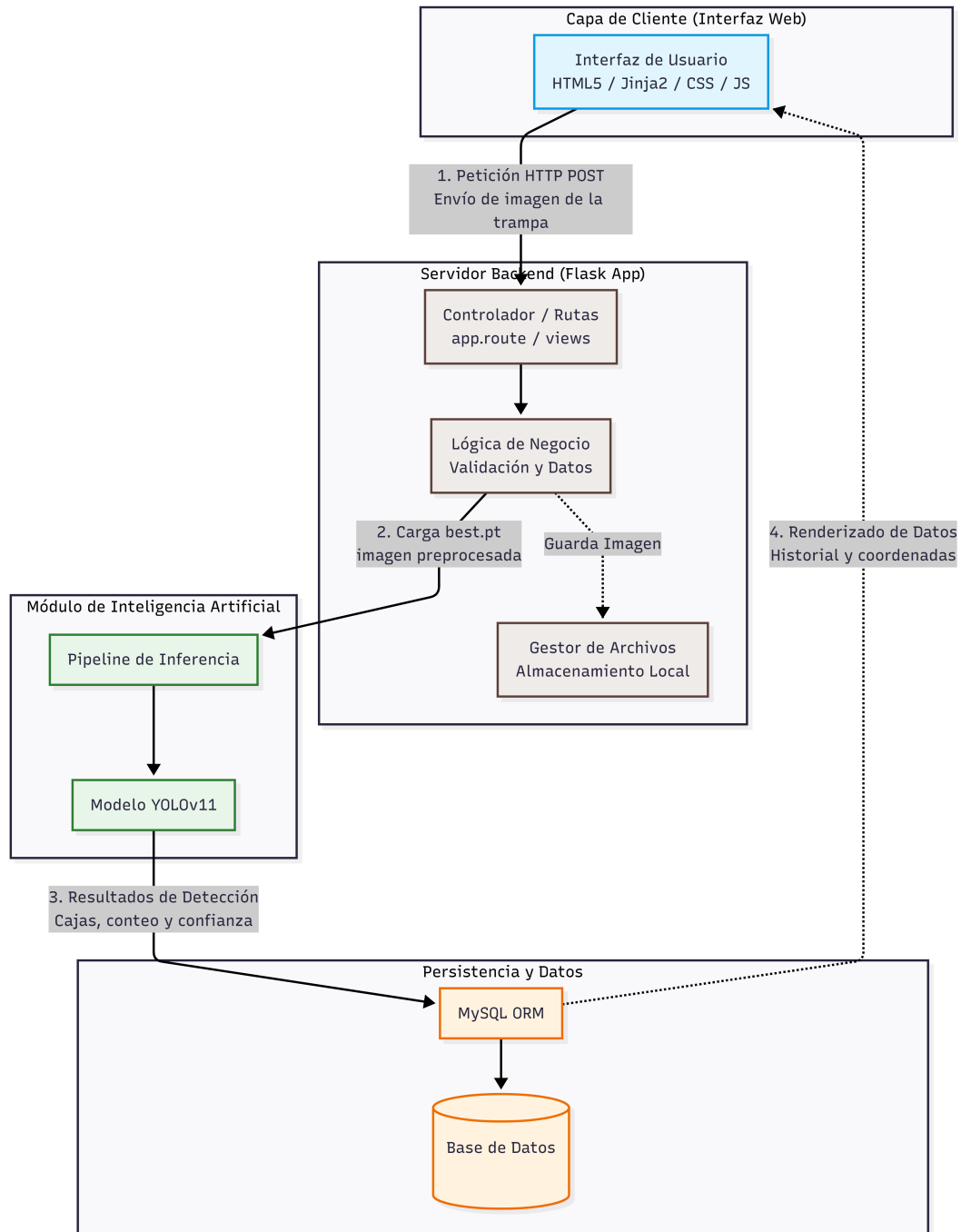


Figura 13: Diagrama de Bloques del Servidor.

CAPÍTULO 4

PROPUESTA DE SOLUCIÓN

Desafío

El monitoreo manual de trampas es costoso, lento y propenso a errores humanos. La demora entre la captura y la identificación permite que las plagas se propaguen antes de que se tomen medidas de control. Además las limitaciones de los dispositivos móviles y la similitud de ciertas especies de moscas hace que sea muy difícil distinguirlas entre sí, y más aún del ejemplar de *Ceratitis capitata* que nos interesa.

Con la gran cantidad de insectos que pueden ser atrapados y/o confundidos en las trampas desplegadas a través de territorios agrícolas se hace inherente contar con alguna forma de ayudar a inspectores y/o gente rural a distinguir ejemplares de insectos que puedan ser similares a la mosca de la fruta para contar con un grado de confianza de que se esta seguro que es el ejemplar que buscamos. Además, esta solución tecnológica debe ser portátil y no muy exigente de hardware, ya que se utilizara principalmente en dispositivos móviles.

Solución

Implementación de una aplicación para dispositivos móviles vinculada a un algoritmo de detección especializado en detección de objetos el cual busca agilizar la detección y revisión de trampas que contengan potencialmente este ejemplar de la mosca, además de ayudar a cualquier tipo de persona obtener una herramienta que funcione en dispositivos móviles para la detección y diferenciación del ejemplar.

Además, esta aplicación contará con sistema de almacenamiento y seguimiento de puntos donde está ubicada una trampa de mosca de la fruta. Así, se puede tener un registro del área de donde se tomo la foto y además se podrá abrir en el mapa el punto específico para una mejor experiencia de usuario. Esto tiene grandes ventajas, por ejemplo si un inspector o personal agrícola se encuentra revisando las trampas de la zona, se podrá tener un registro de todas las fotos de tal punto junto con el análisis de confianza que realizó el algoritmo sobre si era un espécimen de mosca de la fruta o una falsa alarma.

Funcionalidades Esperadas

Característica	Implementación	Ventaja
Detección Inteligente	Modelo de Deep Learning especializado en visión artificial para la identificación de moscas.	Agiliza la revisión y democratiza el acceso a herramientas de identificación técnicas.
Geolocalización	Sistema de trazado GPS con visualización de puntos de trapeo en mapas interactivos.	Precisión en el registro del área de muestreo y mejora de la experiencia de navegación en campo.
Historial y Trazabilidad	Almacenamiento centralizado de fotografías y resultados de análisis por punto específico.	Facilita la auditoría agrícola y permite contrastar detecciones reales frente a falsas alarmas.
Colaboración Técnica	Accesos multiusuario sincronizados para expertos (remotos) e inspectores (en terreno).	Permite la validación conjunta y el intercambio de criterios en tiempo real sobre una misma trampa.
Ordenamiento por Clasificación	Sistema de clasificación de trampas basado en el porcentaje de confianza más alto de los análisis de Ceratitis capitata.	Optimiza el flujo de trabajo al priorizar la revisión de puntos con mayor probabilidad de un brote.

Tabla 6: Especificaciones y Capacidades esperadas del Sistema de Monitoreo.

Basándonos en el contexto chileno y siguiendo las indicaciones de Servicio Agrícola y Ganadero, diferenciaremos las otras 4 especies de moscas que son muy similares y comunes a *Ceratitis capitata* evidenciables en la Figura 14:



Figura 14: Afiche del Proyecto Nacional para la erradicación de moscas de la fruta.

Fuente: www.sag.gob.cl

Para ello se propone la recolección de numerosas imágenes de las especies especificadas en la Figura 14 para crear un dataset de estos 5 especímenes con el objetivo de construir un modelo para la identificación y clasificación de moscas en imágenes de trampas para insectos.

4.1. Metodología

La presente sección describe la configuración experimental empleada recolectar, entrenar y evaluar el modelo YOLOv11 en la detección de los especímenes de moscas mostradas en la Figura 14. Se detalla la adquisición y creación del dataset, la anotación de las regiones de interés, la división en clases, las políticas de aumento y manejo de datos, los hiperparámetros de entrenamiento, así como el protocolo de evaluación y análisis de resultados.

4.1.1. Creación del Dataset

Debido a que actualmente no existe un dataset público ni sistematizado que contemple imágenes de estas 5 moscas, se creó un dataset propio con imágenes de cada espécimen, para que así el modelo pueda aprender y diferenciar la taxonomía de cada espécimen.

En internet existen múltiples imágenes de *Ceratitis capitata*, ya que es uno de los ejemplares de mosca mas estudiados a nivel mundial. Sin embargo, es dificultoso encontrar múltiples imágenes de las otras 4 especies de moscas- más aun en trampas- ya que 3 de ellas no están consideradas una plaga tan destructora como es la propia mosca del mediterráneo.

Especie	Imagenes
Ceratitis capitata	2535
Mosca del Vinagre	253
Mosca de Mesa	245
Mosca Abeja	223
Mosca Azul	166

Tabla 7: Distribución de imágenes por especie de mosca.

En la Tabla 7 se muestra la distribución final de las imágenes recolectadas. Así, un aspecto crítico de este dataset es el manejo del desbalance de clases, donde la especie principal bajo campaña de alerta y erradicación suele estar sobrerrepresentada en comparación con las especies menos representadas. Para mitigar esta estadística sin comprometer la veracidad taxonómica de las muestras, se deben implementar estrategias combinadas de balanceo de datos.

Para poder delimitar las regiones de interés y agregar labels a las imágenes recolectadas de internet se utilizo la web Roboflow. La Tabla 8 muestra cómo se dividió el dataset en una distribución de 70 %, 20 % y 10 % para las fases de entrenamiento, validación y prueba respectivamente por tipo de mosca.

Especie	Entrenamiento 70 %	Validación 20 %	Prueba 10 %
Ceratitis capitata	1775	507	253
Mosca del Vinagre	177	51	25
Mosca de Mesa	172	49	25
Mosca Abeja	156	45	23
Mosca Azul	117	33	16
Total	2397	685	342

Tabla 8: División del Dataset.

4.1.2. Balanceo de datos

Las muestras de *Ceratitis capitata* están muy por encima de las otras moscas. Para abordar este desbalance de clases, se pueden utilizar técnicas como aumento de datos (Data Augmentation) y ponderación de clases (Data weighting) los cuales juntos pueden ser implementados para que nuestro algoritmo sea mas robusto a pesar del desbalance de datos [Bakirarar and Elhan 2023].

Aumento de datos

Aumentar las imágenes de entrenamiento mediante rotaciones, modificaciones de color y otras variaciones de los patrones de sus píxeles, ayuda a los modelos a aprender a generalizar más allá de los ejemplos específicos del conjunto de datos [Yousefimehr et al. 2026]. Sin embargo, modificar los colores en las imágenes de entrenamiento también puede obstaculizar la capacidad del modelo para aprender ciertos patrones. En nuestro caso ocurre con la mosca azul y la mosca de mesa, donde su principal diferencia es el color en el cuerpo de ambas.

