EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Maestría

Sistema de identificación de plagas en imágenes de plantas mediante aprendizaje profundo

presentada por

Ing. Brayan Alejandro Pizano Martínez

como requisito para la obtención del grado de

Maestro en Ciencias de la Computación

Director de tesis

Dra. Andrea Magadán Salazar

Cuernavaca, Morelos, México. Enero de 2024.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

SEP TecNM CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
RECIBIDO
08 DIC 2023
LMZ
SUBDIRECCIÓN ACADÉMICA

Cuernavaca, Mor., 04/noviembre/2023

OFICIO No. DCC/206/2023
Asunto: Aceptación de documento de tesis
CENIDET-AC-004-M14-OFICIO

CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial de BRAYAN ALEJANDRO PIZANO MARTÍNEZ con número de control M21CE065, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis de grado titulado **"SISTEMA DE IDENTIFICACION DE PLAGAS EN IMÁGENES DE PLANTAS MEDIANTE APRENDIZAJE PROFUNDO"** y hemos encontrado que se han atendido todas las observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

ANDREA MAGADÁN SALAZAR
Directora de tesis

RAÚL PINTO ELÍAS
Revisor 1

JORGE ALBERTO FUENTES PACHECO
Revisor 2

C.c.p. Depto. Servicios Escolares.
Expediente / Estudiante

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
08 DIC 2023
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
DEPARTAMENTO DE SERVICIOS ESCOLARES
RECIBIDO
EBN

Cuernavaca, Mor., 11/diciembre/2023
No. De Oficio: SAC/201/2023
Asunto: Autorización de impresión de tesis

BRAYAN ALEJANDRO PIZANO MARTÍNEZ
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"SISTEMA DE IDENTIFICACIÓN DE PLAGAS EN IMÁGENES DE PLANTAS MEDIANTE APRENDIZAJE PROFUNDO"**, ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"Conocimiento y tecnología al servicio de México"

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SUBDIRECCIÓN ACADÉMICA

CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO

C. c. p. Departamento de Ciencias Computacionales
Departamento de Servicios Escolares

CMAZ/lmz

DEDICATORIA

En primer lugar, agradezco a Dios, quien ha puesto en mi vida a personas importantes que me han apoyado en mi crecimiento profesional, y darme la capacidad de afrontar, aprender y desarrollarme en diversas áreas en todo este tiempo, lo cual se ha vuelto valioso en mi vida.

Entre las personas que han contribuido de manera significativa se encuentra mi familia, mi mejor amiga y mis profesores, quienes han sido un apoyo incondicional en cada paso de mi trayectoria, para ser quien soy hoy en día.

AGRADECIMIENTO

Expreso mi agradecimiento al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el respaldo financiero proporcionado a lo largo de mi periodo de estudios de posgrado mediante una beca otorgada. Reconozco al Tecnológico Nacional de México, específicamente al campus Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), por brindarme las facilidades necesarias para el desarrollo de mi desarrollo profesional.

Quiero extender mi gratitud a la Dra. Andrea Magadán Salazar, quien desempeñó el papel de directora de tesis en este trabajo. Agradezco su apoyo, orientación y valiosas enseñanzas durante la ejecución de esta investigación.

Asimismo, agradezco a los miembros de mi comité revisor, el Dr. Raúl Pinto Elías y el Dr. Jorge Alberto Fuentes Pacheco, por sus pertinentes observaciones y sugerencias que contribuyeron significativamente a la mejora de esta tesis.

RESUMEN

Actualmente, el aprendizaje profundo tiene un papel importante debido a sus diversos campos de aplicación, entre ellos la agricultura que es un pilar en el desarrollo económico. Este trabajo de investigación se centra en la identificación de insectos en imágenes considerados como plagas en los cultivos. La detección rápida y correcta brinda a los agricultores, la posibilidad de la eliminación de la plaga, lo cual evita la pérdida completa de sus cultivos.

Se entrenaron tres arquitecturas de redes neuronales las cuales son: YOLOv5m6, YOLOv5L6 y YOLOv7; usando el conjunto de datos de *IP102* con 97 clases para entrenamiento y pruebas. Cada modelo obtuvo un valor de 0.658, 0.619 y 0.635 en *mAP*@0.5 respectivamente.

Los modelos obtenidos fueron ensamblados para operarse de manera conjunta en la computadora obteniendo un *mAP* de 0.680. También se tomó el mejor modelo para optimizarlo y ser utilizado en un dispositivo móvil con sistema operativo Android, pudiendo ser ejecutado en tiempo real usando fotografías.

ABSTRACT

Deep learning plays vital role because it has various fields of application, including agriculture, which is a pillar in economic development. This research work focuses on the identification of insects considered pests of crops. Rapid and correct detection allows farmers to eliminate the pest, which prevents the complete loss of their crops.

Three neural networks YOLOv5m6, YOLOv5L6 and YOLOv7, were trained on an IP102 data set with 97 classes for training and testing. Each model obtained mAP@0.5 value of 0.658, 0.619 and 0.635, respectively.

The models obtained were assembled to run together on the computer, getting a *mAP* of 0.680. Also, the best model was optimized be used on an Android mobile device that can be executed in real-time and through photographs.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ACRÓNIMOS

ANN	Artificial Neural Network
AP	Average precision
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Networks
DL	Deep Learning
FAO	Food and Agriculture Organization
FT	Fine-tuning
HOG	Histogram of oriented gradients
IOU	Intersection Over Union
KNN	K-nearset neighbor
LCP	Local Configuration Pattern
mAP	Mean average precision
MKL	Multiple kernel learning
ML	Machine Learning
NB	Naive Bayes
NMS	Non-maximun suppression
NMW	Non-Maximum Weighted
ONUAA	Organización de las Naciones Unidas para la Alimentación y la Agricultura
ROC	Receiver operating characteristic curve
ROIs	Region of Interest
SIFT	Scale Invariant feature transform
SLIC	Simple Linear Iterative Clustering
SSD	Single-Shot Detector
SUN	Saliency Using Natural statistics
SVM	Support vector machine
SVM	Support vector machines
TL	Tranfer Lerning
UAV	Unmanned Aerial Vehicle
WBF	Weighted Boxes Fusion
WRN	Wide Residual Network
XML	Extensible Markup Language
YOLO	You Only Look Once

CAPÍTULO 1

Introducción

La agricultura es la fuente alimenticia de los seres humanos, el sustento económico de algunos países, así como de agricultores, por lo tanto, las personas son dependientes de que se pueda proporcionar suficiente volumen para abastecer la demanda. Unos de los factores que afectan a la calidad y productividad de los cultivos son los insectos catalogados como “plagas”. Los agricultores presentan dificultades en la identificación de los insectos debido a que suelen confundir entre las especies o con insectos benéficos por carecer de conocimientos sobre el área o no contar con un experto en el momento.

En la actualidad, existen métodos como el aprendizaje profundo que son sobresalientes en la localización de objetos y su clasificación en las imágenes, comparados respecto a las técnicas tradicionales utilizadas en el tratamiento de imágenes, brindando una buena solución para este tipo de problemas.

En el presente proyecto de tesis se tiene planteado el desarrollo de un sistema que, mediante el análisis de una imagen, por medio de aprendizaje profundo, localice y clasifique de manera automática un cierto tipo de insectos denominados como “plagas”, permitiendo al agricultor tomar medidas para el cuidado de sus cultivos.

1.1 Descripción del problema

En el área de la agricultura existen factores que afectan a los cultivos entre los que resaltan las plagas de insectos, estos dan lugar a una disminución de producto, lo que significa pérdidas económicas y de calidad en la producción. Debido a que para controlar a la plaga se requiere de una rápida identificación para el seguimiento adecuado.

Las plagas de los insectos pueden presentar características visuales similares por lo que son fácilmente confundidos entre ellos y utilizar visión artificial con extracción manual de características para determinar la especie, puede resultar una tarea laboriosa. En cambio, el aprendizaje automático combinado con la visión por computadora es especialmente bueno en la clasificación de imágenes por lo que un modelo que clasifique la plaga resulta beneficioso para los agricultores. En esta investigación el problema fue realizar la identificación de un conjunto de plagas en un entorno natural de manera automática por medio de un modelo de aprendizaje profundo.

1.1.1 Objetivo General

Proponer y desarrollar un sistema que mediante aprendizaje profundo realice la identificación de insectos nocivos en imágenes de cultivos.

1.1.2 Objetivos Específicos

1) Estudiar las características visuales de las plagas de insectos seleccionados y sus diferentes etapas de vida en los cultivos agrícolas.

2) Analizar las técnicas de aprendizaje profundo e implementar la mejor solución para el problema propuesto, con la finalidad de poder reconocer si el cultivo contiene alguna plaga.

3) Evaluar el rendimiento del sistema con las métricas clásicas y compararlo con otros modelos similares.

1.1.3 Alcances y Limitaciones

1.1.3.1 Alcances

- La identificación considera imágenes de insectos en un ambiente natural.
- El sistema desarrollado es robusto a cambios en la escala, perspectiva u orientación del insecto.
- El sistema tiene la posibilidad de identificar los insectos presentes en el conjunto de datos seleccionado.

- Las plagas consideradas se pueden presentar en cultivos de la región de Morelos y Tamaulipas.

1.1.3.2 Limitaciones

- El sistema no realiza la identificación de los daños de la plaga en las plantas o las enfermedades que produce, el sistema está enfocado sólo al reconocimiento del insecto.

1.2 Marco Conceptual

En esta sección se describen los temas y conceptos relacionados a esta investigación, con la finalidad de establecer las definiciones que se emplean a lo largo del trabajo.

1.2.1 Agricultura y plagas

México es el duodécimo productor de comida y el undécimo en la producción agrícola además de ser la principal fuente de trabajo y economía para algunas familias (*¿Cómo Beneficia La Agricultura a Las Familias Mexicanas? | Secretaría de Agricultura y Desarrollo Rural | Gobierno | Gob.Mx,* n.d.). Actualmente el 30.7% de la población mundial se dedica a la agricultura en un área de 2,781 millones de hectáreas, en donde uno de los problemas que afectan a la agricultura reportado por la Organización de las Naciones Unidas para la Alimentación y la Agricultura ONUAA o FAO (por sus siglas en inglés, *Food and Agriculture Organization*) estima que cada año las plagas destruyen entre el 20% y el 40% de la comida en el mundo, anualmente las plagas invasoras y las enfermedades de las plantas cuestan a la economía mundial un estimado de entre 70 mil millones y 220 mil millones de dólares respectivamente (FAO, 2021).

A los insectos que se les denomina “plaga” son aquellos cuyas poblaciones causan daños a los cultivos y se encuentran presentes en los sistemas manejados por el hombre, causando también daños a los ecosistemas naturales. Generalmente, los ecosistemas naturales cuentan con factores limitantes que afectan en el crecimiento descontrolado de la población de las plagas (Colorado & Colorado-Formosa, n.d.).

1.2.2 Aprendizaje profundo

Permite que los modelos informáticos procesen distintas clases y representen datos en múltiples niveles de abstracción, simulando cómo el cerebro percibe y comprende la información multimodal y captura implícitamente estructuras de datos complejas (Voulodimos et al., 2018).

Su funcionamiento consiste en considerar una gran cantidad de datos para tomar decisiones sobre nuevos datos introducidos.

Tradicionalmente, los enfoques del aprendizaje automático extraen características manualmente y posteriormente, un algoritmo de clasificación los etiqueta en categorías, realizándose dos procesos separados. En cambio, el aprendizaje profundo combina ambas partes y no quedan los datos a interpretación del usuario (Krishna & Kalluri, 2019) como se muestra en la Figura 1.1.

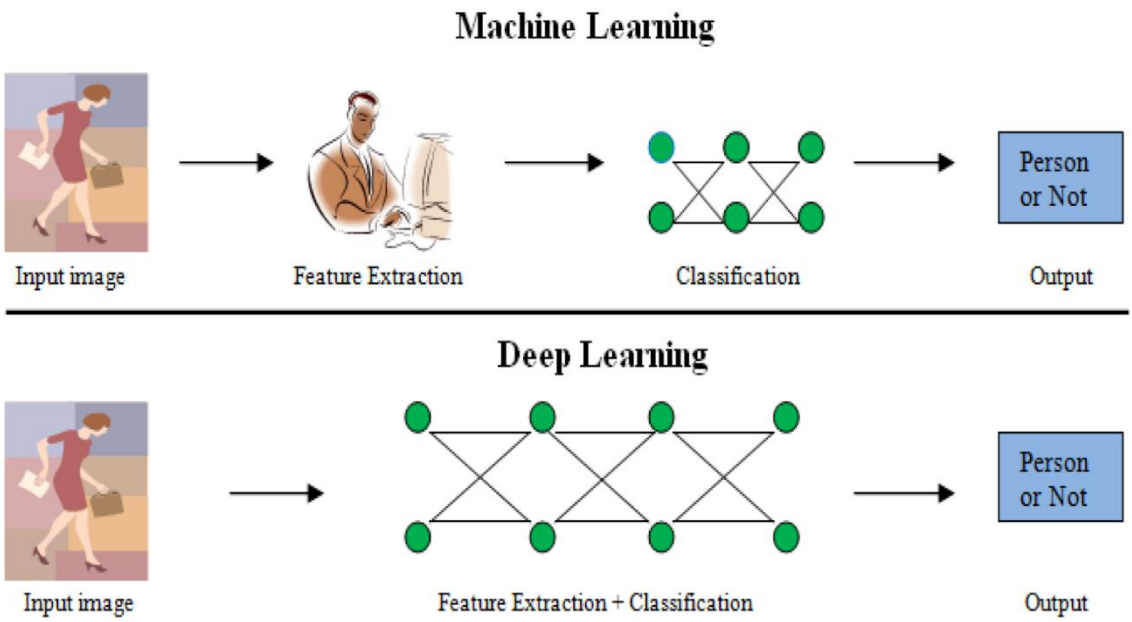


Figura 1.1 Aprendizaje automático vs Aprendizaje profundo (Krishna & Kalluri, 2019).