Es por esto que nuestro aumento de datos se basa principalmente en la modificación de las imágenes mediante rotaciones, reflexiones, ruido y cambios de iluminación representadas en la Tabla 9.

Técnicas	Límite inferior	Límite superior
Reflexión (espejo)	-	-
Rotación (grados)	-45°	45°
Gaussian Blur	3	7
Incremento de contraste	-30	30
Incremento de brillo	-20	20

Tabla 9: Técnicas de aumento de datos aplicadas.

A cada imagen que se le aplica el aumento de datos 6 veces y cada uno tiene una probabilidad específica para que se le aplique una o más transformaciones (50 % para Reflexión, Rotación, Contraste o Brillo, 30 % para Gaussian Blur). En caso de que no se aplique ninguna transformación en esa iteración, la imagen simplemente permanece igual y la iteración se salta.

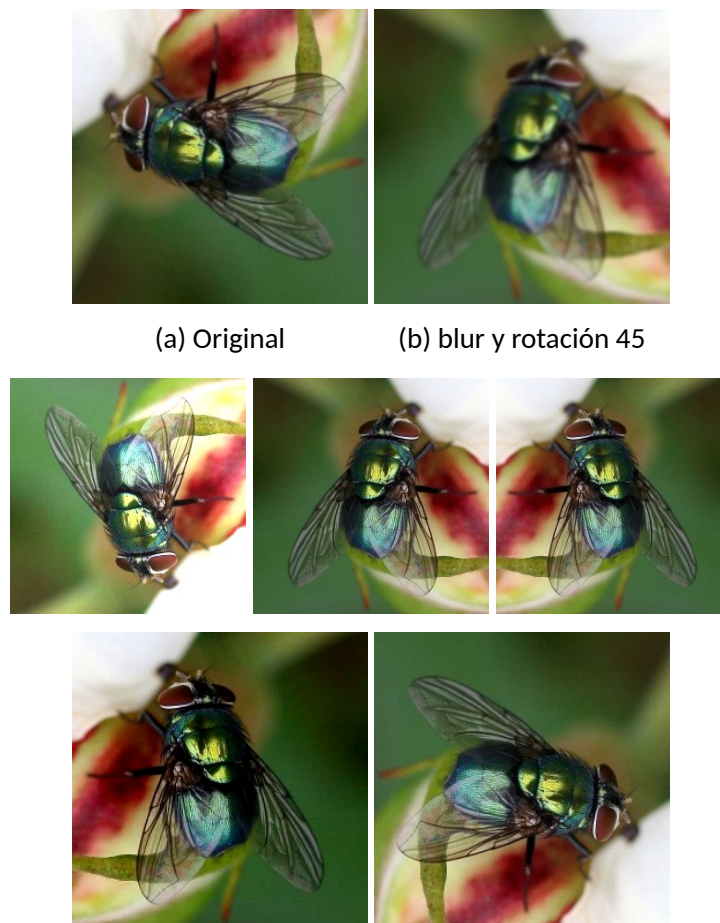


Figura 15: Ejemplo de un aumento de datos.

Con este esquema de aumento de datos representado en la Figura 15 equilibramos los pesos de cada clase y evitamos el *overfitting*.

Especie	Entrenamiento	Total aumentado
Ceratitis capitata	1775	1775
Mosca del Vinagre	177	1207
Mosca de Mesa	172	1127
Mosca Abeja	156	1061
Mosca Azul	117	819
Total	2397	5,989

Tabla 10: Set de entrenamiento aumentado.

Ponderación de clases (Data weighting)

En un escenario sin corregir, los modelos de detección de objetos tienden a optimizar su función de pérdida minimizando el error global a expensas de las clases minoritarias. Esto significa que el algoritmo podría especializarse excesivamente en reconocer *Ceratitis capitata* y clasificar erróneamente otras especies como falsos positivos de la clase mayoritaria [Yousefimehr et al. 2026]. Para neutralizar esta asimetría estadística sin recurrir exclusivamente a la generación artificial de imágenes, la combinación de técnicas de ponderación de datos (Data weighting) y el diseño de un cargador de datos (Dataloader) surge como una solución eficiente.

La técnica de data weighting actúa directamente sobre el cálculo de la función de pérdida del modelo. En lugar de tratar a todas las muestras por igual, se asigna un peso inversamente proporcional a la frecuencia de cada clase en el dataset original [Bakırrarar and Elhan 2023]. Así, si la clase de *Ceratitis capitata* tiene casi el doble de imágenes que las clases minoritarias, el peso asignado a sus errores de clasificación será correspondientemente menor, mientras que los errores cometidos al clasificar las especies menos representadas recibirán una penalización mucho mayor. De este modo, cuando el modelo comete un fallo al identificar una mosca de las otras 4 especies, la función de pérdida penaliza drásticamente al optimizador, forzando a la red a modificar sus pesos sinápticos para prestar especial atención a los sutiles detalles morfológicos que diferencian a esas especies menos comunes de la clase dominante.

Para llevar esta estrategia al entrenamiento de nuestro modelo, se construyó un custom dataloader el cual hace un muestreo ponderado de las clases según su tamaño, siguiendo el modo descrito en [Mohammed 2024]. Gracias a este dataloader, se logra que en lugar de alimentar al modelo con un batch de imágenes extraídas de forma secuencial o puramente

aleatoria (lo que resultaría en lotes dominados estadísticamente por *Ceratitis*), el dataloader hace que el modelo utilice las probabilidades calculadas a partir de los pesos inversos para reestructurar la composición de cada batch. En cada época de entrenamiento, el dataloader selecciona activamente muestras de las clases minoritarias con una frecuencia artificialmente elevada, logrando que cada lote de entrenamiento que ingresa a la GPU esté perfectamente balanceado, conteniendo una proporción equitativa de las 5 especies de moscas. Esta combinación no solo estabiliza los gradientes y acelera la convergencia del modelo, sino que previene el sobreajuste (overfitting) en la clase mayoritaria [Ying 2019].

4.1.3. Entrenamiento de YOLOv11

En esta sección se muestran los procedimientos, parámetros y resultados del entrenamiento de la arquitectura YOLOv11. El objetivo fue comparar su precisión, eficiencia y comportamiento frente a escenarios complejos de detección, aplicados a los ejemplares de las moscas antes descritos.

Con el objetivo de garantizar la validez de los métodos de balanceo de datos explicados anteriormente, se desarrolló una fase comparativa entre la experimentación inicial basada exclusivamente en el uso de técnicas clásicas de data augmentation y la presente versión del modelo que incluye la incorporación de un dataloader que mejore aún el problema de balanceo de datos presente que se tiene en nuestro dataset.

- **Modelo 1:**

El primer modelo fue entrenado utilizando únicamente transformaciones geométricas y fotométricas sobre imágenes reales, tales como rotación, ajuste de brillo y contraste, blur y reflexión. Este enfoque buscó ampliar la variabilidad visual del conjunto de datos sin alterar la morfología natural de las moscas, constituyendo así una línea base de referencia para la evaluación posterior.

- **Modelo 2:**

En el segundo modelo, se mantuvo la técnica de data augmentation pero además se implementó data weighting mediante un dataloader, con el objetivo de equilibrar la representación de las clases minoritarias y reforzar la confiabilidad del modelo frente a estos ejemplares menos frecuentes.

De este modo, se mantuvo un proceso de entrenamiento equivalente en cuanto a parámetros, número de épocas y criterios de validación, permitiendo una comparación directa entre ambos enfoques.

Para ambos modelos el entrenamiento se llevó a cabo en condiciones controladas y equivalentes. Se utilizó Google Colab junto con el modelo de YOLOv11 ofrecido por Ultralytics, una tarjeta gráfica Nvidia T400 ofrecida de manera gratuita por Colab. En ambos casos se utilizó el mismo dataset con el mismo split; 70 % de las imágenes para entrenamiento, un 20 % para validación y un 10 % de prueba, con el conjunto de imágenes de entrenamiento aumentadas por aumento de datos.

Para los parámetros de entrenamiento de ambos modelos, se utilizaron imágenes de 640x640 píxeles, el optimizador AdamW, una tasa de aprendizaje (learning rate) de 0.001, tamaño de lote (batch size) de 32 imágenes y un pre-entrenamiento (transfer learning) de COCO (por defecto de YOLOv11) todo esto con 50 épocas. Se resume en la Tabla 11

Parámetro	Modelo YOLOv11 (small)
Resolución de entrada	640 × 640 px
Optimizador	AdamW
Tasa de aprendizaje	0.001
Tamaño de lote (batch)	32
Épocas	50
Transfer learning	Pre-entrenado en COCO

Tabla 11: Parametros del entrenamiento de YOLOv11.

4.2. Validación de los modelos

Después de ambos entrenamientos, se obtuvo el modelo best.pt resultante de ambos entrenamientos, correspondiente al modelo de mejor rendimiento usando el set de validación. Este archivo se obtiene como un checkpoint donde el rendimiento del modelo fue el mejor obtenido cuando YOLOv11 usa el set de validación en ese momento para encontrar los mejores resultados sin pasarse al sobre ajuste.

En primer lugar el modelo entrenado solamente con el aumento de datos y con los pesos por clase pre-entrenados por transfer learning de COCO sin usar un dataloader, desde ahora se referirá como **Modelo 1**, mientras que al modelo entrenado usando un dataloader para ajustar los pesos dependiendo de la cantidad de imágenes por clase, se nombrará como **Modelo 2**.

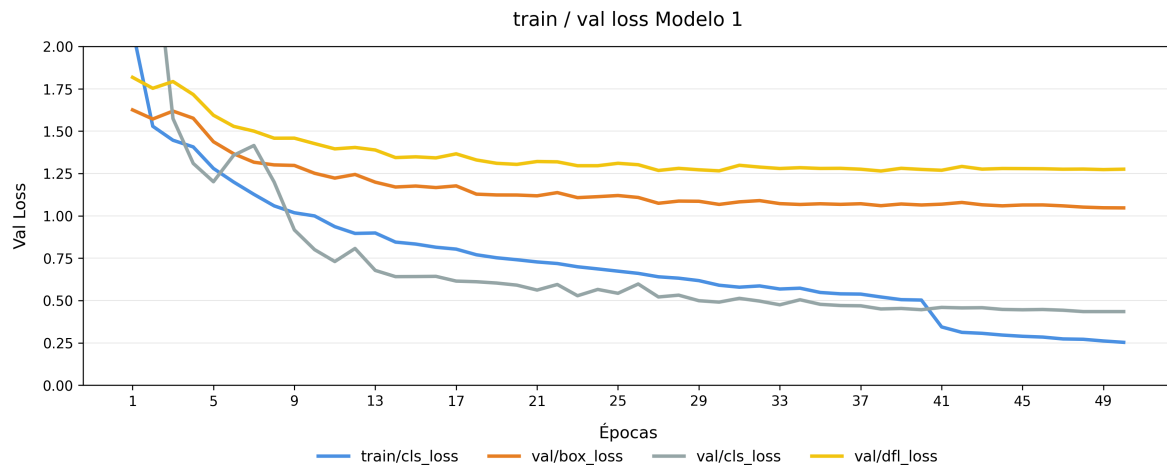


Figura 16: Curva de Training Loss y Validation Loss del Modelo 1.

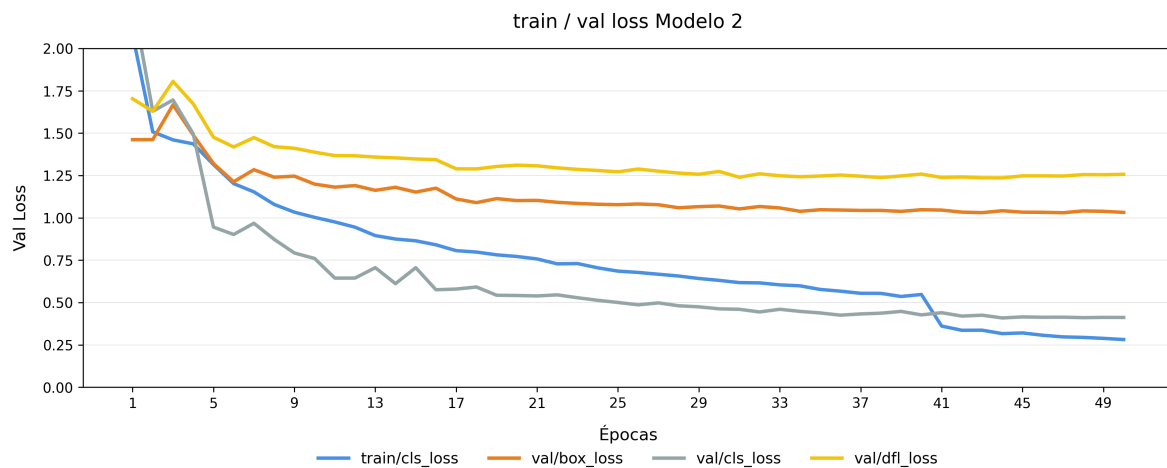


Figura 17: Curva de Training Loss y Validation Loss del Modelo 2.

En 16 y 17, se visualiza la curva de evolución de la pérdida total de entrenamiento a lo largo las 50 épocas. Se observa un pico inicial atribuible al calentamiento sobre el pre-entrenamiento de COCO y al descongelamiento progresivo del backbone; a partir de la época 8 aproximadamente la pérdida de todos los atributos decrece de manera sostenida en ambos casos con oscilaciones menores hasta estabilizarse, lo que evidencia convergencia del optimizador y ausencia de divergencias durante el ajuste en cada época.

Por su parte, el val/cls_loss, que evalúa la capacidad del modelo para asignar correctamente la clase a cada objeto detectado, constituye la mayor contribución a la pérdida total de los

indicadores de validación, lo cual es importante en escenarios con múltiples clases y variabilidad intraclase, como ocurre en imágenes de trampas pegables. En tanto, El val/box_loss y el val/df_loss relacionado con la calidad de la regresión de las cajas delimitadoras, se mantienen estables en ambos casos, evidenciando la estabilidad del modelo para crear las cajas alrededor de las moscas.

4.2.1. Resultados de los Modelos

El rendimiento de ambos modelos se evalúa mediante un conjunto de métricas clave: precisión, recall, F1-Score, IoU, mAP@0.5 y mAP@0.5:0.95 (explicadas en Métricas de Evaluación). En primer lugar, la precisión determina el porcentaje de aciertos reales dentro de las predicciones realizadas, mientras que el recall refleja la capacidad del modelo para localizar todos los elementos verdaderos presentes en la muestra. Para balancear ambas propiedades, se recurre al *F1-Score*. Por otro lado, la calidad geométrica de las delimitaciones se mide con el IoU, estableciendo el éxito de una predicción cuando dicha superposición alcanza o supera el 50 %. Finalmente, el mAP@0.5:0.95 destaca como el indicador de mayor rigurosidad para contrastar la eficacia de un modelo, puesto que promedia el desempeño bajo un rango variable de exigencia en el IoU.

En cuanto al desempeño de ambos modelos, lo podemos visualizar en la matriz de contingencia la representada en 18 y 19, la cual sirve para evaluar el rendimiento de un modelo de clasificación al comparar sus predicciones con los valores reales. Esta permite visualizar de forma rápida si el algoritmo está confundiendo dos clases o categorías distintas.

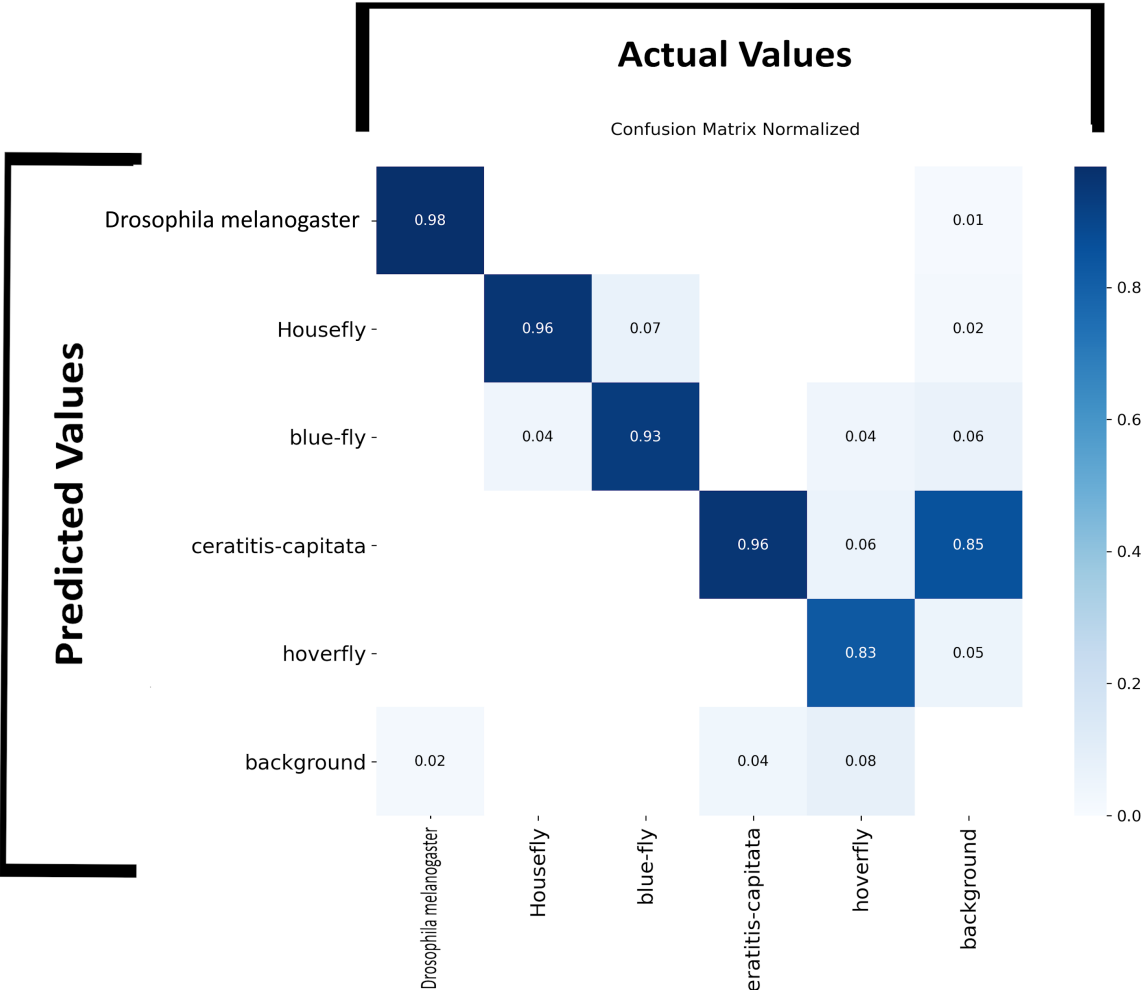


Figura 18: Matriz de Confusión del Modelo 1.

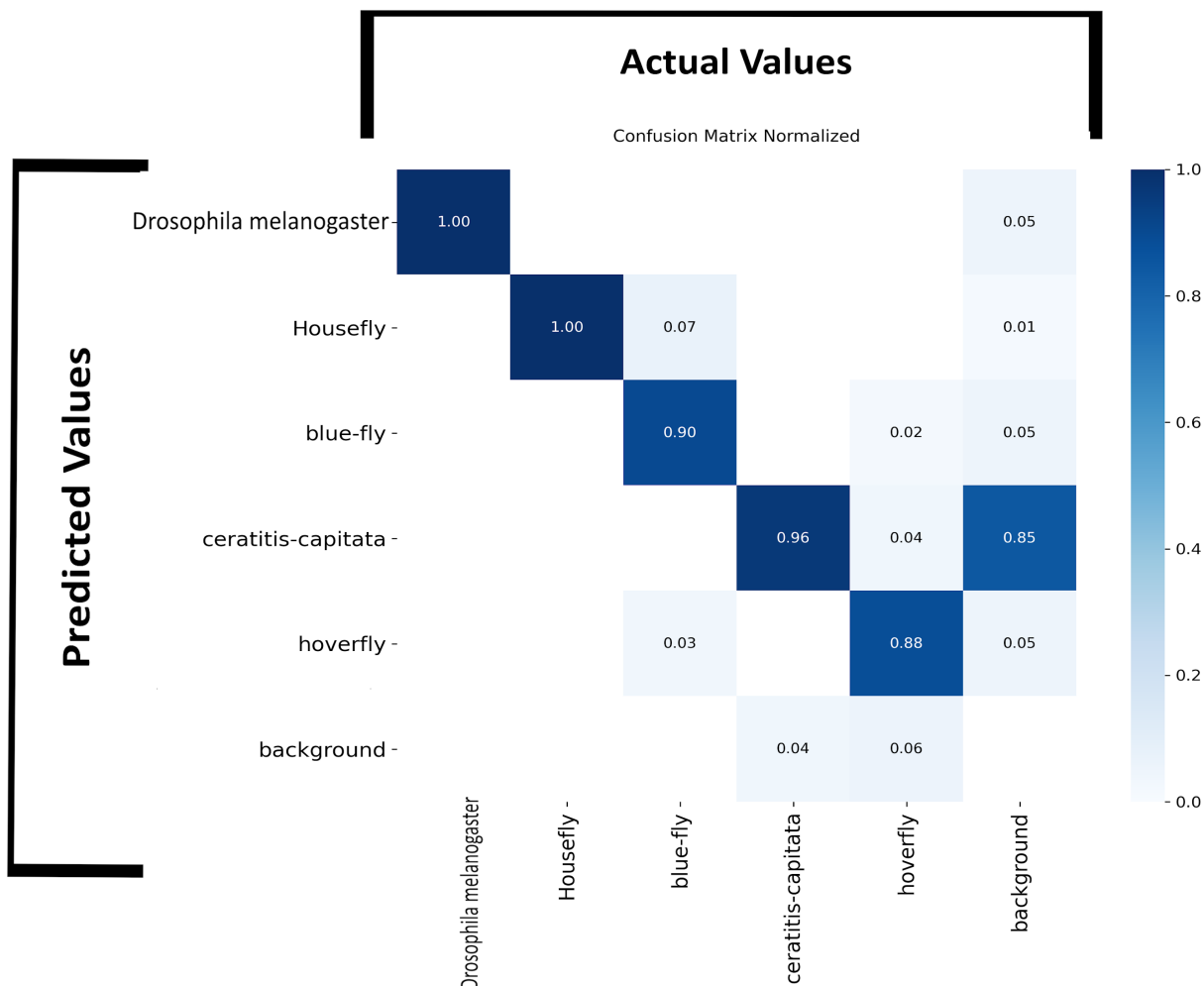


Figura 19: Matriz de Confusión del Modelo 2.