1.2.3 Arquitectura YOLO

La familia YOLO (por sus siglas en inglés, *You Only Look Once*), son algoritmos de detección de objetos que trabajan como regresión. El sistema divide las imágenes en una cuadricula donde cada celda tiene la tarea de encontrar las coordenadas del cuadro delimitador y las probabilidades de la clase del objeto presente en la imagen.

A continuación, se describe las versiones de YOLO utilizadas:

1.2.3.1 YOLOV5

Yolov5 fue desarrollado por Glenn Jocher en 2020 (Jocher et al., 2022), quien cambió el marco que la familia YOLO había estado usando. Es decir, pasando de Darknet, un marco de

bajo nivel, a *Pytorch* para detectores de objetos escritos en el marco *Pytorch*, que es una biblioteca para aprendizaje automático. Aunque este cambio ha sido criticado por algunos, es un marco más amigable, con un código más legible (Jiang et al., 2021). En general, YOLOv5 presenta cinco versiones, YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l y YOLOv5x donde estas difieren en variaciones de escala (profundidad y ancho) de la misma arquitectura siguiendo el modelo de escalado compuesto de *EfficientDet* (*Differences between YOLOv5 Models · Issue #7152 · Ultralytics/Yolov5 · GitHub*, n.d.). La estructura básica de YOLOv5 se muestra en la Figura 1.2 (*Architecture Summary - Ultralytics YOLOv5 Docs*, n.d.).

En general esta versión consiste en tres tipos de pérdidas: (*Architecture Summary - Ultralytics YOLOv5 Docs*, n.d.)

- Pérdida de clases (Classes loss la cual la nombran como BCE loss): da una idea de cómo un algoritmo puede predecir correctamente la clase de un objeto determinado y calcular la calibración correspondiente.
- Pérdida de objetividad (Objectness loss también conocida como: BCE loss): el valor corresponde a la confianza de que se encuentra un objeto en la región. Se obtienen resultados altos cuando se encuentra un objeto.
- Pérdida de ubicación (conocida como Location loss o CIoU loss): Es la pérdida del cuadro predictivo. Muestra el error en el centro de un objeto y el cuadro delimitador predicho cubre el objeto predicho.

Yolov5 se divide en cuatro partes:

- Entrada: incluye un nuevo mosaico de datos con la imagen original y 3 imágenes aleatorias, esto puede ser efectivo en un objetivo pequeño.
- Backbone: una red neuronal convolucional para extraer y formar características de imagen con varios detalles.
- Cuello: es un conjunto de capas para mezclar y combinar características de la imagen para pasar a la predicción.
- Predicción: consume características del cuello para completar los resultados tomando pasos de *predicción de cuadro y clase*. Aquí se usa el método de regresión CIOU_LOSS y para la supresión no máxima (NMS), el NMS es ponderado.

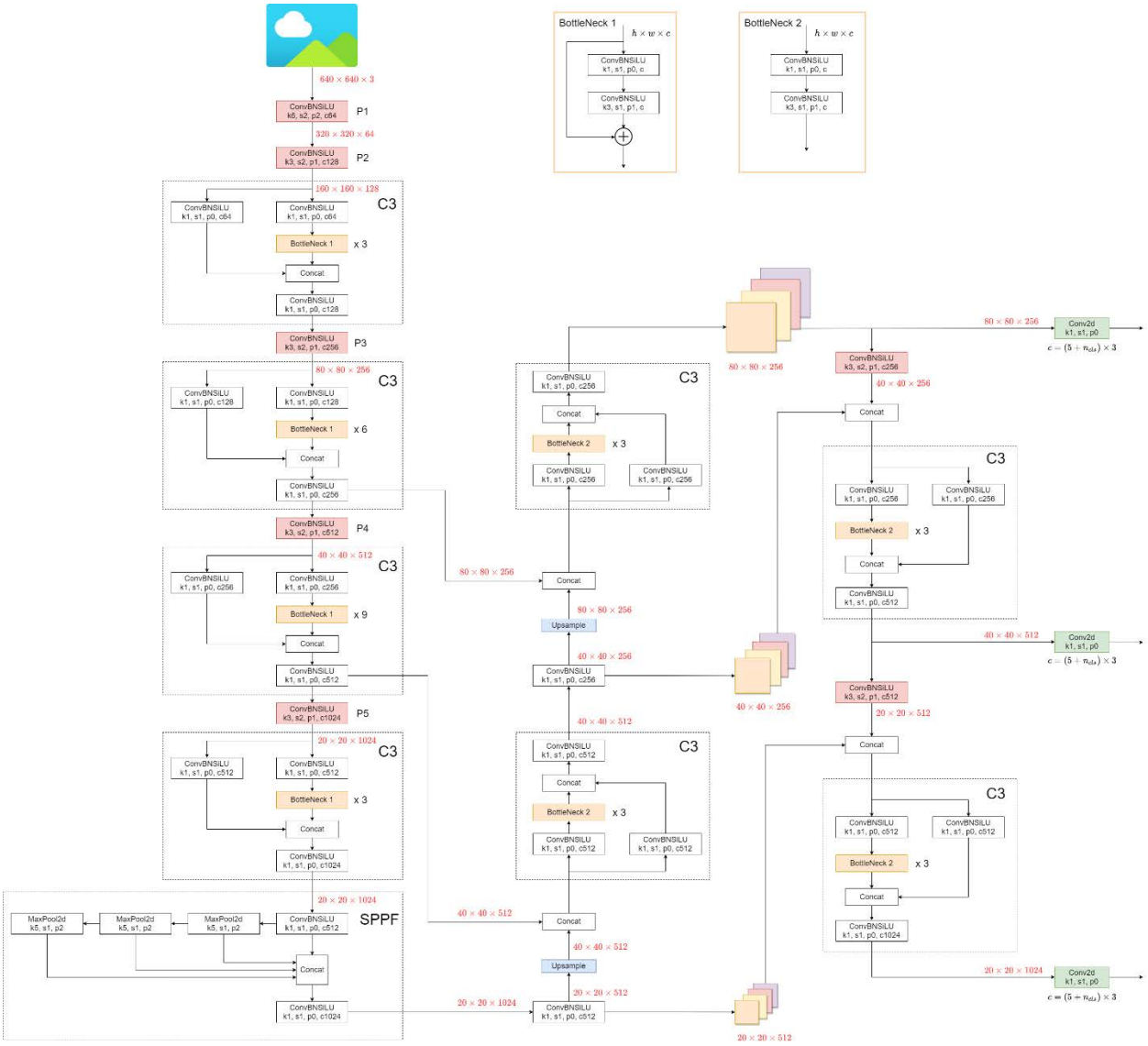


Figura 1.2 Arquitectura de YOLOv5L6 (Architecture Summary - Ultralytics YOLOv5 Docs, n.d.)

1.2.3.2 YOLOV6

Fue implementado por Li C et al., en 2022 (C. Li et al., 2022) y la versión continúa con el cambio en el *framework* con *Pytorch*. Además, implementó otra asignación etiquetas, funciones de pérdidas y técnicas de aumento de datos y presenta una estrategia de auto destilación. Cuenta con 6 versiones: YOLOv6-N, YOLOv6T, YOLOv6-S, YOLOv6-M, YOLOv6-L-ReLu, YOLOv6-L en las cuales se incrementa el número de parámetros, además de cambios en los módulos. Una de las mejoras que agregaron es un aprendizaje de destilación que consiste en utilizar un modelo maestro para que se entrene a un modelo estudiante que es una red más pequeña e intente replicar sus salidas en varios niveles y no solo en la pérdida final como se muestra en la Figura 1.3. La Figura 1.4, muestra la arquitectura general de YOLOv6.

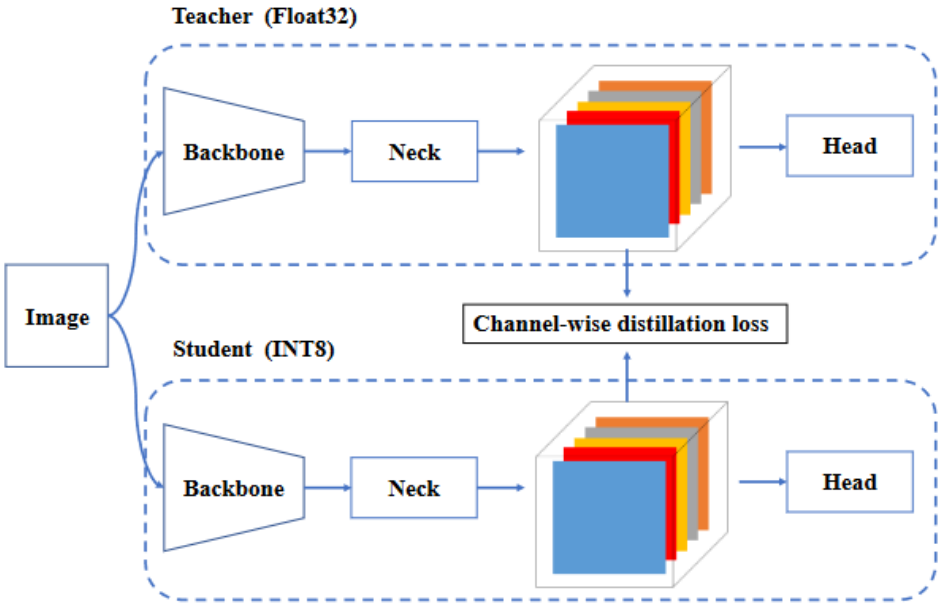


Figura 1.3 Esquemático de aprendizaje de auto destilación (C. Li et al., 2022)

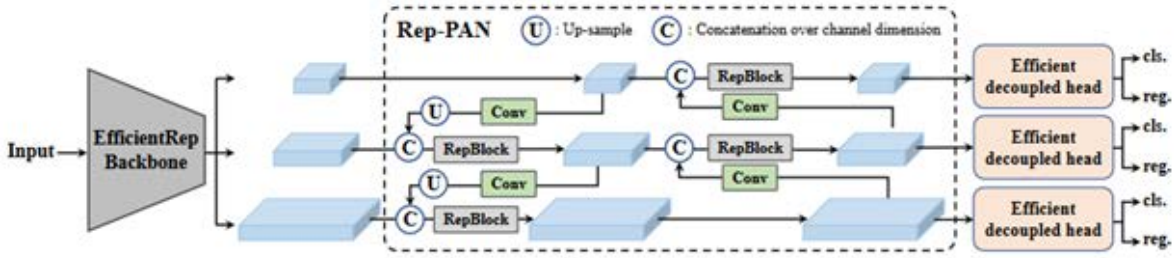


Figura 1.4 Arquitectura de YOLOv6 en su versión n y s (C. Li et al., 2022)

La pérdida de YOLOv6 consta de dos partes

- Pérdida de la clasificación (*VariFocal Loss*): trata ejemplos positivos y negativos de manera diferente, con esto logra equilibrar las señales de aprendizaje en ambos casos.
- Pérdida de regresión (*SIoU/GIoU*): en donde utilizan SIOU para versiones N y T, y trata de contemplar la dirección entre las cajas predictoras con respecto a la caja verdadera. El resto de las versiones utilizan la GIOU que resuelve el problema de IOU cuando los cuadros delimitadores no se superponen (no crean la intersección).

En general yolov6 está constituido por:

- Columna: Utiliza un método de re-parametrización llamado *EfficientRep* que consiste en usar dos tipos de columna. En los modelos pequeños utiliza *RepBlock* durante el entrenamiento y *RepConv* blocks para la inferencia, para modelos medianos y grandes utiliza *CSPStackRep* block. Este enfoque busca obtener la mayor precisión y mayor velocidad en la inferencia.
- Cuello: Adoptan la topología de YOLOv4 y YOLOv5 para la detección de objetos en distintas escalas.
- Head: Utiliza la misma que en YOLOv5 que consiste en trabajar por separado la clasificación y la detección por lo que no comparten parámetros con el fin de reducir el tiempo de cálculo y proporcionar una mayor precisión.

1.2.3.3 YOLOv7

Fue lanzado por los mismos creadores de YOLOV4 (Bochkovskiy et al., 2020) y YOLOR (C.-Y. Wang et al., 2021) en julio del 2022 (C.-Y. Wang et al., 2022) . Fue entrenado con el conjunto de datos de MS COCO, su arquitectura en la versión básica se muestra en la Figura 1.5. Actualmente esta versión cuenta con distintos modelos que son YOLOv7, YOLOv7-X, YOLOv7-W6, YOLOv7-E6, YOLOv7-E6, YOLOv7-D6 y YOLOv7-E6E. A diferencia del resto de detectores en donde la serie de sus modelos varían en la cantidad de épocas con la que fue entrenado, aumentando y disminuyendo su tamaño para aplicarse en distintas aplicaciones, los autores escalan en la profundidad y también concatenan capas de convolución. Además, agregaron una cabeza auxiliar que no tiene el mismo entrenamiento como la principal, su característica es estar en un punto intermedio de la red (no cuenta con toda la red como la principal) pero sirve como una asistencia en la pérdida para guiar la predicción.

La versión 7 se puede resumir como:

- Columna: ELAN, E-ELAN, que significa *efficient layer aggregation networks* y *Extended efficient layer aggregation networks* en donde un grupo de convoluciones aumenta la cardinalidad agregando y combinando características, cambiando la arquitectura que se había estado usando en el resto de las versiones.
- Cuello: CSPSPP + (ELAN, E-ELAN) PAN, en donde el primer módulo se encarga de concatenar mapas de características generadas y, el segundo módulo combina características de escalas diferentes generando un mapa de características intermedio.
- Head: Utiliza el conocimiento implícito de YOLOR, este vector combina los pesos y las capas anteriores.