Al evaluar ambas matrices de contingencia, se evidencia un comportamiento dividido en la clase ceratitis-capitata. Los ejes de una matriz de confusión representan la relación entre la realidad y la predicción de un modelo de clasificación. Si bien el algoritmo demuestra una consistencia del 96 % para clasificar correctamente los rasgos morfológicos de la especie cuando logra localizarla, se identifica una tasa de omisión crítica del 85 % asociada al fondo (background). Este elevado índice de falsos negativos revela que el modelo adolece de una marcada subdetección espacial, catalogando a la mayoría de los ejemplares reales como espacios vacíos. Tras un análisis del dataset de validación, este fenómeno se atribuye directamente a que las instancias de ceratitis presentan dimensiones extremadamente reducidas en los cuadrantes de captura o muestran una baja resolución, lo que dificulta su visualización y diferenciación frente al entorno físico de la trampa. Aun así, se toma como un resultado aceptable, ya que una de los requerimientos de la aplicación es que las imágenes

sean de cerca. Un ejemplo del set de validación se muestra en la Figura 20.

Si se analizan ambas matrices, se nota una pequeña mejora en las predicciones del Modelo 2 en las clases de mosca del vinagre, mosca domestica y la mosca Abeja.



Figura 20: Muestra de predicción de dataset de validación con el Modelo 2.

En tanto se evidencia el gráfico de precisión de ambos modelos. La precisión cuantifica la proporción de predicciones correctas que realiza el modelo sobre el total de objetos que detecta.

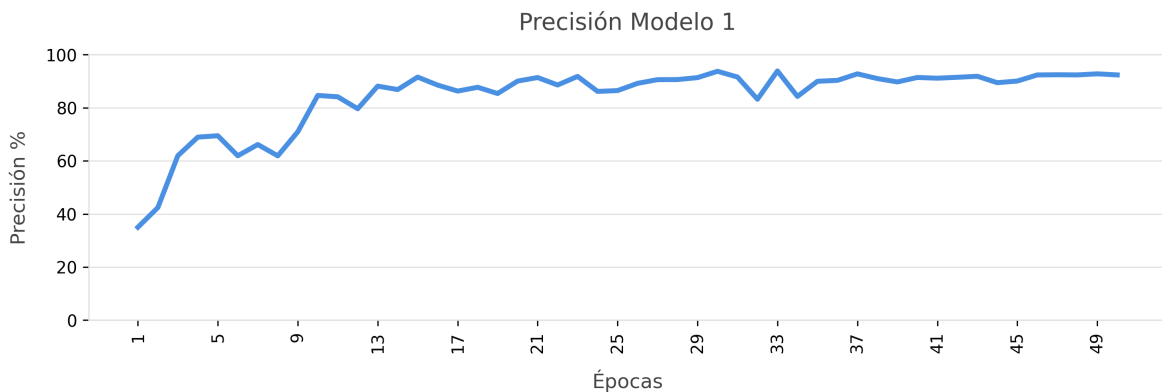


Figura 21: Gráfico de Precisión del Modelo 1.

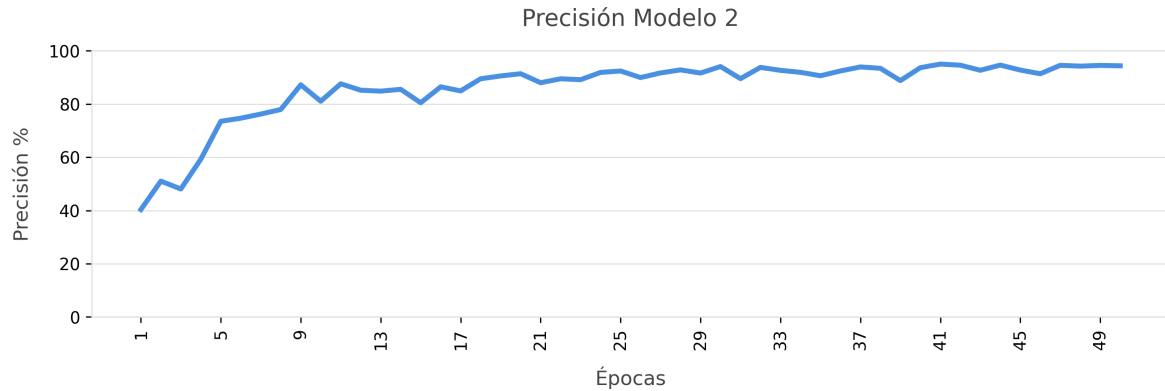


Figura 22: Gráfico de Precisión del Modelo 2.

En ambos gráficos 21 y 22, se observa una mejora progresiva de la precisión del modelo a lo largo de cada época, lo que indica que el modelo va mejorando progresivamente sin divergir. Además, se observa una pequeña mejora de precisión del Modelo 2 a comparación del Modelo 1 al final de las 50 épocas, además de mantenerse mucho mas estable.

Asimismo, resultados similares se observan el recall representado en 23 y 24, que mide la capacidad del modelo para encontrar todos los objetos que existen en una imagen:

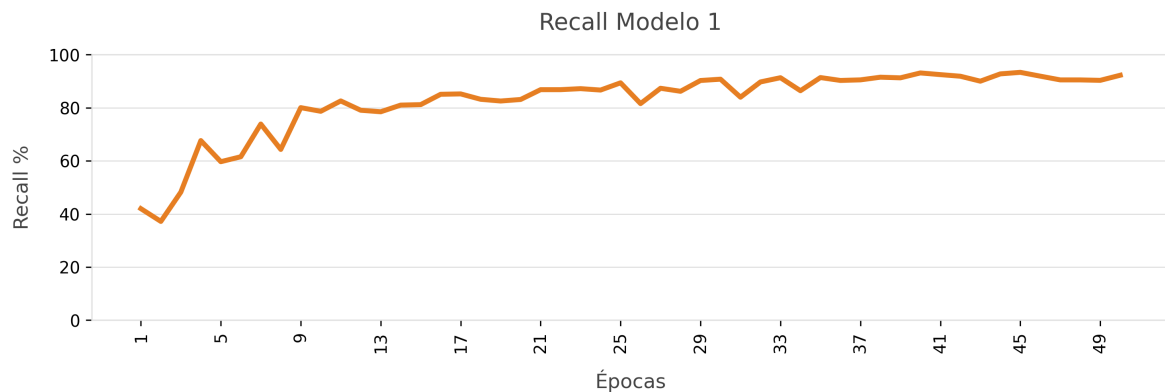


Figura 23: Gráfico de recall del Modelo 1.

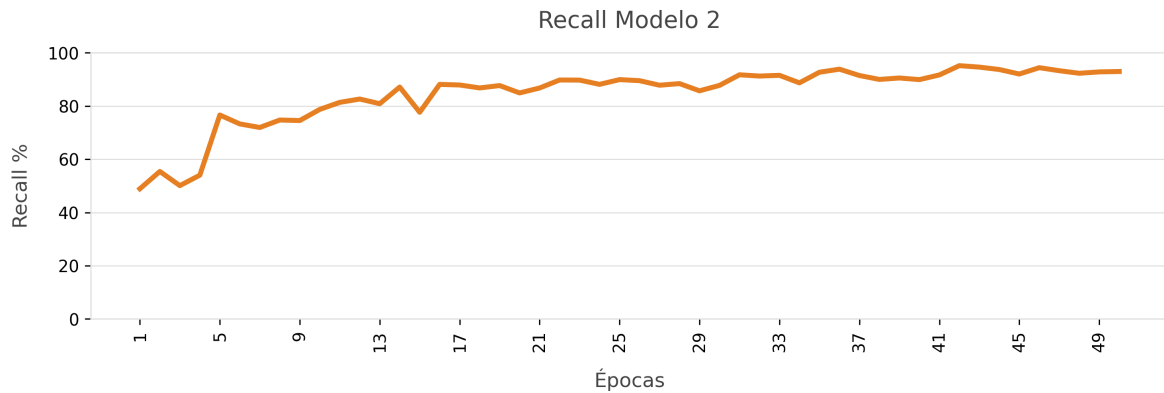


Figura 24: Gráfico de recall del Modelo 2.

En general, ambos modelos presentan un rendimiento bastante bueno en las métricas anteriores pero al momento de evaluar el mAP de los modelos se evidencia que la técnica de ponderación de pesos ayudo a mejorar el rendimiento del modelo y equilibrar el dataset:

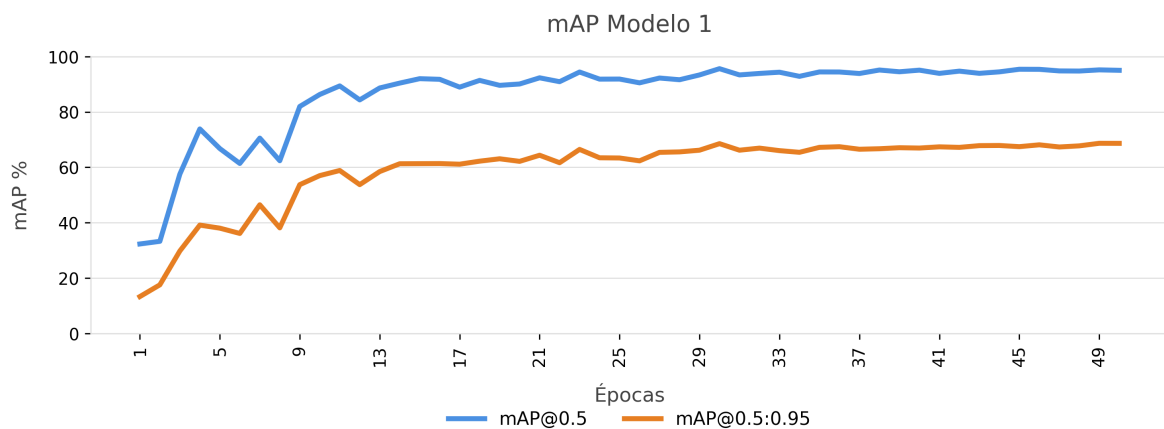


Figura 25: Gráfico de mAP del Modelo 1.

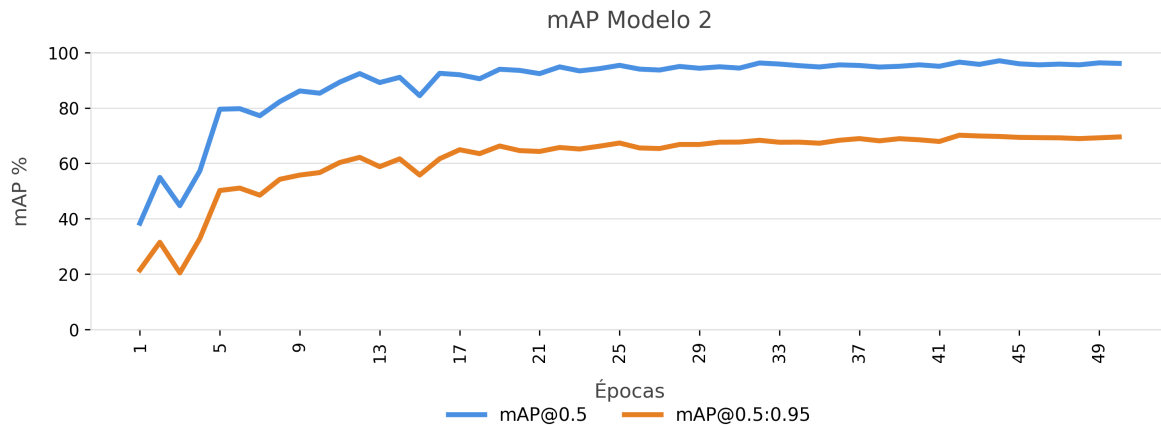


Figura 26: Gráfico de mAP del Modelo 2.

Con ambas gráficas de mAP representadas en 25 y 26 podemos extraer el mAP final para ambos modelos al final de las 50 épocas.

Métrica	Modelo 1	Modelo 2	Diferencia Comparativa
mAP50	0.952	0.965	+0.013
mAP50-95	0.686	0.701	+0.015

Tabla 12: Comparativa de mAP entre modelos.

Métrica	Clase	Modelo 1	Modelo 2	Diferencia Comparativa
mAP50	Mosca del vinagre	0.991	0.995	+0.004
	Mosca domestica	0.960	0.957	-0.003
	Mosca azul	0.951	0.961	+0.010
	Ceratitis capitata	0.965	0.972	+0.007
	Mosca Abeja	0.892	0.941	+0.049
mAP50-95	Mosca del vinagre	0.815	0.834	+0.019
	Mosca domestica	0.713	0.732	+0.019
	Mosca azul	0.755	0.762	+0.007
	Ceratitis capitata	0.662	0.670	+0.008
	Mosca Abeja	0.485	0.505	+0.020

Tabla 13: Comparativa de mAP entre modelos y clases.

Al observar los resultados se puede afirmar que la implementación de un dataloader resultó ser una buena técnica para incrementar el mAP del modelo debido al dataset desbalanceado, con efecto al mitigar el impacto del desbalance de clases en el conjunto de entrenamiento. Priorizando de mejor forma el muestreo de los insectos menos representados en cada set de imágenes de entrenamiento por época, se evitó el sesgo del modelo hacia las clases dominantes y se pudo mejorar su rendimiento. Esto optimizó la función de pérdida durante el entrenamiento, obligando a la red a aprender características más robustas y discriminativas para todas las categorías, lo que se tradujo en una mejora sustancial de la precisión general de detección evidenciable en las Tablas 12 y 13. Con estos resultados, en general el modelo 2 muestra un rendimiento bastante confiable donde su mAP50-95 es superior a 0.7 [Rainio et al. 2024].

4.3. Validación de la Aplicación

En esta sección se muestran como operan las funciones implementadas en la aplicación indicadas en la Tabla 6 sobre los requerimientos y funcionalidades esperadas, a modo de brindar una mayor transparencia sobre el funcionamiento de la aplicación implementada.

Gestión de imágenes:

El usuario sube una imagen de una trampa pegable y la envía a través de la interfaz web mediante una petición POST. El servidor valida la integridad del archivo (formato y peso) y la imagen se almacena temporalmente en un búfer de memoria utilizando la librería Pillow, se analiza mediante el modelo implementado **best.pt** (Modelo 2), le asigna un id único para evitar colisiones en el sistema de archivos y se guarda en el sistema junto con un registro en la base de datos MySQL, donde se detalla la cantidad clases detectadas, los usuarios que tienen acceso a la imagen y la ubicación de la imagen perteneciente a esa lista de trampas.

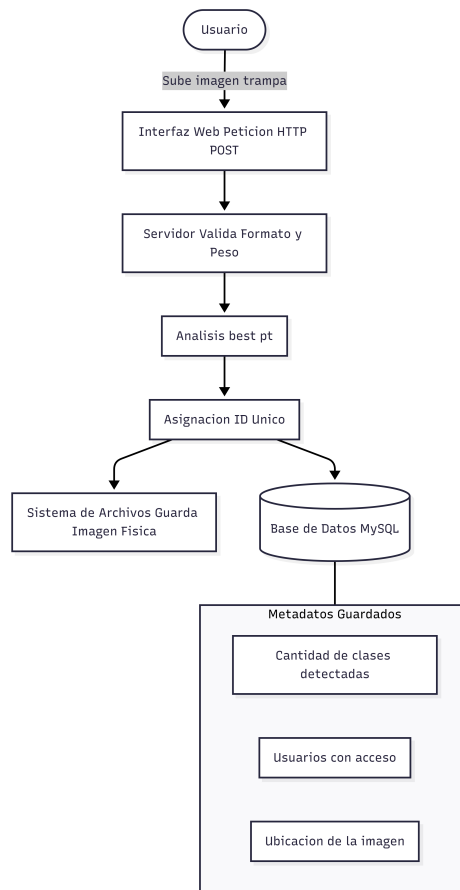


Figura 27: Carga y análisis de imágenes.

Gestión de archivos SHP (Listas de trampas):

El usuario sube un archivo comprimido con la extensión .shp que contiene la lista de trampas y sus coordenadas, enviándolo a través de la interfaz web mediante una petición POST. El servidor valida el archivo, se descomprime y procesa utilizando la librería pyshp. Se analiza y mapea su información geográfica, le asigna un id único para evitar colisiones en el sistema de archivos y se guarda un registro en MySQL, donde se detallan los puntos de las trampas extraídas, los usuarios que tienen acceso a la lista y la ubicación de las imágenes de esa lista.

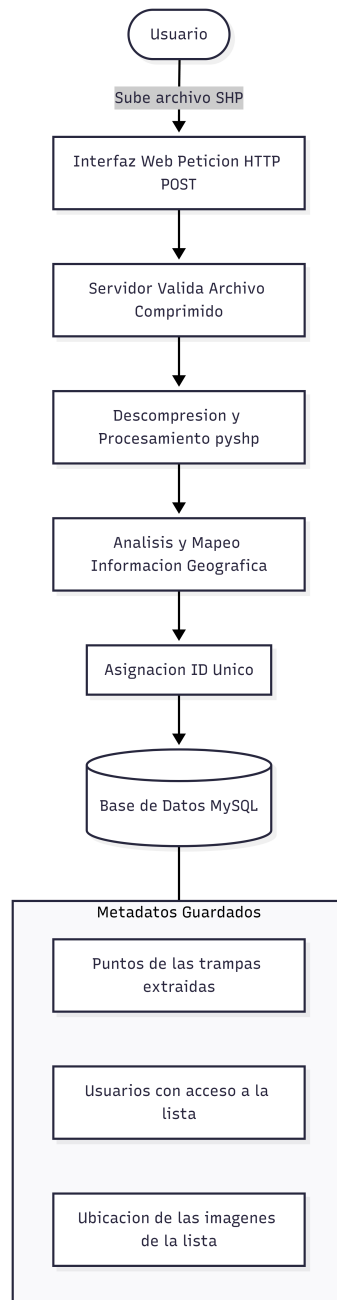


Figura 28: Carga y análisis de mapas.

Visualización e historial de imágenes analizadas:

El usuario accede al módulo de historial a través de la interfaz web, la cual emite una petición HTTP GET al servidor solicitando los registros asociados a su cuenta. El backend procesa

la consulta y realiza una búsqueda indexada en la base de datos MySQL para recuperar los datos de las capturas previamente analizadas. El sistema retorna un objeto JSON que contiene los id, las rutas de almacenamiento físico, las fechas de carga y las métricas de conteo de clases generadas por el modelo **best.pt** (Modelo 2). Finalmente, el frontend procesa esta información de forma asíncrona, renderizando una galería dinámica interactiva donde el usuario puede previsualizar las imágenes y examinar los cuadros delimitadores calculados por el sistema.