Por último, en la Tabla 1.1 se presenta una comparativa de las diferentes versiones de arquitecturas YOLO en capas y parámetros.

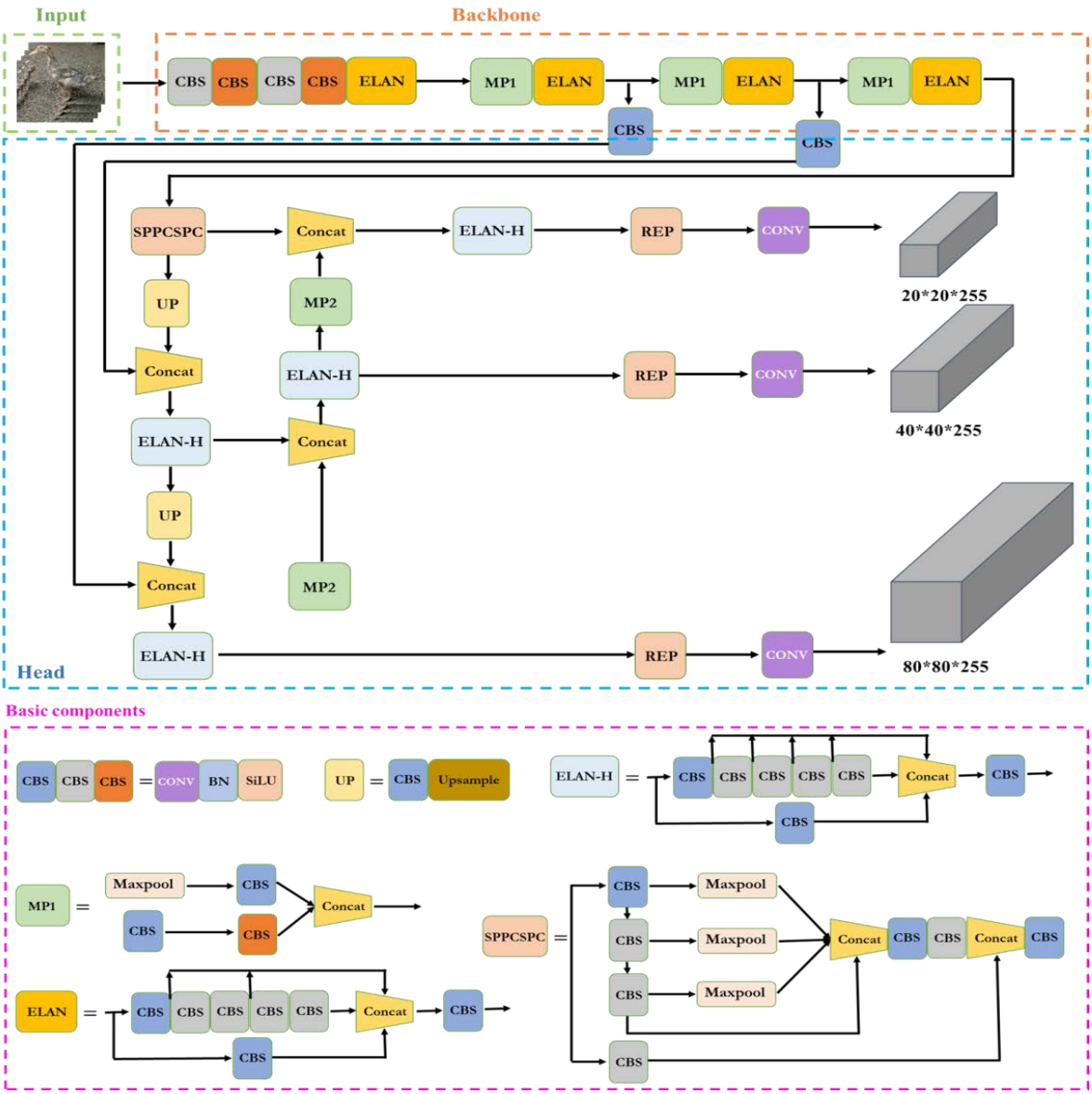


Figura 1.5 Diagrama de la arquitectura de red YOLOv7 (C.-Y. Wang et al., 2022)

Tabla 1.1 Comparativa de capas y parámetros de las versiones de YOLO.

Modelo	Capas	Parámetros
YOLOv5n	213	1,867,405
YOLOv5n6	280	3,239,884
YOLOv5s	213	7,225,885
YOLOv5s6	280	12,612,508
YOLOv5m	290	21,172,173
YOLOv5m6	378	35,704,908
YOLOv5l	367	46,533,693
YOLOv5l6	476	76,726,332
YOLOv5x	444	86,705,005
YOLOv5x6	574	140,730,220
YOLOv6n	275	4,662,410
YOLOv6n6	354	10,373,970
YOLOv6s	275	18,567,978
YOLOv6s6	354	41,394,482
YOLOv6m	432	34,855,958
YOLOv6m6	577	79,640,114
YOLOv6l	526	59,607,563
YOLOv6l6	703	140,359,342
YOLOv7	407	37,620,125
YOLOv7-w6	446	70,426,236
YOLOv7x	459	71,344,389
YOLOv7-e6	614	97,246,652
YOLOv7-d6	702	133,810,044
YOLOv7-6e6	1,032	151,757,244

1.2.4 Ensambles de modelos de detección

Generalmente los modelos de detección trabajan con cuatro puntos que describen un cuadro delimitador que puede ser expresado de distintas maneras según la arquitectura. En las arquitecturas de YOLO este cuadro es descrito a partir de *x, y, w* y *h* donde *x* y *y* es el centro de cuadro delimitador *w* y *h* representan el ancho y largo del cuadro. También se regresa la información de la clase a la que pertenece el objeto, así como su valor de confianza. Para mejorar la detección se pueden mezclar estas cajas predictoras de la siguiente manera:

a) Supresión no máxima NMS (del inglés, *Non-maximun suppression*) (Neubeck & Van Gool, 2006)

Para cada una de las cajas que pertenecen a un solo objeto, si la intersección sobre la unión de las cajas (IOU) es mayor a un umbral que se establece para IOU, la filtración dependerá de la probabilidad más alta, que en este caso será la confianza del modelo descartando el resto de las cajas La determinación de un umbral es una tarea compleja debido a que si no se eligen los valores correctos, se pueden eliminar objetos que si pertenecen.

b) Supresión No Máxima Suave (soft-nms por sus siglas en inglés, *Soft-Non-Maximum Suppression*) (Bodla et al., 2017)

Trabaja de manera similar a NMS al seleccionar la caja con la puntuación máxima, pero a diferencia de esa técnica que elimina por completo las detecciones, reduce el valor de la confianza de las predicciones propuestas (que cumplan con un valor IoU significativo) en esa superposición, esto permite preservar las cajas verdaderas parcialmente superpuestas.

c) Ponderado No Máximo (NMW en inglés, *Non-Maximum Weighted*) (Ning et al., 2017)

En métodos como NMS y soft-NMS no se utilizan todas las casillas al excluirlas de cierta manera. Sin embargo, *Non-Maximun Weighted* utiliza pesos para ponderar las cajas delimitadoras superpuestas utilizando todas las detecciones y, el valor que resulta en IOU (del inglés, *Intersection Over Union*) sirve para establecer valor o peso de la caja, tomando como base la caja delimitadora con la confianza más alta obtenida en esa superposición.

d) Fusión de cajas ponderadas (en inglés, *Weighted Boxes Fusion*) también conocido WBF (Solovyev et al., 2021)

Funciona similar a NMW pero cambiando algún aspecto, como por ejemplo que cada caja delimitadora se asocie con un peso que indica la confianza o la precisión del detector que la generó. Estos pesos se utilizan para ponderar las cajas delimitadoras y determinar su contribución a la detección final. Al combinar la caja con un peso, la confianza de la caja cambia dependiendo de los pesos de las detecciones. Está técnica utiliza información sobre el grupo de modelos que entran en esa superposición.

Este tipo de enfoque tiene la particularidad de que al no excluir casillas como en NMS y soft-NMS, se pueden corregir casos en donde todos los modelos predicen de manera incorrecta y no solo deja la casilla con un cuadro inexacto, como se muestra en la Figura 1.6.

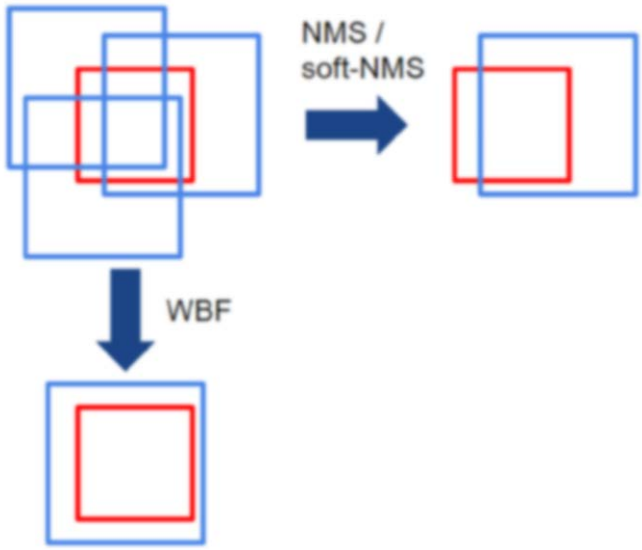


Figura 1.6 Esquemático de NMS/sof-NMD vs WBF en donde todos los modelos predicen incorrectamente. La caja roja significa que es la verdadera y las cajas azules son las predichas por algún modelo (Solovyev et al., 2021)

1.2.5 Métricas

Para evaluar lo que es un verdadero positivo se utiliza la intersección sobre la unión, IOU (del inglés, *Intersection Over Union*) que es una medida que indica qué tanto se superponen dos cajas delimitadoras. En este caso, la caja predicha respecto a la caja verdadera. Si este valor supera el 0.5 se considera que la detección ha sido correcta y está dada por la ecuación 1.

$$TP = \frac{BB\ Ground\ truth\ area\ \cap\ BB\ Predicted\ area}{BB\ Ground\ truth\ area\ \cup\ BB\ Predicted\ area} \geq 0.5 \quad (1)$$

Para la evaluación en el estudio se calcularon varias métricas como:

Precisión: la cantidad de instancias detectadas correctamente en relación con la cantidad total de instancias detectadas está dada por la ecuación 2.

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

Recall: corresponde a la sensibilidad e indica la fracción de instancias relevantes que se recuperaron, la cual se expresa en la ecuación 3:

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

F1- score: representa la media armónica entre la precisión y la recuperación, lo que significa qué tan robusto es el modelo. Para calcular este valor, se utiliza la ecuación 4**:**

$$F1 - Score = 2\,x\,\frac{Precision\,x\,Recall}{Precision + Recall} \tag{4}$$

La precisión promedio media o mAP (del inglés Mean Averge Precision) consiste en encontrar el área debajo de la curva considerando la recuperación y precisión. Gráficamente se puede observar cómo se presenta en la Figura 1.7 donde el área bajo la línea punteada de color rojo representa el número de clases y/o los umbrales generales de IOU. La métrica se usa para evaluar que tan bien el detector localiza y clasifica los objetos correctamente; está dada por la ecuación 5.

$$mAP = \frac{1}{C}\sum_{k=t}^{N} p(k)\Delta\ R(k) \tag{5}$$

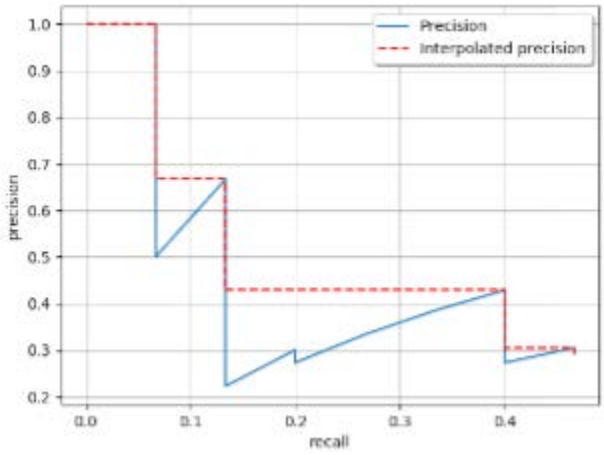


Figura 1.7 Representación gráfica del mAP (Padilla et al., 2020)

1.3 Organización del documento de tesis

En el Capítulo 1, se expone la propuesta de solución para resolver la problemática para el reconocimiento de insectos, desde introducción hasta alcances y limitaciones, además se explica los conceptos claves en la sección del marco teórico.

El Capítulo 2, se muestra una recopilación de artículos del estado del arte relacionados con el tema de tesis, los cuales se encuentran divididos por núcleos temáticos.

En el Capítulo 3, contiene el análisis y el diseño del sistema, donde se explica la arquitectura empleada, el conjunto de datos utilizado y sus características, así como las configuraciones empleadas.

El Capítulo 4, se detalla los casos de experimentación, organizados de tal manera que el desarrollo y evaluación de cada uno permitió la obtención de un mejor resultado del sistema.

En el Capítulo 5, se presentan las conclusiones de este trabajo de tesis, así como el trabajo futuro sugerido para continuar con la investigación.

CAPÍTULO 2

Estado del arte

En la siguiente sección se presenta el análisis de trabajos relacionados con la detección y clasificación de insectos catalogados como plagas, los cuales permitieron el desarrollo de la propuesta de solución.

2.1 Antecedentes en el CENIDET

En la investigación realizada en el repositorio de tesis del CENIDET, no se localizó tesis alguna que se relacione con la identificación de plagas de insectos. El único trabajo similar elaborado en el plantel es el trabajo “Sistema de visión artificial para la detección de plantas enfermas mediante aprendizaje profundo” (Andros Meraz Hernández, 2022) que tiene como objetivo: “Desarrollar e implementar un sistema de visión artificial, utilizando técnicas de aprendizaje profundo, que identifique si una planta presenta síntomas de enfermedad.”. En este caso las plantas consideradas son el jitomate y maíz.