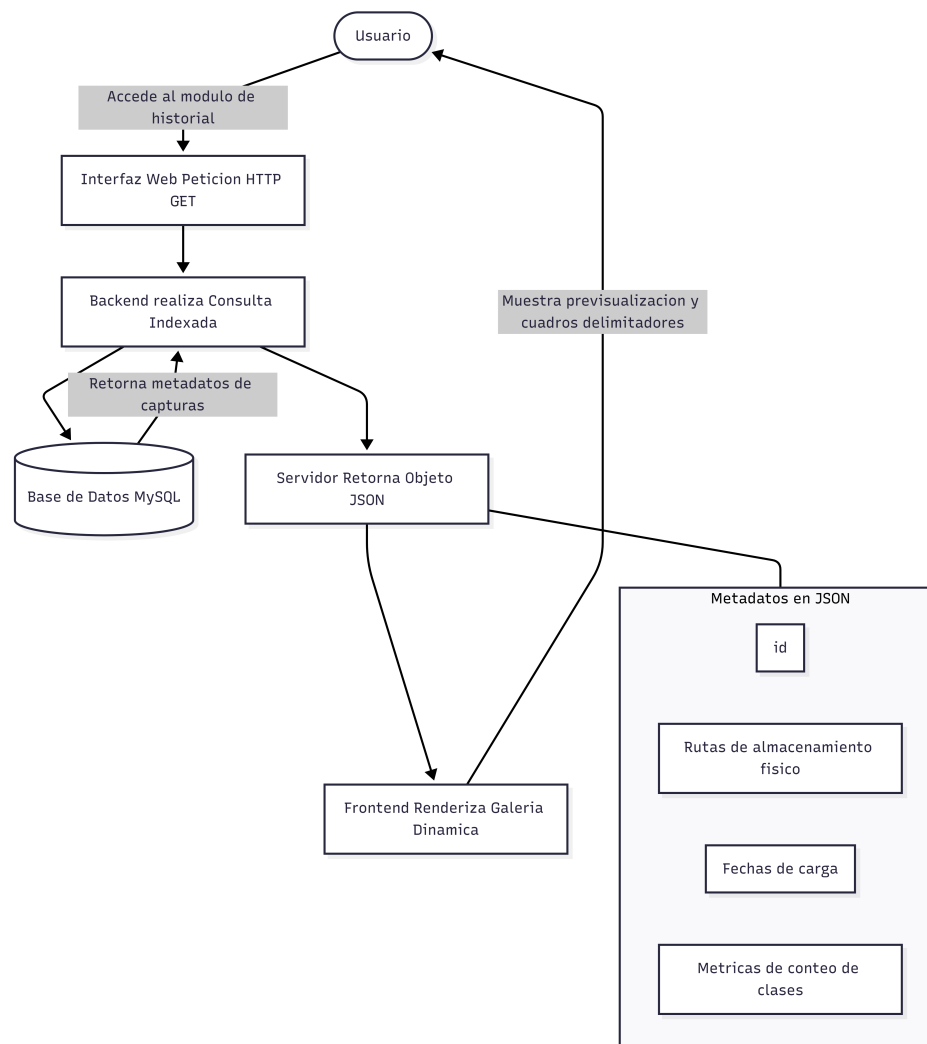


Figura 29: Historial de análisis de imágenes.

CAPÍTULO 5

VALIDACIÓN DE LA SOLUCIÓN

En el presente capítulo se muestra el funcionamiento, interfaz y pruebas de una herramienta web, el cual tiene como finalidad ser el medio por el cual las personas que trabajen en control de plagas puedan hacer un análisis (identificación, clasificación y conteo) sobre las imágenes que el usuario pueda sospechar que se trata de la mosca del mediterráneo. El usuario tiene el poder para recolectar fotos y subirlas para controlar la presencia de *Ceratitis capitata* en sus plantíos y, además, puedan diferenciar entre especies de moscas similares descritas por el SAG con tal de que el usuario tenga un cierto grado de confianza por el cual guiarse si es probable que se trate de un ejemplar de mosca de la fruta.

Se requiere que el usuario suba una imagen clara de los ejemplares a analizar con tal de ayudar a la aplicación a brindar un grado de confianza mayor, ya que si la foto no es clara, o es muy lejana, el modelo puede entregar resultados no confiables (véase Matriz de Confusión en la sección de Validación)

La interfaz de usuario es intuitiva y está diseñada para ser utilizada en condiciones de campo apoyando las operaciones agrícolas y las exportaciones en áreas rurales del país, con menús, botones, consejos y fuentes de gran tamaño, además de opciones intuitivas sin necesidad de una mayor explicación al usuario como se muestra en la Figura 30.

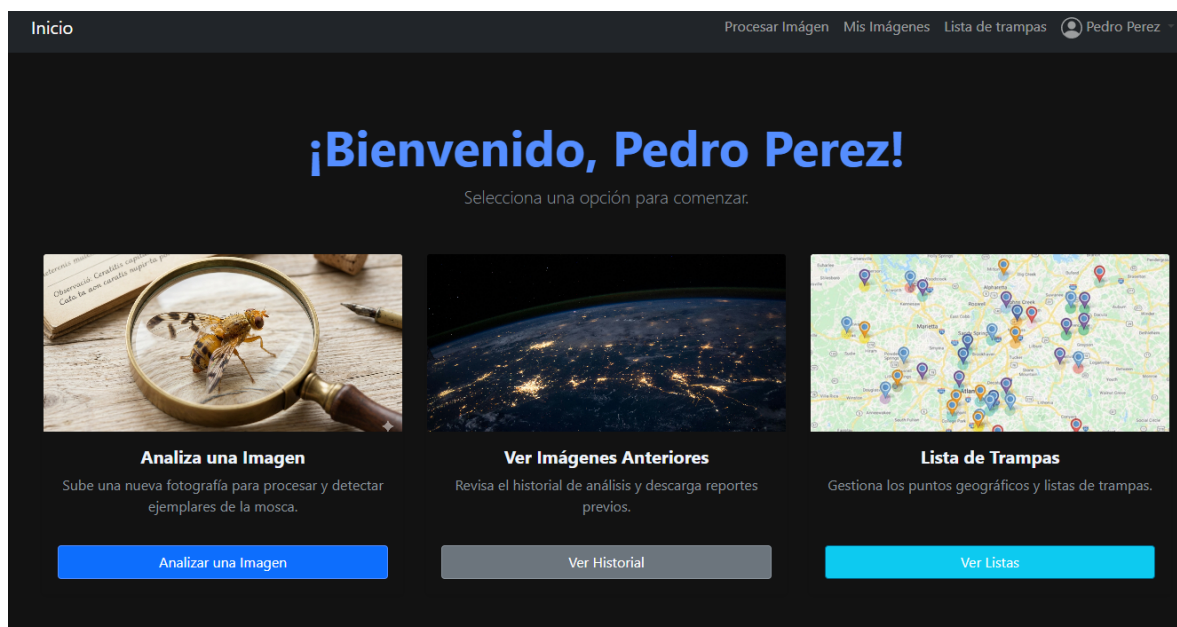


Figura 30: Página principal de la aplicación

Para explicar el funcionamiento y los requisitos antes definidos a cumplir por la aplicación, a continuación se muestran en secciones de la aplicación web desarrollada, junto con las funcionalidades esperadas que se definieron en la Tabla 6.

5.1. Funcionalidades Requeridas

5.1.1. Detección inteligente e Historial y Trazabilidad

La aplicación permite 2 formas de analizar imágenes de moscas, una pensada para usuarios normales, sin una mayor organización ni con un sistema de captura o trampeo muy complejo, los cuales simplemente pueden fotografiar un ejemplar de mosca el cual estén en sospecha, o que quieran analizar. Además, la aplicación ayuda al monitoreo al brindar una opción de localización de la fotografía, además de fecha y hora a la cual fue ingresada al sistema para su revisión con tal que el usuario pueda tener un recuerdo de donde fue tomada la foto para futuros controles, comparaciones y mapeo de la zona con tal de dar brindar futuras conclusiones como se muestra en la Figura 31 y 32.

Cada imagen es analizada y se indica el ejemplar de mosca detectado, además del grado de confianza que tiene el modelo sobre ese ejemplar y la cantidad de ejemplares detectados.

Nuevas Detecciones

Sube una fotografía de la trampa para realizar el conteo automático con IA.

Imagen

Choose File

No file chosen

Formatos aceptados: JPG, PNG. Máximo 16MB.

Locacion (optional)

Sector Norte Fundo Olivar

Subir

Resultado del Análisis Inteligente

Se identificaron 5 ejemplares en total.

Detalle por espécimen:

Clase	Confianza
Housefly	90.25%
ceratitis-capitata	87.01%
ceratitis-capitata	83.4%
ceratitis-capitata	83.34%
ceratitis-capitata	73.82%

Ver en el Historial

Figura 31: Análisis de una imagen subida por el usuario.

Página 52 de 64

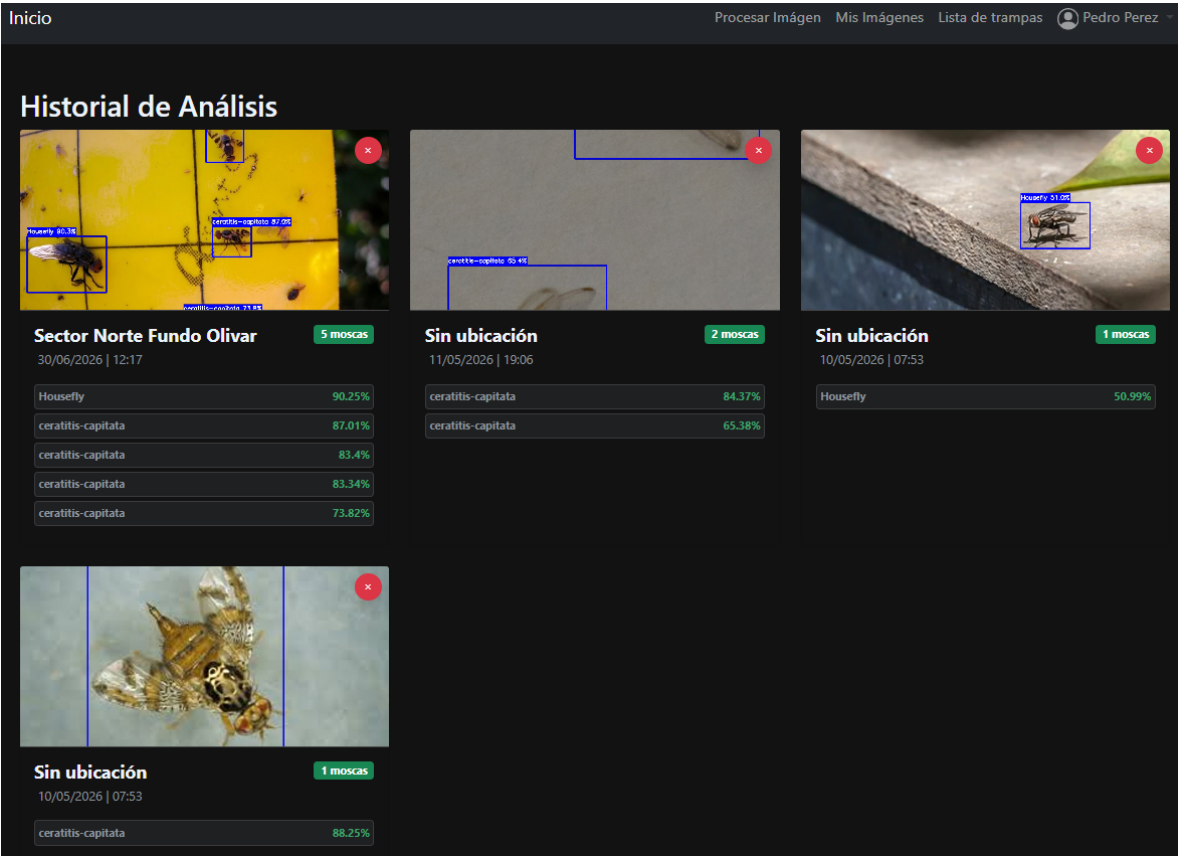


Figura 32: Historial de imágenes subidas por el usuario junto con su análisis.