2.2 Estado del arte

2.2.1 Conjuntos de datos más empleados en la literatura

Automatic classification for field crop insects via multiple-task sparse representation and multiple-kernel learning (xie1, xie2015) (Xie et al., 2015)

En este trabajo se busca y aporta la clasificación de 24 especies de plagas comunes en los cultivos de maíz, soya, trigo y canola. Cada una de las clases tiene 60 imágenes reuniendo alrededor de 1,440 que fueron tomadas a partir de una cámara digital en condiciones de iluminación uniforme con una resolución de 1,280x960 píxeles. Las imágenes fueron normalizadas y escaladas a 300 x 300. Un ejemplo de este conjunto de imágenes se encuentra en la Figura 2.1.

La solución propuesta para clasificar los insectos es fusionando características basadas en una codificación dispersa, en lugar de características manuales en función de la apariencia como el color, la textura, la forma o el histograma de gradientes orientados HOG (del inglés, *histogram of oriented gradients*). Esta codificación dispersa consiste en incorporar en una representación lineal de elementos básicos en un diccionario, y combinarlos a partir de un método MKL (del inglés, *Multiple Kernel Learning*) que utiliza ejemplos negativos y positivos de imágenes de insectos. Al optimizar los pesos del *kernel* pueden clasificarlos considerando estos histogramas. Se evaluó esta estrategia con la exactitud obteniendo como un máximo de 90.7.

Figura 2.1 Submuestra del conjunto de datos XIE2015 (Xie et al., 2015).

The ICAR-National Bureau of Agricultural Insect Resources (NBAIR) (Dr. J. H. Kulkarni et al., 2018)

NBAIR es un conjunto de datos libres, donde la recopilación de cada clase de insecto se obtiene de manera individual. Contiene 1,329 especies distribuidas en 2,953 imágenes reportando en (Dr. J. H. Kulkarni et al., 2018). En la actualidad cuenta con 78 cultivos, la distribución de estas muestras adquiridas hace que sean escasas por clase. Otra de las características de este conjunto de datos es que presenta información acerca de la plaga como: el nombre científico, la posición en taxonomía, nombre común que por lo general se le otorga y plantas a las que ataca, lo que lo vuelve abundante el conjunto en ese aspecto. Además, la actualización del conjunto de imágenes va en aumento año tras año. Un ejemplo de las muestras se observa en la Figura 2.2.

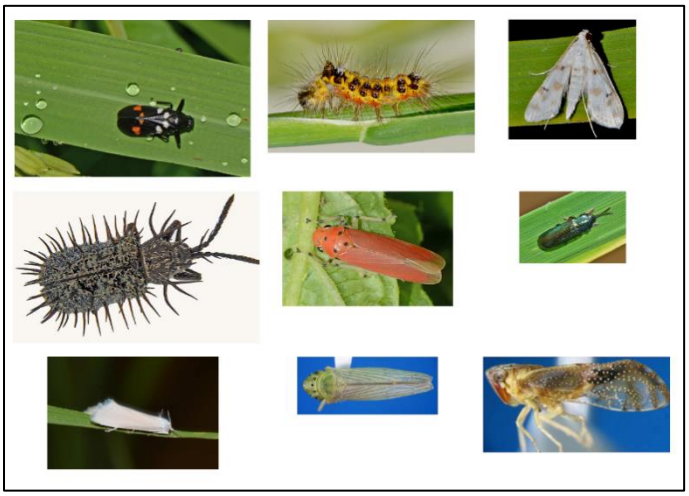


Figura 2.2 Submuestra del conjunto de datos de NBAIR *(*The ICAR-National Bureau of Agricultural Insect Resources (NBAIR)*, 2013)*

Multi-level learning features for automatic classification of field crop pests (Xie et al., 2018)

El trabajo presenta la creación de un conjunto de datos llamado D0 (también llamada xie2018) para los cultivos de maíz, soja, trigo y canola donde se reunieron 4,500 imágenes para 40 tipos de especies de plagas comunes del campo. Estas fueron tomadas a partir de fotografías para posteriormente ser etiquetados por expertos y fueron preprocesadas para mantener una uniformidad en la iluminación con una resolución de 200x200 píxeles. Una submuestra de este conjunto se muestra en la

Figura **2.3**. Para poder clasificar correctamente las especies en el trabajo original, se utilizaron características de aprendizaje en varios niveles. Este tipo de aprendizaje consiste en entrenar un diccionario a partir de parches de imágenes sin etiquetar, luego agrupan espacialmente estas características de bajo nivel por su dispersión, generando los parches que son la entrada para el clasificador que fue máquinas de vector soporte (del inglés, S*upport-Vector Machines*, SVM). Este tipo de estrategias obtiene mejores resultados que utilizar características manuales como algoritmos HOG o la transformación de característica invariante de escala (SIFT del inglés, S*cale-Invariant Feature Transform*) que utilizan el color o la textura para la extracción de características. Los resultados obtenidos se muestran en la

Tabla **2.1**.

Figura 2.3 Submuestra del conjunto de datos de XIE2018 también conocido como D0 o xie2 (Xie et al., 2018)

Tabla 2.1 Comparación en exactitud ante la combinación de diferentes técnicas de extracción de características usadas en (Xie et al., 2018)

Feature combination	Fusion framework				
	MKL	MTJSRC	IOIAICM	ACFCI	Proposed
Color+Texture+SIFT	70.5 ± 2.4	80.5 ± 1.8	74.8 ± 1.5	**83.5 ± 1.4**	82.6 ± 1.8
Color+Texture+ HOG	67.1 ± 1.6	78.5 ± 0.8	70.1 ± 2.0	80.3 ± 2.6	80.0 ± 2.0
Color+ +SIFT+ HOG	63.5 ± 1.2	68.5 ± 1.9	65.3 ± 3.2	78.5 ± 2.5	77.2 ± 1.9
Texture+SIFT+ HOG	59.5 ± 1.0	61.5 ± 2.6	63.6 ± 2.1	73.2 ± 3.0	71.5 ± 2.8
Our learned features	-	-	-	**81.6 ± 2.1**	**89.3 ± 2.8**

Research on insect pest image detection and recognition based on bio-inspired methods (Deng et al., 2018)

Se propone un conjunto de datos para 10 especies de plagas que afectan principalmente a las plantas del té, cada una de estas clases cuenta entre 40 y 70 imágenes obtenidas a partir de recursos de internet y tomadas usando una cámara digital. Un subconjunto de estas imágenes es mostrado en la Figura 2.4 donde se muestra una variación tanto de escala de la plaga, como en la perspectiva y condiciones de iluminación y fondos.

La metodología que se siguió fue utilizar un modelo SUN (*Saliency Using Natural statistics*) el cual detecta al insecto y obtiene el área de interés (*ROIs, Region of Interest*) para extraer las características, mediante dos algoritmos de extracción manual, que fueron LCP (*Local Configuration Pattern*) y SIFT-HMfi. Finalmente, las características se normalizaron a una escala de 0 a 1 y se usa un clasificador SVM dando una exactitud de 85.5%. Esta metodología resulta como segunda mejor sólo superada por una red neuronal llamada *MatConvNet* con 86.9% de exactitud.

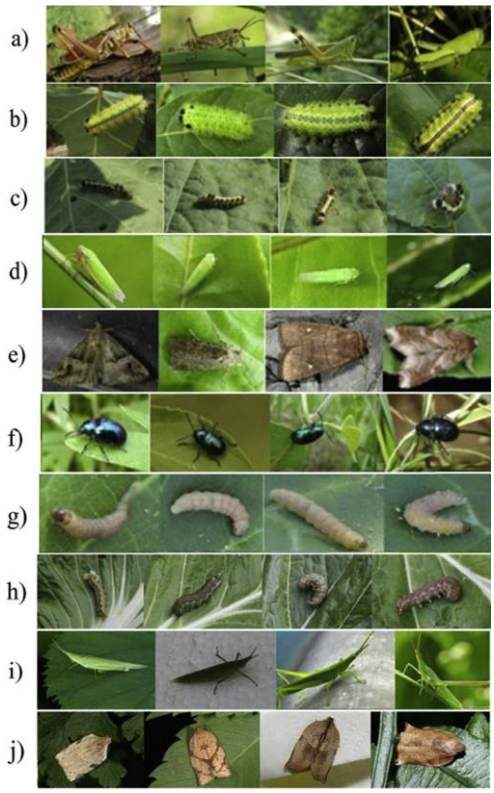


Figura 2.4 Muestra de las 10 clases en ambientes naturales donde cada fila corresponde a una clase, mostrada de la siguiente manera: *a) Locusta migratoria, b) Parasa lepida, c) Gypsy moth l*arva*, d) Empoasca flavescens, e) Spodoptera exigua, f) Chrysochus chinensis, g) Laspeyresia pomonella larva, h) Spodoptera* (Deng et al., 2018).

***IP102*: A Large-Scale Benchmark Dataset for Insect Pest Recognition** (Wu et al., 2019)

El artículo presenta la creación de un conjunto de datos para la identificación de los insectos llamado *IP102* que cuenta con 75,000 imágenes, categorizados en 2 superclases, 8 subclases y 102 clases. El proceso de creación se inició usando motores de búsqueda comunes de internet como lo son *Google, Flickr* y *Bingetc* obteniendo un conjunto de 300,000. Al remover imágenes repetidas y teniendo en cuenta que sean relevantes a las plagas se redujo la muestra a 120,000; posteriormente, las imágenes fueron categorizadas por expertos en plagas. El conjunto de plagas presenta una distribución de datos desequilibrada como puede ocurrir en la realidad, cuyas imágenes pueden incluir insectos en sus cuatro etapas que son huevo, larva, pupa y adulto. En comparación con el resto de los conjuntos de datos disponibles, esta base presenta significativamente más imágenes y más clases, que los conjuntos existentes. Además de que fue asistida en su construcción con la ayuda de expertos en plagas y así tener categorías correctas.

Una característica de este conjunto es que se adquirieron imágenes en escenarios reales como se muestra en la Figura 2.5.

Figura 2.5 Varias muestras del conjunto IP102 obtenida del trabajo de (Wu et al., 2019)

Algunos de los resultados reportados en el estado del arte utilizando el IP102 se muestran en la Tabla 2.2, en ella se puede observar que los algoritmos clásicos de extracción tienen peores resultados respecto a las técnicas de aprendizaje profundo. La Tabla 2.3 muestra los resultados de comparar varias técnicas de aprendizaje profundo para detectar insectos. Como se puede ver, la implementación del detector es significativamente menor en otras arquitecturas de aprendizaje profundo debido a que las clases no se encuentran balanceadas; además, de que algunas plagas presentan características distintas en el fondo.

Tabla 2.2 Resultados del conjunto de datos IP102 con aprendizaje profundo y extracción de características manuales (Wu et al., 2019)

#	Methods	SVM						KNN					
		Pre	Rec	F1	GM	M_{AUC}	Acc	Pre	Rec	F1	GM	M_{AUC}	Acc
Handcrafted Feature	CH	9.7	3.2	2.5	0.3	12.0	12.9	18.2	14.2	15.0	8.3	16.8	15.8
	Gabor	8.5	3.9	3.6	0.5	12.1	14.2	22.0	14.9	16.5	9.1	20.0	19.2
	GIST	12.2	3.8	3.8	0.6	12.1	13.1	19.1	15.1	15.4	9.2	19.2	18.2
	SIFT	25.1	6.3	6.8	1.0	19.9	18.1	19.4	10.3	12.1	5.6	15.9	13.1
	SURF	28.2	7.3	8.3	1.5	21.2	19.5	21.3	11.5	13.4	7.1	17.5	14.7
	LCH	7.2	5.0	4.7	0.9	11.1	13.1	21.6	14.7	16.1	8.3	19.0	16.8
Deep Feature	Alexnet	41.5	16.4	21.0	9.3	32.5	28.3	36.7	32.4	33.5	23.9	41.0	40.7
	GoogleNet	45.8	25.8	30.4	16.0	41.9	40.5	36.8	31.7	33.0	23.3	41.6	40.7
	VGGNet	43.4	37.6	39.1	28.3	48.1	48.7	41.9	37.8	39.0	29.8	47.6	47.1
	ResNet	43.6	39.1	40.6	31.0	48.7	49.5	43.7	39.1	40.5	30.7	48.2	49.4

Tabla 2.3 Implementación del detector de objetos en el conjunto de datos IP102 (Wu et al., 2019)

Method	Backbone	AP	AP^{50}	AP^{75}
FRCNN [34]	VGG-16	21.05	47.87	15.23
FPN [20]	ResNet-50	28.10	54.93	23.30
SSD300 [22]	VGG-16	21.49	47.21	16.57
RefineDet [45]	VGG-16	22.84	49.01	16.82
YOLOv3 [33]	DarkNet-53	25.67	50.64	21.79

A Deep-Learning Approach for Automatic Counting of Soybean Insect Pests (Tetila, MacHado, et al., 2020)

El trabajo busca la identificación y conteo de siete clases de plagas de insectos en el cultivo de la soja por medio de un sistema de visión llamado *PYNOVISÃO*, para la detección automática de la plaga por medio de imágenes. Durante la etapa de recolección se creó el conjunto de datos *INSECT10K7C* reuniendo alrededor de 1,000 imágenes en condiciones de iluminación y clima diferentes, con tamaño de 4,032x2,268 píxeles en un ángulo de 45° grados hacia el suelo, a un metro de distancia.

El enfoque del trabajo se basa en la implementación del algoritmo Superpíxeles SLIC (*Simple Linear Iterative Clustering*) para segmentar los insectos en la imagen, el método SLIC consiste en utilizar k-means para la generación de regiones similares a las cuales se les denomina superpíxeles y agruparlos de acuerdo su color. Este enfoque reduce la complejidad

computacional al limitar el espacio de búsqueda a una región proporcional. Una vez segmentadas las imágenes fueron etiquetadas por un experto, separando aquellos insectos que no provocan daños económicos de los que sí lo causan. Al utilizar el algoritmo SLIC se generaron 10,000 superpíxeles distribuidos como se muestra en la Figura 2.6, estos superpíxeles fueron aumentados mediante transformaciones como rotación, escalado, desplazamiento y zoom consiguiendo 450,000 muestras nuevas.

Para el entrenamiento se eligieron tres modelos de aprendizaje profundo reconocidos y de código abierto tales como: Inception-Resnet-v2, ResNet-50 y DenseNet-201. Bajo tres estrategias diferentes que son: 1) 100% FT (siglas en inglés, *fine-tuning*) con los pesos obtenidos de ImageNet, 2) No TL (siglas en inglés de *tranfer learning*) redes completas con los pesos inicializados aleatoriamente y 3) TL con pesos obtenidos de ImageNet, mostrando que el ajuste fino es la mejor opción de las tres. Los resultados también mostraron que el conteo de la plaga se ve perjudicada cuando el insecto se presenta en dos superpíxeles diferentes, por lo cual es un tema abierto y todavía no resuelto. Los resultados de los entrenamientos se muestran en la Tabla 2.4.

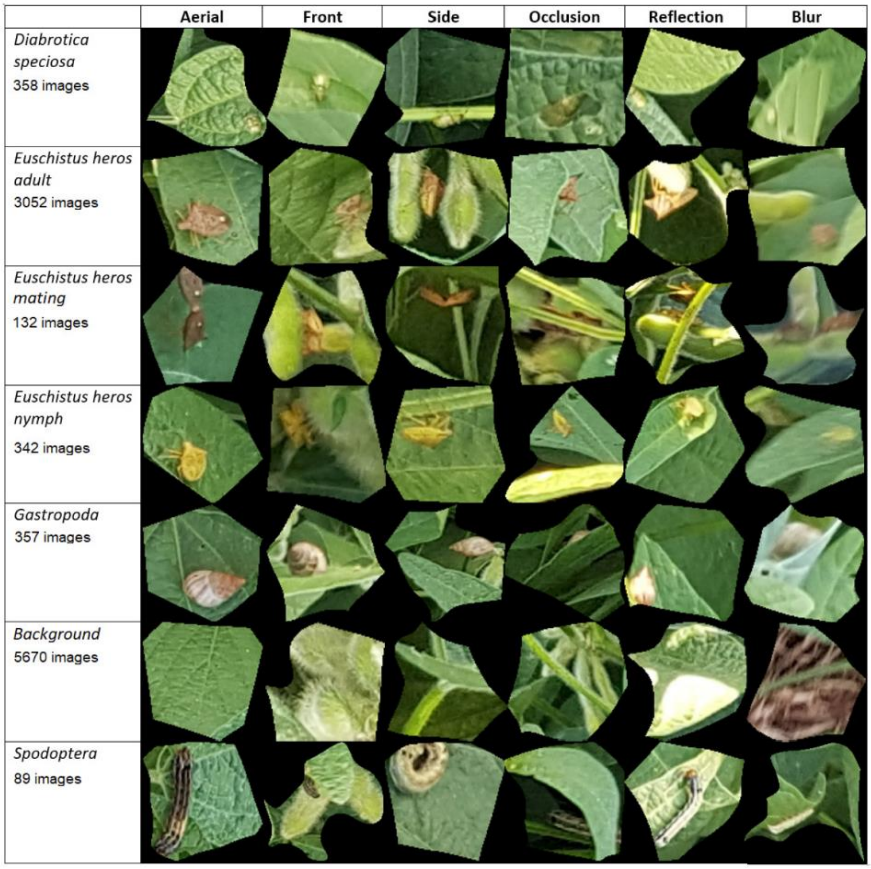


Figura 2.6 Submuestra de conjunto de datos INSECT10K7C (Tetila, MacHado, et al., 2020)

Tabla 2.4 Resultados de entramientos (Tetila, MacHado, et al., 2020)

Model	Training strategy	Training time (h)	Accuracy (%)
DenseNet-201	FT	24.47	**94.89**
DenseNet-201	TL	19.40	61.16
DenseNet-201	No TL	23.59	85.29
Inception-Resnet-v2	FT	31.48	**93.40**
Inception-Resnet-v2	TL	18.42	66.79
Inception-Resnet-v2	No TL	28.01	85.86
Resnet-50	FT	22.38	**93.78**
Resnet-50	TL	19.31	59.27
Resnet-50	No TL	22.16	84.32

Common pests image recognition based on deep convolutional neural network (J. Wang et al., 2020)

El artículo tiene como objetivo mostrar la propuesta de una red neuronal híbrida llamada CPAFNet que utiliza el conjunto de imágenes propio llamado CPAF-Dataset. Dicho dataset consta de plagas agrícolas y forestales comunes construida por 19 insectos comunes y uno de larvas, de los cuales 10 dañan a los árboles frutales, cuatro especies a los cereales y cinco son insectos benéficos, separados en dos categorías (adultos y larvas). La recopilación de estas imágenes está formada de dos maneras:

- Son 973 imágenes compuestas de: 643 insectos inofensivos para la silvicultura, 156 insectos inofensivos para la agricultura y 174 insectos benéficos.
- Por el otro lado, 3,936 imágenes se integran de 1,456 insectos inocuos para la silvicultura, 969 insectos inocuos para la agricultura y 1,511 insectos benéficos con enfoques de rastreo.

Para evitar el sobreajuste se aumentó el volumen de datos de 4,909 imágenes a 73,635 colocando cada imagen original 14 veces en distintas posiciones. Se diseñó el modelo CPAFNet utilizando la estructura mejorada de *VggA* e *Inception* construyendo la arquitectura híbrida con una capa de entrada, cuatro capas de convolución, seis capas de agrupación máxima, tres módulos de inicio, una capa de agrupación media, una capa de convolución y una capa de salida. Los resultados se muestran en la Tabla 2.5.

Tabla 2.5 Resultados con tiempo del entrenamiento (J. Wang et al., 2020)

Network model	Learning rate					
	0.002		0.0002		0.00002	
	Balanced accuracy	Training time	Balanced accuracy	Training time	Balanced accuracy	Training time
VggA	0.6612	5h23m	**0.8924**	5h18m	0.8718	5h19m
Vgg16	0.8255	7h32m	**0.8545**	7h54m	0.5746	7h52m
ResNet50	**0.9074**	7h36m	0.8446	5h12m	0.5439	7h18m
Inception V3	**0.9186**	7h51m	0.8425	3h13m	0.6175	3h6m
CPAFNet	**0.9226**	**1h18m**	0.9110	1h11m	0.7698	1h2m

Crop pest recognition in natural scenes using convolutional neural networks (Y. Li et al., 2020)

El trabajo tiene el propósito de clasificar 10 tipos de plagas comunes y de reproducción rápida. La obtención de las imágenes se hizo por medio de búsquedas en internet, además de fotografías tomadas al aire libre recopilando un total de 5,629 imágenes. Aplicando técnicas de aumento de datos el conjunto se incrementó a 14,475 imágenes.

Al conjunto de datos se le aplicó un preprocesamiento para la eliminación del fondo, dado que la plaga puede estar en dos situaciones diferentes. Es decir, la plaga y el fondo son contrastantes, o cuando la plaga y el fondo son similares. Se utilizan dos métodos para la segmentación: para el primer caso se usaron técnicas de umbral adaptativo, detección de contornos y transformación divisoria. Para el segundo caso se utilizó el algoritmo de *GrabCut.*

Para encontrar el mejor modelo se entrenaron cinco de las arquitecturas más populares de CNN, obteniendo los siguientes resultados en exactitud: *GoogLeNet* con 94.61% mientras que *ResNet152* con 93.02%, *VGG-19* con 93.05%, *VGG-16* con 91.94% y *ResNet50* con 91.89%. El mejor modelo fue *GoogLeNet* y se optimizaron sus parámetros logrando una exactitud más alta del 98% al cambiar la capa completamente conectada a 4,096 y utilizar el optimizador *Adagrad*.

Pest24: A large-scale very small object data set of agricultural pests for multi-target detection (Q. J. Wang et al., 2020)

En este artículo se presenta la creación de un nuevo conjunto de imágenes de insectos, los autores mencionan que los modelos para la clasificación de plagas se centran en el reconocimiento y clasificación de una imagen de una única plaga, pero no en la detección de plagas de múltiples objetivos. Para la detección de plagas de múltiples objetivos se deben reconocer, simultáneamente, cada instancia de la plaga y contar el número de insectos que pertenecen a cada categoría en la imagen, pero se carece de un conjunto de datos para múltiples plagas a gran escala por lo que en el trabajo el desarrollo Pest24 ayuda en esta área.

El conjunto de imágenes de plagas Pest24 fueron tomadas por un dispositivo de adquisición automático especializado fabricado por el Instituto de Máquina Inteligente de la Academia de Ciencias de China, como se observa en la Figura 2.7. Las fotografías están en alta resolución y dan un total de 25,378 imágenes de 2,095x1,944 píxeles, con tamaños de insectos que varían del 0.00125% al 2.002% en tamaño relativo a la imagen dependiendo la clase, así como una densa distribución y alta similitud entre los objetos de plagas tanto en forma como color (véase Figura 2.8). Las imágenes involucran 38 categorías de plagas en cultivos de campo pertenecientes a 5 órdenes de insectos los cuales son *Coleoptera, Homoptera, Hemiptera, Orthoptera* y *Lepidoptera* juntando 13 familias de insectos. Dado que 14 de las 38 categorías de plagas tienen pocas instancias presentes (entre 1 y 11) solo se consideran 24 categorías restantes como objetivos a detectar.

Figura 2.7 Dispositivo usado para la recolección de imágenes de insectos del dataset Pest24 (Q. J. Wang et al., 2020)

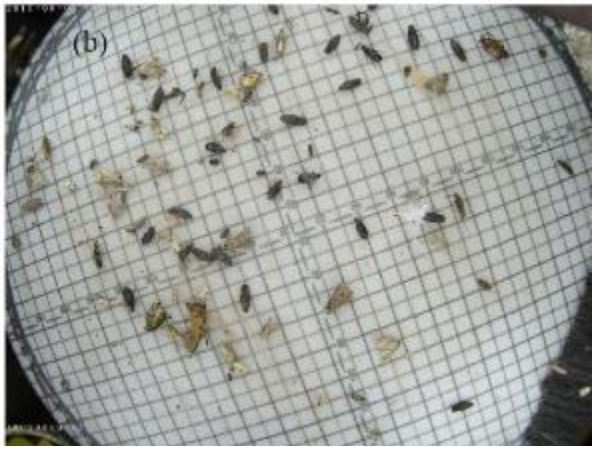


Figura 2.8 Ejemplo de la captura del dispositivo de recolección de imágenes para el conjunto de imágenes Pest24 (Q. J. Wang et al., 2020)

Entre las propiedades que presenta este conjunto se tiene que incluye imágenes de plagas que no pertenece a ningún grupo, un fondo complejo, superposición o adhesión de objetos, distintas escalas del mismo objeto, discrepancias entre el color del insecto y el fondo además de similitud entre especies del 45% al 70%.

Pest24 se utilizó con cuatro métodos de aprendizaje profundo para evaluarlo. Se midió el proceso de detección de insectos mediante las métricas AP (*average precision,* "precisión promedio") y mAP. YOLOv3 obtuvo los mejores resultados con un 63.54%, el resto de los modelos entrenados dieron un resultado mAP de la siguiente manera: SSD con 50.51%, Faster-RCNN con 51.75% y Cascade R-CNN con 59.97%. En los objetos individuales de AP para YOLOv3 existen 20 categorías con valores superiores al 50% mientras que para *faster RCNN* son 12 solamente. Con las categorías de saltamontes del arroz, *plutella xylostella* y tigre de ocho caracteres tienen valores menores al 30%. Durante el análisis de los datos los experimentos mostraron que el número de instancias y el tamaño de los objetos son los puntos críticos de la precisión de cada categoría

AgriPest: A Large-Scale Domain-Specific Benchmark Dataset for Practical Agricultural Pest Detection in the Wild (R. Wang et al., 2021)

El artículo presenta *AgriPest* que es un conjunto de datos que contiene pequeñas plagas silvestres con 49,700 imágenes capturadas por ellos y un total de 14 especies de plagas con hasta 264,700 de cuadros delimitadores de localización, para cuatro tipos de cultivos (trigo, arroz, maíz y colza). En AgriPest las imágenes están a distintas distancias, poses y perspectivas, y el *dataset* es validado por un grupo de expertos en el área, agregando datos de temperatura y humedad en las medidas tomadas. Las imágenes fueron tomadas en campos silvestres a

diferencia de otros conjuntos de datos existentes que fueron creados en laboratorios y no en su hábitat. Además, los conjuntos de datos existentes como PASCAL VOC y MS COCO presentan los insectos pequeños a un porcentaje más grande en comparación con la imagen como se ve en la Figura 2.9.

Figura 2.9 Muestra del dataset AgriPest (R. Wang et al., 2021)

Para el método de detección utilizaron arquitecturas de dos etapas ya que la detección de la plaga tiene mejores resultados que los de una etapa al ser los insectos demasiados pequeños. Con Cascade R-CNN como detector, el mAp@0.5 varía del 41.05% al 90.92% dependiendo de la especie.