La segunda forma de detección esta enfocada al análisis de trampas en especifico, donde cada punto de trampa tiene un conjunto de imágenes la cual el usuario puede subir a dicha trampa para su posterior análisis por el mismo o un usuario invitado. La interfaz indica el id de la trampa, sus coordenadas y un resumen de todas las imágenes subidas a esta trampa para que el o un usuario a distancia experto puedan analizarlas en cualquier momento. Este paso es ligado fuertemente con el requerimiento de Colaboración como se aprecia en la Figura 33.

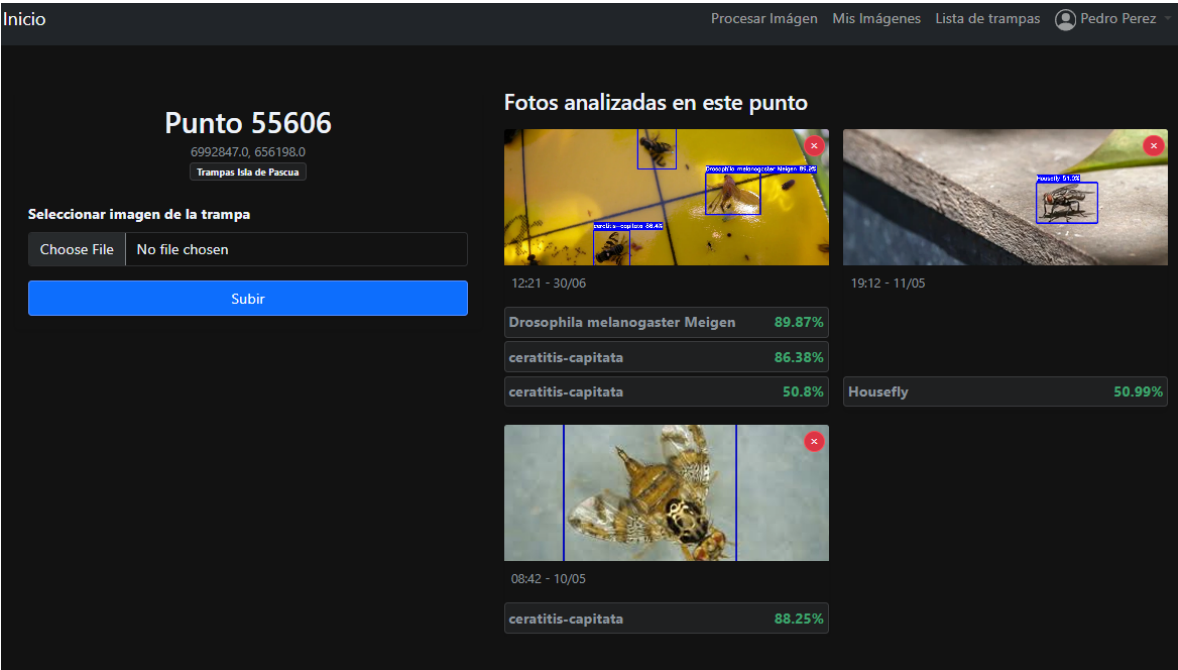


Figura 33: Historial de imágenes subidas por el usuario a un punto en específico.

5.1.2. Geolocalización y Colaboración Técnica

Además del enfoque analizado para usuarios independientes, la aplicación cuenta con una forma de que los usuarios normales o inspectores de SAG puedan subir un mapa SHP con puntos de interés (trampas) los cuales son introducidos al sistema y recordados para su posterior carga y análisis de imágenes en el punto en específico. Estos mapas pueden ser compartidos con usuarios específicos dentro de la aplicación con tal de que ambos puedan tener acceso a cada punto y a las imágenes subidas (Figura 34).

Sube una nueva lista de trampas

Shapefile ZIP

Choose File

No file chosen

El archivo ZIP debe contener al menos .shp, .shx y .dbf.

Nombre de la lista

Compartir con usuarios (separados por coma)

Separa los nombres de usuario por comas.

Upload Shapefile

Revisa Trampas

 Juan Fernandez

8 puntos

Figura 34: Interfaz para la carga de listas de trampas con usuarios invitados.

Para una mejor experiencia de los usuarios independientes y los inspectores en campo, se incluye un sistema de localización de puntos con tal ayudar la navegación y la ubicación de las trampas. Este sistema es sumamente útil debido a las medidas de prevención que se aplican alrededor de una trampa la cual fue detectada con un ejemplar confirmado de *Ceratitis capitata* (Véase Costos de una zona libre de Mosca de la Fruta). Estos puntos son fácilmente accesibles desde cualquier mapa virtual.

5.1.3. Ordenamiento por Clasificación

En lugar de recorrer los cultivos siguiendo un orden geográfico tradicional o aleatorio, los agricultores y técnicos pueden enfocar sus esfuerzos de inmediato donde el riesgo es crítico. El algoritmo ordena la evaluación de cada trampa subida a un punto y extrae el mayor porcentaje de detección que se evaluó sobre cada imagen de la trampa. Esto es clave en tiempo real para predecir qué puntos específicos requieren atención urgente, ahorrando tiempo en la revisión de cada trampa.

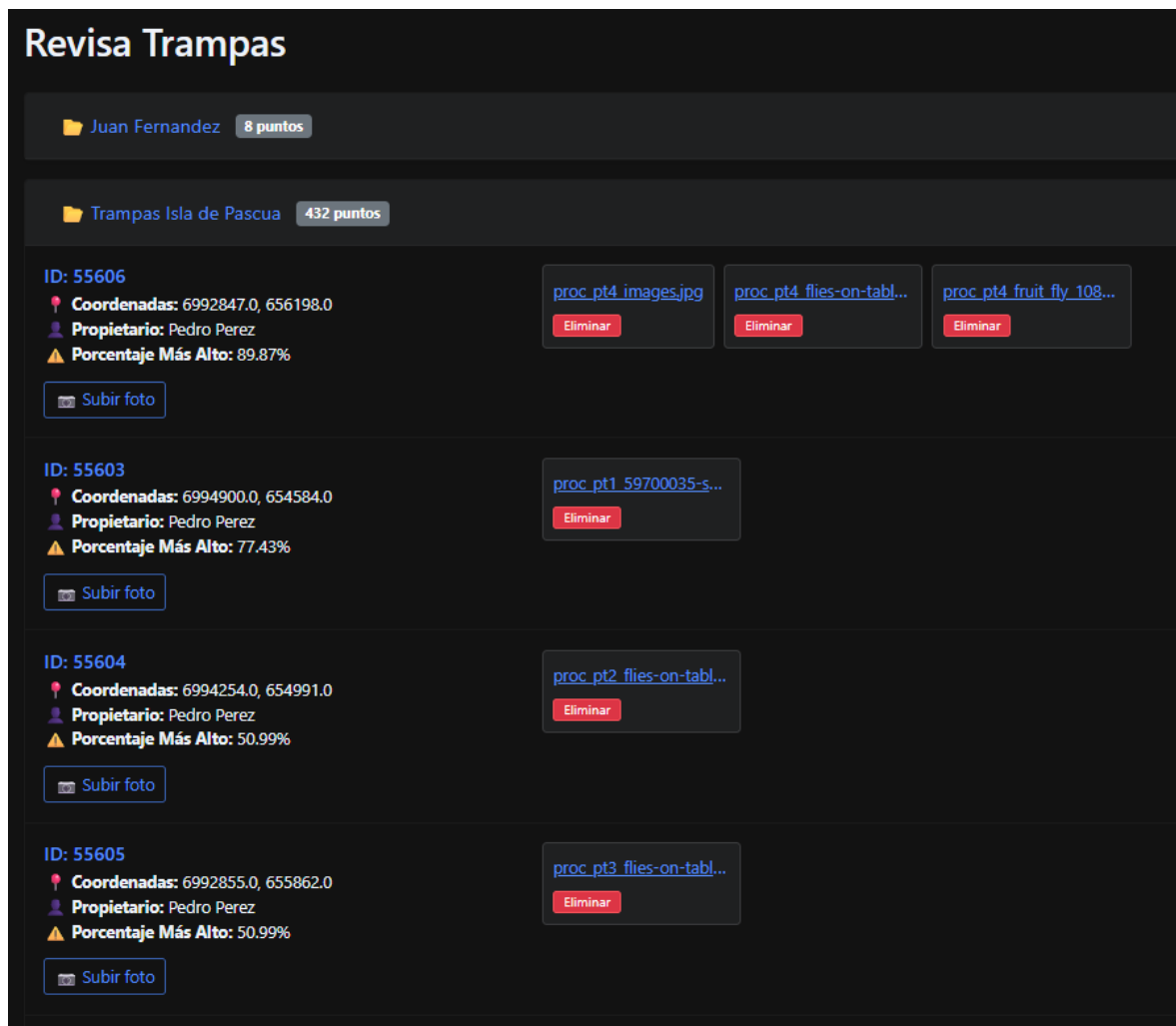


Figura 35: Interfaz de los puntos de trampas con sus imágenes subidas, ordenados.

Ademas, los usuarios invitados a esa lista de trampas pueden ver los datos, puntos, imágenes subidas y la información de la trampa, ademas de borrar imágenes o acceder a su análisis cada vez que el usuario lo requiera.

5.2. Pruebas de Humo

Se implementaron una serie de pruebas de humo utilizando el framework *unittest*¹⁰ de Python con el objetivo de validar la estabilidad del sistema antes de su despliegue técnico. Estas pruebas preliminares verificaron el correcto despliegue de la interfaz web, la carga

¹⁰<https://docs.python.org/3/library/unittest.html>

en memoria del modelo YOLOv11, la recepción de archivos mediante peticiones HTTP (asegurando códigos de estado 200) y la correcta renderización visual de las detecciones.

Para evaluar el rendimiento del backend bajo condiciones de estrés, se utilizaron las librerías *time* y *psutil* para monitorear el uso de memoria RAM y los tiempos de procesamiento durante la inferencia. Asimismo, se simuló concurrencia mediante *ThreadPoolExecutor* para analizar el comportamiento del servidor frente a solicitudes simultáneas. Los resultados, sintetizados en la Tabla 14, demostraron la alta integridad de la plataforma, registrando tiempos de respuesta óptimos de 1,5 segundos para la inicialización de la página principal y de 2 segundos para el procesamiento y despliegue del análisis predictivo de las imágenes.

Métrica	Resultado	Unidad
Tiempo de carga de la página principal	1.5	Segundos
Éxito de carga de la página principal	Éxito	.
Tiempo de carga de imagen	2	Segundos
Éxito de carga de imagen	Éxito	.
Visualización de resultados	Éxito	.

Tabla 14: Resultado de las Pruebas de Humo.