2.2.2 Clasificación de insectos

Crop pest classification based on deep convolutional neural network and transfer learning (Thenmozhi & Srinivasulu Reddy, 2019)

El objetivo del trabajo fue desarrollar un sistema de clasificación de plagas en cultivos mediante una red neuronal convolucional y transferencia de aprendizaje. El modelo propuesto consiste en seis capas convolucionales, cinco capas de agrupación máxima, una capa completamente conectada y la capa de salida con clasificador softmax, como se muestra en la Figura 2.10. El modelo recibe una imagen de 227x227 píxeles de tres conjuntos de datos, NBAIR y Xie2 ambos con 40 clases de insectos y Xie1 con 24 clases de insectos. El modelo propuesto se realizó por medio de matlab2018a, con los parámetros especificados en la Tabla 2.6.

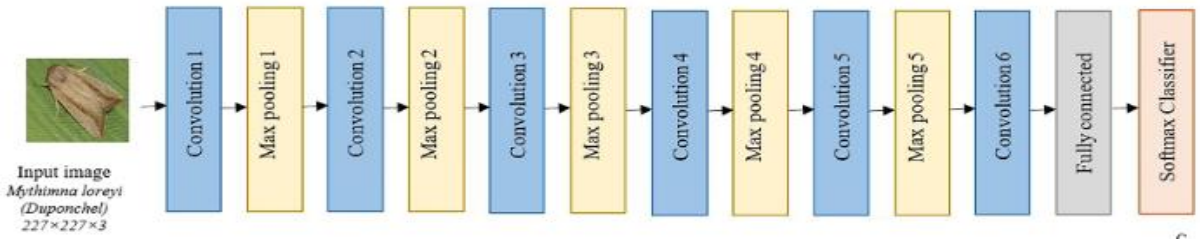


Figura 2.10 Composición de arquitectura *(Thenmozhi & Srinivasulu Reddy, 2019)*

Tabla 2.6 Parámetros empleados en trabajo *(Thenmozhi & Srinivasulu Reddy, 2019)*

Layers	Output size	Kernel size	Stride	Pad
Input image	(3,227,227)			
Convolution 1	(8,223,223)	5	1	0
Max pooling 1	(8,111,111)	2	2	0
Convolution 2	(16,107,107)	5	1	0
Max pooling 2	(16,52,52)	2	2	0
Convolution 3	(32,50,50)	3	1	0
Max pooling 3	(32,25,25)	2	2	0
Convolution 4	(64,23,23)	3	1	0
Max pooling 4	(64,11,11)	2	2	0
Convolution 5	(128,9,9)	3	1	0
Max pooling 5	(128,4,4)	2	2	0
Convolution 6	(256,2,2)	3	1	0
Fully connected	(*C)			

En la creación del modelo se utilizaron 10 épocas con un mini lote de 64 y una taza de 0.0001; variaciones de estas métricas afectaban la exactitud del modelo. Utilizaron estas métricas para evaluar, en términos de exactitud y eficiencia, cuatro modelos pre entrenados por medio de aprendizaje por transferencia que fueron: AlexNet, ResNet, GoogLeNet y VGGNet dando para cada conjunto los rendimientos mostrados en Tabla 2.7

El modelo propuesto obtuvo mejores rendimientos para los tres conjuntos de datos considerados, siendo útil para que los agricultores identifiquen y clasifiquen los insectos.

Tabla 2.7 Rendimientos obtenidos en el entrenamiento de modelos *(Thenmozhi & Srinivasulu Reddy, 2019)*

	NBAIR	**Xie1**	**Xei2**
Modelo propuesto	96.75	97.47	95.97
AlexNet		94.23	92.25
ResNet-50		95.87	92.96

ResNet-101	95.02	95.95	93.96
VGG-16		96.25	93.67
VGG-19		95.89	93.03

Identification of Crop Consuming Insect Pest from Visual Imagery Using Transfer Learning and Data Augmentation on Deep Neural Network (Reza et al., 2019)

En este trabajo se utilizó el conjunto de datos más grande disponible públicamente denominado como *IP102*. Debido a que *IP102* cuenta con un desequilibrio de muestras de cada clase es propenso al sobreajuste mayormente en aquellos insectos con menor número de imágenes, por lo que se optó por aumento de datos por medio de rotación, desplazamiento, volteo y reflexión, etc. Además de normalizar los valores de los píxeles entre 0 y 1, con una resolución de 224x224.

Se evaluaron diferentes arquitecturas para la transferencia de aprendizaje como VGG19, Inceptionv3 y ResNet50. La mayoría de los modelos de clasificación consta de dos partes, la primera es el conjunto de capas convolucionales que funcionan como extractor de características para generar el mapa de características y la segunda parte funciona como clasificador a partir de los mapas de características. Para este trabajo modifican la segunda parte utilizando dos capas completamente conectadas (FC), la primera de 1,024 nodos y la capa FC de salida final en 102 nodos, obteniendo resultados de exactitud del 55,7% para VGG19, 56,35% para ResNet50 y 56,73% para InceptionV3. Como recomendación el autor sugiere que debido a que los insectos se presentan en diferentes orientaciones y también con distinta perspectiva, es recomendable aplicar más técnicas de aumento de datos o juntar conjuntos dado que, aunque el conjunto sea grande aún se presentan problemas con el sobreajuste.

Identification of the agricultural pests based on deep learning models(Song et al., 2019)

En este trabajo se utilizó el modelo de Inception-v3 y el modelo de *Inception-v4* en *GoogLeNet* para el aprendizaje por transferencia. Se utilizó un conjunto de 35,000 imágenes para la identificación de 71 tipos diferentes de insectos de plaga. Del modelo original de las redes *Inception-v3* y *Inception-v4* se modificó descomponiendo la convolución de 7x7 en dos convoluciones seriales unidimensionales y la convolución de 3x3 se descompone en dos

convoluciones seriales unidimensionales (1x3 y 3x1) esto para acelerar el cálculo de la red, así como aumentar la no linealidad. Además de agregar la estructura de bloques residuales en ResNet al modelo Inception-v4. Para el artículo se utilizó como medida de exactitud en la segmentación la relación de intersección y la unión del gráfico de la máscara de segmentación y para el rendimiento del entrenamiento el valor de pérdida dada por la entropía cruzada.

El modelo propuesto obtuvo una exactitud del 100% con algunos insectos como la larva y el *limax* y su exactitud más baja fue del 75% con insectos como la polilla parásita. La diferencia se debe a que en algunos insectos tienen tonalidades semejantes al fondo, lo que provoca su confusión, otro factor que influye y disminuye la exactitud es que una misma plaga puede presentar diferencias dependiendo la estación del año. Inception-v4 obtuvo en promedio 97.3% de exactitud logrando mejores resultados respecto a otras arquitecturas como *LeNet* con un 90% de exactitud, CNN con un 93.75%, SVM con un 89.26%, BP neuronal *network* con un 92.67%, SSGAN con 94.32% e Inception-v3 con 95.62%.

Recognition pest by image-based transfer learning (Dawei et al., 2019)

La investigación se centra en la identificación de 10 tipos de plagas de té, usando la arquitectura AlexNet, mediante aprendizaje por transferencia, creando un conjunto de datos a partir de búsquedas en internet, obteniendo 484 imágenes de insectos de plaga con variación en la escala, posición, vista, condiciones de iluminación y fondo.

El reentrenamiento del AlexNet fue evaluado a partir de la matriz de confusión donde la exactitud promedio fue del 93.84% que, comparado contra modelos de redes neuronales tradicionales como SIFT-HMAX y CNN, mostró mejores resultados que del 85.5% y 90.41% de los modelos mencionados respectivamente. También se comparó la métrica de exactitud obtenida por AlexNet respecto humanos expertos y se observa que el aprendizaje por transferencia mostró mejores resultados que 4 de los 6 expertos en el área, como se muestra en la Figura 2.11.

Es importante mencionar que, hubo especies en donde ningún experto identificó correctamente a las larvas *laspeyresia pomonella* y *Spodoptera*, mientras que el modelo de AlexNet si lo llevó a cabo. Mostrando que el aprendizaje por transferencia puede evaluar mejor que la mayoría de los expertos en el experimento. Los autores mencionan que esto puede deberse a que en el aprendizaje por transferencia no existe inferencia de factores externos, a diferencia del humano que puede ser afectado por fenómenos psicológicos y cognitivos que derivan en prejuicios, lo que le lleva a errores en la detección.

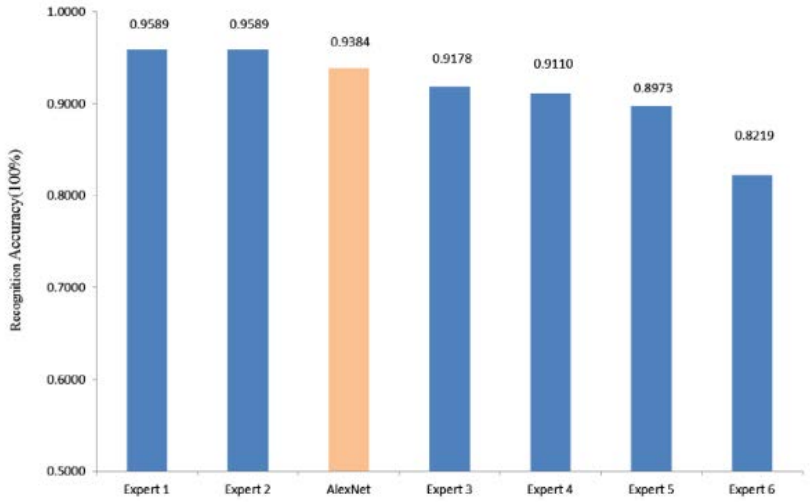


Figura 2.11 Exactitud de Alexnet contra expertos humanos en el área de reconocimiento de plagas (Dawei et al., 2019)

Crop pest classification with a genetic algorithm-based weighted ensemble of deep convolutional neural networks (Ayan et al., 2020)

Se presenta un modelo por votación ponderada llamado GAEnsemble para la clasificación de 40 tipos de insectos, tomando la decisión de la votación de acuerdo a la máxima probabilidad de la predicción de cada modelo seleccionado (SMPEnsemble). Para evitar resultados repetidos en la predicción, se optó por un algoritmo genético basado en la teoría de la evolución y con este encontrar pesos optimizados en la votación, se propone combinar arquitecturas distintas dado que cada red extrae características diferentes.

Para el entrenamiento de los modelos se utilizan tres tipos de conjuntos de datos diferentes en el trabajo: D0 (Xie et al., 2018), *SMALL* (Deng et al., 2018) e *IP102* (Wu et al., 2019)

Se analizaron siete arquitecturas diferentes de modelos CNN más populares para la clasificación del conjunto de datos D0 modificando cada arquitectura en sus hiperparámetros. Posteriormente, los tres mejores modelos de este entrenamiento (Inception-V3, Xception y MobileNet), se volvieron a entrenar con los conjuntos SMALL e *IP102*. Además, de realizar un aumento de datos como rotación, zoom y duplicación. Al combinar los resultados de los modelos con un método de conjunto ponderado GAEsemble se obtienen los resultados de la Tabla 2.8, en el conjunto D0. La Tabla 2.9 presenta los resultados de cada conjunto por separado.

Tabla 2.8 Resultados de experimentación de los diferentes algoritmos (Ayan et al., 2020)

Model	Accuracy	Avg Precision	Avg Recall	Avg f1 score
VGG-16	93.05	93.45	93.05	92.99
VGG-19	85.55	86.35	85.56	85.36
ResNet-50	92.18	92.74	92.18	92.07
Inception-V3	97.06	97.27	97.07	97.05
Xception	97.93	98.07	97.94	97.92
MobileNet	97.39	95.57	97.39	97.39
SqueezeNet	94.02	95.07	94.03	94.18
SMPEnsemble	98.37	98.50	98.37	98.38
GAEnsemble	**98.81**	**98.88**	**98.81**	**98.81**

Tabla 2.9 Resultados en cada uno de los conjuntos de datos empleados (Ayan et al., 2020)

Conjunto de datos	Modelo	Exactitud	Promedio en precisión	Promedio en Recall	Promedio en F1 score
D0	SMPEnsamble	98.37	98.50	98.37	98.38
	GAEnsamble	98.81	98.88	98.81	98.81
SMALL	SMPEnsamble	92.74	94.13	92.74	92.65
	GAEnsamble	95.16	95.41	95.16	95.07
IP102	SMPEnsamble	66.21	65.65	66.21	64.44
	GAEnsamble	67.13	67.17	67.13	65.76

Detection and classification of soybean pests using deep learning with UAV images

(Tetila, Machado, et al., 2020)

La investigación tiene por objetivo la clasificación de 13 tipos de insectos diferentes de plaga en la planta de soja bajo condiciones reales de campo, esto quiere decir que se buscó diferentes condiciones de iluminación, tamaño del objeto y variaciones en el fondo.

Para la adquisición de imágenes se utilizó un vehículo aéreo no tripulado UAV (del inglés, *unmanned aerial vehicle*) volando a dos metros del suelo en un campo de dos hectáreas en Brasil y recolectó 300 imágenes, y un teléfono celular que adquirió 5,000 imágenes. Una vez obtenidas fueron segmentadas con la técnica de súper píxeles SLIC (técnica que permite agrupar regiones de píxeles en el espacio 5D definido por L,a,b y las coordenadas *x*, *y*) y ser etiquetadas por un biólogo entomólogo. Los resultados de las imágenes segmentadas por la técnica SLIC se muestran en la Figura 2.12 y, cómo fueron guardadas después de etiquetarlas se muestra en la Figura 2.13. Se optó también por realizar un aumento de datos para hacer el sistema invariante a la rotación y escala.

Figura 2.12 Segmentación por medio de la técnica SLIC (Tetila, Machado, et al., 2020)

Figura 2.13 Submuestra de la técnica de súper píxeles en el conjunto de datos (Tetila, Machado, et al., 2020)

El entrenamiento en el aprendizaje profundo fue utilizando técnicas de aprendizaje con y sin transferencia y ajuste fino mostrando mejores resultados en este último. Además, se compararon los resultados con métodos de aprendizaje automático y descriptores locales, estos resultados son mostrados en la Tabla 2.10.

Tabla 2.10 Resultados con métodos de aprendizaje automático y descriptores locales (Tetila, Machado, et al., 2020)

Approach	Training strategy	Training time (s)	Accuracy (%)
Inception-v3	FTuning25%	5,077.79	91.87
Resnet-50	FTuning50%	4,968.79	**93.82**
VGG-16	FTuning100%	4,884.36	91.80
VGG-19	FTuning50%	4,909.60	91.33
Xception	FTuning75%	5,330.41	90.52
SVM	Combined feature	43.93	**60.46**
Random Forest	extraction based on	9.83	56.42
J48	color[a], gredient[b],	5.57	48.28
Naive Bayes	texture[c] and shape [d]	0.25	12.80
k-NN		0.01	42.04
AdaBoost		0.47	47.18
SIFT	SVM e k = 25	7,730.66	48.80
SIFT	SVM e k = 50	12,101.26	51.40
SIFT	SVM e k = 100	18,710.90	**52.13**
SURF	SVM e k = 25	7,391.27	48.73
SURF	SVM e k = 50	13,238.70	49.53
SURF	SVM e k = 100	23,487.67	50.73
OPFSEMImst	($Z_1' = 10\%$ e $Z_1'' = 90\%$)	1.78	51.28
	($Z_1' = 20\%$ e $Z_1'' = 80\%$)	1.79	52.29
	($Z_1' = 30\%$ e $Z_1'' = 70\%$)	1.80	52.34
	($Z_1' = 40\%$ e $Z_1'' = 60\%$)	1.80	52.72
	($Z_1' = 50\%$ e $Z_1'' = 50\%$)	1.81	52.63
	($Z_1' = 60\%$ e $Z_1'' = 40\%$)	2.32	52.95
	($Z_1' = 70\%$ e $Z_1'' = 30\%$)	2.33	**53.61**
	($Z_1' = 80\%$ e $Z_1'' = 20\%$)	2.39	53.19
	($Z_1' = 90\%$ e $Z_1'' = 10\%$)	2.41	53.52

DFF-ResNet: An Insect Pest Recognition Model Based on Residual Networks (W. Liu et al., 2020)

Se plantea una nueva red neuronal basada en *ResNet* con arquitectura de estructura residual cuyo bloque residual se fusiona con características llamado DFF-ResNet, esta a su vez combina dos redes residuales comunes: Pre-*ResNet* y *Wide Residual Network* (WRN). También se implementó la red DFF-Pre-*ResNet* aunque la segunda tuvo un poco menos de exactitud y sirvió para comprobar que agregar los bloques residuales de grupos anteriores promueven la capacidad de generalización del modelo al probar la red con distintos números de capas en este caso, 218 y 182 capas.

Durante las pruebas y análisis de los entrenamientos con los conjuntos CIFAR (Krizhevsky et al., 2009) y SVHN (Netzer et al., 2011) al cambiar varias profundidades y el ancho de la red residual,

se encontró mejorar en el rendimiento. Al probar el modelo con cada base de datos individualmente se iban modificando los hiperparámetros. Una vez obtenidos los mejores resultados, se probó DFF-ResNet para comprobar la adaptabilidad del modelo en un conjunto de datos con mejor resolución por lo que se pasó a evaluar en el conjunto de datos de *IP102*, mostrando que da mejores resultados que arquitecturas del estado del arte como se ve en la Tabla 2.11.

Tabla 2.11 Resultados en comparación de puntuaje F1 de DFF-Pre ResNet con métodos del estado del arte, usando el conjunto de datos IP102 (W. Liu et al., 2020)

Model	Number of parameters ($\times 10^6$)	F1 score (%)	Test accuracy (%)
AlexNet[36]	57.42	48.22	49.41
50-layer ResNet[6]	23.72	52.93	54.19
101-layer ResNet[6]	42.63	52.00	53.07
Googlenet[26]	10.24	51.24	52.17
16-layer VGG[37]	134.68	51.20	51.84
121-layer DenseNet[7]	7.06	52.97	54.59
62-layer DFF-Pre-ResNet	22.54	53.98	55.39
82-layer DFF-Pre-ResNet	30.20	54.18	55.43

Efficient Convolutional Neural Network for Pest Recognition-ExquisiteNet (Zhou & Su, 2020)

Los autores propusieron una arquitectura llamada *ExquisiteNet* como se muestra en la Figura 2.14 para el reconocimiento de plagas. El modelo consta de dos módulos, uno tiene una doble función al ser un bloque de cuello de botella y de excitación (bloque DFSEB) y el otro es el bloque de función máxima (bloque ME).

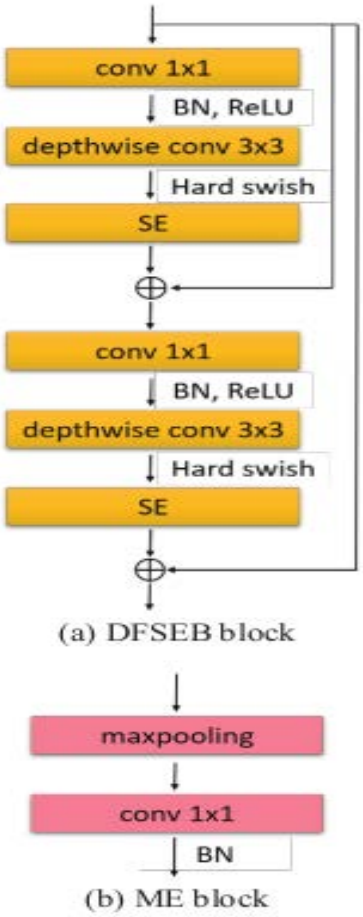


Figura 2.14 Arquitectura *ExquisiteNet* *(Zhou & Su, 2020)*

La estructura de su modelo es:

- Fusión doble: extrae características para aumentar la precisión de la clasificación, se fusiona dos veces cada bloque DFSEB y así concatenar la capa superficial y las características de la capa profunda.
- No expansión: Si el número de mapas de características de entrada es igual a la cantidad de mapas de características de salida de la capa convolucional, la velocidad de cálculo se maximiza.
- Cuello de botella: Se producen pesos cuyo número es el mismo que el de los mapas de características de salida de la capa anterior- Estos pesos multiplican la salida de la capa anterior para cambiar la importancia de cada mapa de características.
- Bloque ME: Está compuesto por una capa de agrupación máxima, una convolucional y una de normalización por lotes para aumentar la velocidad de cálculo, más capas de características suelen generar una mayor precisión.

Como conclusión, se comparó su propuesta con distintos modelos tales como SqueezeNet, ShuffleNetV2, DenseNet121, MoblieNetV3-large, ResNet18, ResNet50, ResNet101, EfficientNet-b0 (ver Tabla 2.12). Los parámetros utilizados en SqueezeNet eran menores que el resto de las arquitecturas, todos los modelos fueron entrenados bajo el conjunto de datos *IP102* y se obtuvo una exactitud del 52.32% siendo la segunda más alta y con menos parámetros, pero se recomienda el uso de un detector de objetos combinado con el modelo.

Tabla 2.12 Comparación de parámetros entrenados bajo el conjunto de datos de IP102 *(Zhou & Su, 2020)*

Model	Params (M)	FPS (img/s)	Accuracy (%)
DenseNet121	7.05	684.80	56.11
EfficientNet-b0	4.13	1031.88	44.63
ResNet18	11.22	1577.33	46.85
ResNet50	23.71	767.52	46.79
ResNet101	42.70	500.53	48.90
ShuffleNetV2	5.55	1686.72	43.63
MobileNetV3-Large	4.33	1612.18	47.44
GhostNet	4.03	1768.49	39.68
SqueezeNet	0.77	1944.88	38.26
ExquisiteNet	**0.98**	**1933.24**	**52.32**

Insect pests recognition based on deep transfer learning models (Khalifa et al., 2020)

Se utilizó el conjunto de datos *IP102* (Wu et al., 2019). Para contrarrestar el sobreajuste y con esto evitar la memorización de los datos haciendo más robusto los modelos, se utilizaron técnicas de aumento de imágenes como reflexión del eje *X*, *Y* y *X-Y* con ello se logró un aumento del número de imágenes en cuatro veces del conjunto original obteniendo 300,844 imágenes.

Los modelos contemplados para utilizar con arquitecturas CNN de aprendizaje por transferencia son *AlexNet*, *SqueezeNet* y *GoogLeNet*. Se seleccionaron estos modelos por tener menos capas reduciendo así la complejidad de los cálculos como lo sería con modelos como *Xception*, *DenseNe*t e I*nceptionResNet*. Los modelos fueron entrenados con un paquete de software (MATLAB) usando como métricas de rendimiento las obtenidas a partir de la matriz de confusión tales como la precisión y otras.

Como resultado de los entrenamientos, el modelo de *AlexNet* tuvo un mejor puntaje con 89.33% seguido de *GoogLeNet* con 88.80% y por último *SqueezeNet* con 67.51%. Comparando los resultados con trabajos anteriores, donde no se realizó un aumento de datos, el rendimiento fue de *AlexNet* con 41.8%, *GoogLeNet* con 43.5%, VGG con 48.2%, *Resnet* con 49.2%, demostrando que se mejoró considerablemente los resultados respecto trabajo original donde presentan IP102 (Wu et al., 2019).

Como recomendación los autores proponen que, para trabajos futuros, se usen arquitecturas de redes neuronales generativas adversas o utilizar arquitecturas como *Xception*, *DenseNe*t e *InceptionResNet*

A Crop Pest Classification Model Using Deep Learning Techniques (Malek et al., 2021)

El modelo propuesto pretende distinguir entre plagas beneficiosas de las nocivas mediante técnicas de aprendizaje profundo, identificando 10 plagas beneficiosas: *geocoris, nabidae, chysopiac, coccinellidae, orius insidiosus*, superfamilias de himenópteros, *mantodae, syrphidae, diptera y perillus bioculatus* y, 10 plaga dañinas: *aphidoidea, aphis fabae, meloidae, leptinotarsa decemlineata, agrotis segetum, alticini, globodera, empoasa fabae, holotrichia y melanotus.* Recopilaron fotos en campos de Bangladesh e imágenes conseguidas de internet; para evitar un sobreajuste utilizaron técnicas de aumento de imágenes como Rotación, Escalado y Transformación en el conjunto de datos original, obteniendo un total aproximado de 9,500 imágenes; utilizando el 25% el conjunto para la validación; todas las imágenes fueron reescaladas a 50x50 píxeles.

La

Meta-architecture	Faster R-CNN	R-FCN		SSD	
Feature extractor	ResNet101	Inception	ResNet101	Inception	MobileNet
Athalia rosae japanensis	0.8321	0.8393	0.7703	0.8218	0.8302
Creatonotus transiens	0.5743	0.5527	0.6083	0.586	0.5951
Entomoscelis adonidis	0.8765	0.8246	0.9161	0.6176	0.6030
Entomoscelis suturalis	0.6562	0.4162	0.4344	0.7555	0.4980
Hellula undalis	0.7563	0.7029	0.7106	0.7738	0.6247
Lipaphis erysimi	0.2149	0.3003	0.3606	0.3772	0.3236
Mamestra brassicae	0.9288	0.8445	0.9280	0.967	0.8431
Meligethes aeneus	0.5375	0.2637	0.3556	0.2247	0.3476
Phyllotreta striolata	0.7604	0.5197	0.6124	0.6275	0.6760
Pieris rapae	0.8143	0.8040	0.8432	0.9111	0.6609
Plutella xylostella	0.8629	0.7890	0.7880	0.8767	0.7846
Psylliodes punctifrons	0.4537	0.4202	0.5368	0.554	0.7593
mAP@0.6	0.6890	0.6064	0.6554	0.6744	0.6288
Time (s)	0.158	0.13	0.148	0.052	0.045
Memory (MB)	191.3	52.9	201.7	60.6	23.6

Tabla *2.15* muestra la configuración del modelo junto con las capas y la activación de cada una representada por un color, los números 48x48x32, 22x22x64, 9x9x128, 2x2x200 y 1x1x64 son el total de neuronas del modelo propuesto.

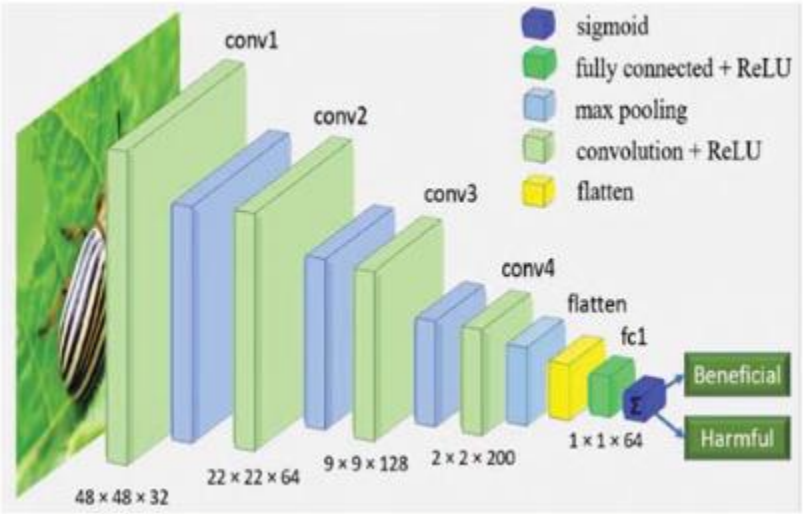


Figura 2.15 Configuración del modelo DL (Malek et al., 2021).