5.3. Pruebas Unitarias

Se diseñaron y ejecutaron pruebas unitarias detalladas utilizando el framework *unittest*¹¹, enfocadas en asegurar la calidad y el funcionamiento correcto de los componentes individuales del servidor backend. Estas pruebas, implementadas con la ayuda de la librería *unittest.mock* para simular interacciones externas como las consultas a la base de datos MySQL y el entorno de pruebas nativo de Flask, permitieron validar de forma aislada la lógica interna y las rutas de la aplicación antes de su integración global en el sistema.

La Tabla 15 refleja que las pruebas unitarias, ejecutadas de manera individual sobre cada componente del sistema, arrojaron resultados positivos en todas las instancias, confirmando la funcionalidad adecuada de cada elemento con un índice de aprobación del 100 %. Estas pruebas cubrieron aproximadamente el 95 % del código del sistema web, concentrándose principalmente en las funcionalidades esenciales. Con un tiempo de ejecución promedio de solo 0.8 segundos, este resultado es considerado rápido y eficiente, demostrando que cada componente opera correctamente de forma aislada

¹¹<https://docs.python.org/3/library/unittest.html>

Métrica	Resultado	Unidad
Total de pruebas	5	-
Pruebas pasadas	5	-
Pruebas fallidas	0	-
Tiempo de ejecución	0.8	Segundos

Tabla 15: Resultado de las Pruebas Unitarias.

CAPÍTULO 6

CONCLUSIONES

El objetivo principal de esta investigación fue desarrollar una herramienta capaz de detectar y agilizar el monitoreo de la mosca de la fruta en trampas de campo, empleando técnicas modernas de aprendizaje profundo y optimización de datos en condiciones reales. Para cumplir dicho objetivo, se evaluó e implementó la arquitectura YOLOv11, orientada a la detección de los especímenes dentro de las trampas. Se construyó un dataset con especies de moscas similares a la mosca de la fruta para que el modelo pudiera diferenciarlas en con el objetivo de ayudar a usuarios a diferenciarlas. Los experimentos demostraron que el modelo base alcanzaba un rendimiento inicial sólido, pero enfrentaba limitaciones críticas debido al marcado desbalance de clases en el dataset, un desafío común en el monitoreo agrícola donde la prevalencia de ciertas especies o estados de captura varía drásticamente.

Para resolver esta condición, la experimentación comparativa demostró que la combinación de técnicas clásicas de aumento de datos junto con una estrategia de ponderación de pesos a través de un Custom Dataloader superó consistentemente al enfoque convencional basado únicamente en transformaciones geométricas y fotométricas. Este modelo se implemento junto con una aplicación web que combina el procesamiento de imágenes para detectar ejemplares de *Ceratitis capitata*, además de agregar datos geográficos y una interfaz web dinámica para facilitar su monitoreo.

Durante el proceso de pruebas el sistema presento ciertas limitaciones. Estas van desde que las imágenes debían ser de cerca y con una buena iluminación, en otro caso, el modelo no reconocía de manera confiable las imágenes confundiendo con el background, como se muestra en la Matriz de Confusión. Otra limitación fue que las imágenes debían tener una resolución razonable, dependiendo de que tan grande estaban representados los ejemplares a analizar en la imagen.

Con estos requisitos, se validaron que los modelos implementaron eran altamente eficientes al detectar los ejemplares de moscas del dataset. El segundo modelo alcanzo un mAP@50 de 96,5 % y un mAP@50-95 de 70,1 % (frente al 95,2 % y 68,6 % del primer modelo, respectivamente).

En síntesis, esta memoria provee un precedente tecnológico viable y de bajo costo computacional que optimiza el flujo de trabajo de agentes de control de plagas en el contexto chileno. Al agilizar la revisión inicial de capturas y priorizar de manera inteligente el ordenamiento de trampas basado en el porcentaje de confianza. La herramienta minimiza los retrasos logísticos que actualmente pueden tardar semanas ayudando a Chile a permanecer como

país libre de la mosca de la fruta.

Como trabajo futuro, se propone aumentar el dataset de imágenes de las otras 4 clases de moscas poco representadas, ya que también un punto importante en el combate de esta plaga es la disminución de falsas alertas de este ejemplar. Otra ventaja de resolver este problema del balanceo del dataset es que se abren puertas para la utilización de otras técnicas de deep learning que tienen requisitos mas estrictos en cuanto a tamaño de imágenes o formas de entrenamiento.

ANEXOS

6.1. Link video de propuesta

<https://youtu.be/IH_KVev9mjA>

REFERENCIAS BIBLIOGRÁFICAS

- L. P. Andrade S. Manejo integrado de salivazo: Uso de trampas adherentes. *Carta Trimestral (Cenicaña)*, 29(2-3):9, apr/sep 2007. URL https://www.cenicana.org/pdf_privado/carta_trimestral/ct2007/ct2y3_07/ct2y3_07_p9.pdf.
- B. Bakırarar and A. Elhan. Class weighting technique to deal with imbalanced class problem in machine learning: Methodological research. *Turkiye Klinikleri Journal of Biostatistics*, 15: 19–29, 01 2023. doi: 10.5336/biostatic.2022-93961.
- J. Cancino and P. Montoya. Control biológico por aumento en moscas de la fruta (diptera: Tephritidae). *Folia entomologica Mexicana.*, 43, 01 2004.
- D. Carillo, J. Peña, M. Aluja, Z. Mendel, and K. Wyckhuys. *Pests and Pollinators*. 02 2026. ISBN 9781800622623. doi: 10.1079/9781800622647.0014.
- Centro de Información de Recursos Naturales (CIREN). Coloca aquí el título exacto del documento. Informe técnico / manual, CIREN, Santiago, Chile, Año de publicación (ej. 2018). URL <https://bibliotecadigital.ciren.cl/items/29cbc3d3-0310-412c-a8c6-6dd27daea217>.
- M. M. Cerezos. Control de mosca de la fruta, situación nacional. 2024.
- A. J. Dhruv, R. Patel, and N. Doshi. Python: The most advanced programming language for computer science applications. *Proceedings of the International Conference on Culture Heritage, Education, Sustainable Tourism, and Innovation Technologies*, 2020. URL <https://api.semanticscholar.org/CorpusID:234894687>.
- D. E. Jiménez. Métodos de control de plagas. UNIVERSIDAD NACIONAL AGRARIA FACULTAD DE AGRONOMÍA, 2009.
- R. Khanam and M. Hussain. Yolov11: An overview of the key architectural enhancements, 2024. URL <https://arxiv.org/abs/2410.17725>.
- M. Kibira, H. Affognon, B. Njehia, B. Muriithi, S. Mohamed, and S. Ekesi. Economic evaluation of integrated management of fruit fly in mango production in embu county, kenya. *African Journal of Agricultural and Resource Economics*, 10, 12 2015.
- C. Lobos A. Supresión erradicación y exclusión de las moscas de la fruta (diptera: Tephritidae) de chile. *Agricultura Técnica*. v. 56, 1996.
- C. Lobos A. Guía para la detección de moscas de frutas [diptera: Tephritidae]. *Servicio Agrícola y Ganadero.*, 2005.

N. Mahato, S. Dey, G. Chakraborty, and S. Debnath. Integrating artificial intelligence with insect phototaxis: Transformative innovations in pest control. *UTTAR PRADESH JOURNAL OF ZOOLOGY*, 46(2), 2025. doi: 10.56557/upjz/2025/v46i24765. URL <https://www.mbimph.com/index.php/UPJZ/article/view/4765>.

N. Martínez. Manejo integrado de plagas: una solución a la contaminación ambiental. *Comunidad y Salud*, 8(1):73–82, 2010. ISSN 1690-3293. URL <https://www.redalyc.org/pdf/3757/375740246010.pdf>.

Y. Mohammed. Balance classes during yolo training using a weighted dataloader. 2024. URL <https://y-t-g.github.io/tutorials/yolo-class-balancing/>.

OIEA. Guía para el trampeo en programas de control de la mosca de la fruta en áreas amplias. Manual técnico OIEA/FAO TG/FFP, Organismo Internacional de Energía Atómica (OIEA) / Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO), Viena, Austria, apr 2005. URL <https://www-pub.iaea.org/MTCD/Publications/PDF/7245web.pdf>.

K. O'Shea and R. Nash. An introduction to convolutional neural networks, 2015. URL <https://arxiv.org/abs/1511.08458>.

D. Quiroga and W. Ramírez, editors. *Libro de resúmenes / Book of abstracts: 9a Reunión del Grupo de Trabajo en Moscas de la Fruta del Hemisferio Occidental (9TWWH 2016)*, Buenos Aires, Argentina, oct 2016. Servicio Nacional de Sanidad y Calidad Agroalimentaria (SENASA). URL https://nucleus.iaea.org/sites/naipc/twd/Lists/Publications/Attachments/29629/libro_de_resumenes_9twwh.pdf.

O. Rainio, J. Teuho, and R. Klén. Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, 14, 03 2024. doi: 10.1038/s41598-024-56706-x.

J. P. Rodríguez Carlos. Control de la mosca de la fruta con trampas. *PlantwisePlus Knowledge Bank*, 2012. doi: 10.1079/pwkb.20137804454. URL <https://doi.org/10.1079/pwkb.20137804454>.

P. Roosjen, B. Kellenberger, L. Kooistra, D. Green, and J. Fahrentrapp. Deep learning for automated detection of drosophila suzukii: potential for uav-based monitoring. *Pest Management Science*, 76, 04 2020. doi: 10.1002/ps.5845.

D. J. Rustia, J.-J. Chao, L.-Y. Chiu, Y.-F. Wu, J.-Y. Chung, J.-C. Hsu, and T.-T. Lin. Automatic greenhouse insect pest detection and recognition based on a cascaded deep learning classification method. *Journal of Applied Entomology*, pages 1–17, 11 2020. doi: 10.1111/jen.12834.

I. H. Sarker. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2(6):420, aug 2021. doi: 10.1007/s42979-021-00815-1. URL <https://pmc.ncbi.nlm.nih.gov/articles/PMC8372231/>.

X. Ying. An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, 1168:022022, 02 2019. doi: 10.1088/1742-6596/1168/2/022022.

B. Yousefimehr, M. Ghatee, J. Fazli, S. Ghaffari, Z. Rafei, M. A. Seifi, S. Tavakoli, A. Nikahd, M. R. Gandomani, A. Orouji, R. M. Kashani, S. Heshmati, and N. S. Mousavi. Data balancing strategies: A systematic survey of resampling and augmentation methods, 2026. URL <https://arxiv.org/abs/2505.13518>.

Y. Zhong, J. Gao, Q. Lei, and Y. Zhou. A vision-based counting and recognition system for flying insects in intelligent agriculture. *Sensors*, 18:1489, 05 2018. doi: 10.3390/s18051489.