Los autores realizaron dos tipos de comparación, el primero mediante la transferencia de aprendizaje con otros modelos CNN y en el segundo caso fue con respecto a técnicas de aprendizaje automático.

a) Transfer Learning:
    1) VGG: VGG16 tiene un error de 7.3% durante el entrenamiento con el conjunto de validación de *ImageNet*, reemplaza filtros *kernel* 3x3. De las tres capas conectadas, los dos primeros 4,096 nodos y los últimos 1,000 nodos se combinan para crear un clasificador *Soft-Max* (VGG19 funciona igual, pero con más capas y parámetros)
    2) InceptionV3: Entrenó con I*mageNet*, con 48 capas de profundidad capaz de entrenar 1 millón de imágenes a la vez. InceptionV1 debería ser GoogLeNet, InceptionV3 e InceptionV2 son casi igual con ligeras variaciones.
    3) ResNet: es una red residual CNN de 50 capas de profundidad con un error de del 3.75% con un filtro de 3x3 de acceso directo. ResNet50 consta de tres capas de bloques de cuellos de botella con un bloque de 34 capas en cada uno de los bloques.

4) MobileNetV2: usado para dispositivos móviles con una red residual. La relación residual entre dos capas de cuello de botella es una estructura residual invertida

b) Técnicas de aprendizaje automático:
    1) K-nearest neighbor (KNN): es un método de aprendizaje supervisado simple para propósitos de regresión y clasificación. Este enfoque tiene como objetivo principal entrenar el modelo con cierta información por adelantado y establecer dos grupos separados. Cuando se proporcionan nuevos datos calculando su distancia a la siguiente clase, identifican los datos existentes para obtener su distancia
    2) Máquina de vectores de soporte: se utiliza para detectar contornos. El principal objetivo es lograr una función lineal de discriminación, que distingue dos grupos utilizando una superficie de decisión de hiperplano.
    3) Regresión logística: es la conexión entre una variable dependiente y una variable independiente es una regresión probabilística. El resultado potencial puede ser '0' o '1'; para la variable dependiente. Se utiliza para identificar datos rápidamente

Los resultados muestran que el modelo propuesto presenta una mejor exactitud (90%) contra modelos de aprendizaje automático como lo son KNN (81%), Máquina de vectores de soporte (64%), Regresión logística (67%). También fue mejor en comparación con los modelos de transferencia de aprendizaje como VGG16 (86%), VGG19 (82%), InceptionV3 (88%), Resnet50 (69%) y MobileNetV2 (87%)

Classification of pest detection in paddy crop based on transfer learning approach

(Malathi & Gopinath, 2021)

El trabajo realiza la clasificación de 10 tipos de plagas que atacan a la planta del arroz, en lo que se encuentran insectos en su etapa adulta, así como en su etapa de huevo, mediante el enfoque de aprendizaje profundo. Se utilizaron 3,549 imágenes del trabajo de (Alfarisy et al., 2018), pero se aplicó un aumento de datos mediante el ajuste fino de los parámetros en: rotación de 15 grados, desplazamiento de ancho en 0.2, cambio de altura en 0.2, rango de corte en 0.2, zoom en 0.2, y volteo horizontal.

Se probaron distintas arquitecturas previamente entrenadas mostrando los resultados de la Tabla 2.13, donde el afinado mediante ajuste fino en ResNet-50 logró una exactitud superior al resto

de modelos. También se incorpora la curva ROC para calcular el rendimiento de los clasificadores, estimando los parámetros contando verdaderos, falsos positivos y el recuento de negativos de la matriz de confusión, esto con el objetivo de buscar una mayor exactitud.

Tabla 2.13 Resultados de experimentación de las arquitecturas *(Malathi & Gopinath, 2021)*

Model	Training accuracy	Training loss	Validation accuracy	Validation loss	Testing accuracy
AlexNet	0.9113	0.2506	0.9027	0.2890	90.267
GoogleNet	0.922	0.2408	0.9111	0.211	91.02
ResNet34	0.9011	1.7659	0.912	1.067	90.19
ResNet50	0.9003	0.2923	0.9002	0.2849	90.016
Fine-tuned Resnet-50	0.9403	0.123	0.9211	0.1765	95.012

High performing ensemble of convolutional neural networks for insect pest image detection (Nanni et al., 2022)

El trabajo presentado tiene el objetivo de optimizar seis arquitecturas de redes neuronales convolucionales que son *EfficientNetB0, ResNet50, GoogLeNet, ShuffleNet, MobileNetV2* y *DenseNet201*, con el objetivo de clasificar tres conjuntos de datos de insectos catalogados como plagas. Para lograr la optimización se busca evitar el problema donde el gradiente se desvanece lo que ocasiona que los pesos de las redes no se encuentren suficientemente bien entrenados. La solución que proponen es utilizar dos variantes del optimizador Adam, denominadas como Exp y EXPLR, donde la primera está basada en DGrad y hace que la curva de aprendizaje sea más pronunciada y concreta mientras que la segunda propuesta es una variación de la primera, pero con un factor multiplicativo en la tasa final de aprendizaje para obtener una mejor distribución.

El conjunto seleccionado para el entrenamiento fue SMALL mientras que *IP102* y D0 fueron usados para la etapa de validación. Como el conjunto SMALL es simple, se utilizó un aumento de datos mediante la reflexión y escala aleatoria en ambos ejes. Los resultados muestran que las variaciones del optimizador Adam generan mejores resultados que el original con 94.75 de exactitud, por lo cual probó este para el conjunto *IP102* y, para el conjunto DO, obteniendo 0.73 de exactitud y 99.81 de exactitud respectivamente.

An intelligent VegeCareAI tool for next generation plant growth management (Ikeda et al., 2021)

El artículo presenta un sistema integrado de VegeCareAi tool on Edge devide y VegecareAI system on Cloud, como se muestra en la Figura 2.16. Donde el primero es una aplicación móvil para dispositivos android que presenta 3 funcionalidades básicas: a) clasificación de vegetales (6 especies: berenjena, limón, cebolla, papa, shiso y camote); b) una clasificación de enfermedades en las plantas (2 especies: maíz y papa) y c) una clasificación de plagas de insectos (4 especies para los cultivos de arroz y maíz). El segundo componente del sistema se encuentra en la nube donde se procesa el modelo en tiempo real mediante 3 funcionalidades: módulo informático, gestión del crecimiento de la planta y planificador de trabajos.

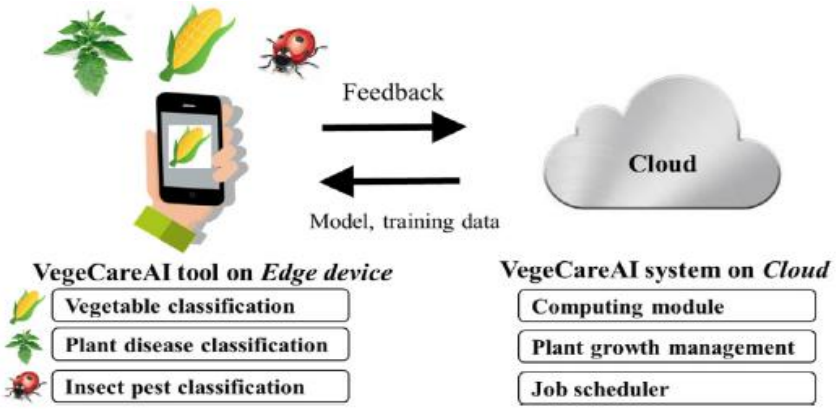


Figura 2.16 Modelo del sistema VegecareAI (Ikeda et al., 2021)

Para propósito de esta investigación, el resumen se centrará en la función de la clasificación de insectos. Los autores se enfocaron en los insectos de rodillo de hojas de arroz, salta hojas de arroz, barrenador del maíz y gusano soldado. Utilizan 2,233 imágenes con técnicas de aumento de datos de las 4 etapas del ciclo de vida del insecto (huevo, larva, pupa y adulto) del Conjunto de datos CPAF-Dataset.

El modelo propuesto por ellos consta de 4 capas convolucionales, 4 capas de agrupación y 4 capas completamente conectadas con la función de activación ReLU. Aunque el modelo presentó valores superiores al 70% de exactitud para todas las enfermedades y mayores al 96% de exactitud para la identificación del cultivo, esto no ocurre en el caso de la clasificación de los insectos en sus distintas etapas como se muestra en la Tabla 2.14. Mencionan que puede ser debido a la semejanza en las etapas de evolución de los insectos por lo que necesitan más datos para el entrenamiento y con ello aumentar la exactitud, en especial, en la etapa de huevo, larva y pupa.

Tabla 2.14 Resultados de prueba para diferentes ciclos de vida de los insectos de plaga (Ikeda et al., 2021)

Class	Sub-class	Quantity	Train.	Valid.	Test	Accuracy
IP1: Rice leaf roller	Eggs	27	19	3	5	40.00
	Larva	317	222	32	63	49.21
	Pupa	54	38	5	11	63.63
	Adult	140	98	14	28	71.43
IP2: Rice leafhopper	Eggs	13	9	1	3	33.33
	Nymph	34	24	3	7	28.57
	Adult	203	142	20	41	43.90
IP3: Corn borer	Eggs	80	56	8	16	68.75
	Larva	431	302	43	86	80.23
	Pupa	26	18	3	5	60.00
	Adult	264	185	26	53	64.15
IP4: Armyworm	Eggs	13	10	1	2	0.00
	Larva	451	316	45	90	77.78
	Pupa	14	10	1	3	33.33
	Adult	166	116	17	33	75.76
Total		2233	1565	222	446	

2.2.3 Detección de insectos

Localization and Classification of Paddy Field Pests using a Saliency Map and Deep Convolutional Neural Network (Z. Liu et al., 2016)

El trabajo presenta un sistema para la detección de 12 tipos de plagas de insectos de arrozales y la modificación de la arquitectura *Alexnet*. Los autores crearon el conjunto de datos Pest ID que se obtuvo adquiriendo imágenes mediante búsquedas en *Google, Naver* y *Fresh Eye*; fotografías a las cuales les aplicó un aumento de datos obteniendo más de 5,000 imágenes normalizados a 256x256 píxeles donde 1,210 son usadas para la validación. Una submuestra de este conjunto de datos se observa en la Figura 2.17.

El conjunto está constituido por imágenes donde los insectos muestran un alto contraste en relación con el fondo por lo que se optó por el método de detección de regiones sobresalientes basada en contraste global que, en sí es una localización basada en regiones destacadas. Dado que los insectos presentan índices de prominencia altos es posible representar los tres canales RGB en uno sólo y posteriormente obtener 10 características para su segmentación, basada en grafos, y con esto generan mapas de prominencia para cada región y con el algoritmo de GrabCut separar el insecto del fondo para encontrar su localización lo que reduce los falsos positivos y mejora la exactitud. Ejemplo de las etapas del sistema se observa en la Figura 2.18.

Para el modelo de DCNN se modificó la arquitectura de *AlexNet* para mejorar la exactitud y acelerar el tiempo de entrenamiento, eliminando y ajustando capas redundantes, y aumentando los campos receptivos locales por la falta de variabilidad de consistencia que presenta cada

insecto. Con estos cambios pasaron del 83.4% de exactitud sin modificar nada al 95.1%, superando también modelos tradicionales de clasificación como SVM con 80.2%, Fisher con 81.17% e incluso la estructura general de *AlexNet* de 92.3% de exactitud.

Figura 2.17 Submuestra de conjunto de datos (Z. Liu et al., 2016).

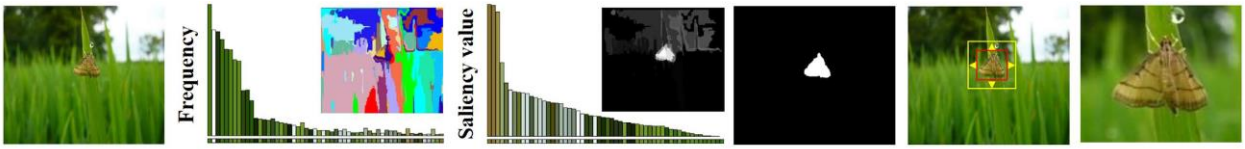


Figura 2.18 Seguimiento del proceso para la detección de PestID (Z. Liu et al., 2016).

Application of Deep Learning in Integrated Pest Management: A Real-Time System for Detection and Diagnosis of Oilseed Rape Pests (He et al., 2019)

Se presentó una aplicación para dispositivos Android llamada Dr. Pest. La interfaz se muestra en la Figura 2.19, y pretende la detección de 12 especies de insectos plaga de la Colza (Rapeseed) que tiene la ventaja que puede capturar, detectar e identificar al insecto para luego mostrar los resultados con información detallada de la especie, todo en el mismo dispositivo.

El conjunto de datos se obtuvo mediante la captura de imágenes en el campo mediante un laboratorio y fotografías descargadas de internet, obteniendo un total de 3,022 imágenes de insectos que afectan este cultivo. Debido a que se presentó un desequilibrio entre las clases se optó por un aumento de datos pasando a 10,575 imágenes, utilizando métodos de fluctuación de color, rotación y recorte aleatorio para el mejorar la detección en distintas condiciones de iluminación, brillo, contraste, saturación y la influencia del fondo; lo que mejoró significativamente la exactitud de los modelos entrenados. Sin embargo, algunas clases presentaron un

sobreajuste, para evitar esto se utilizó la deserción durante el entrenamiento que implica apagar aleatoriamente un porcentaje de neuronas durante el entrenamiento.

Dado que los insectos suelen tener un tamaño bastante reducido, se tomó como arquitectura de detección de objetos a los modelos SSD sobre Faster R-CNN y R-FCN. El modelo óptimo para la función multiescala fue SSD al presentar una precisión buena en el menor tiempo; y se eligió como extractor de características a Inception por ser el óptimo, como se observan los resultados en la Tabla 2.15

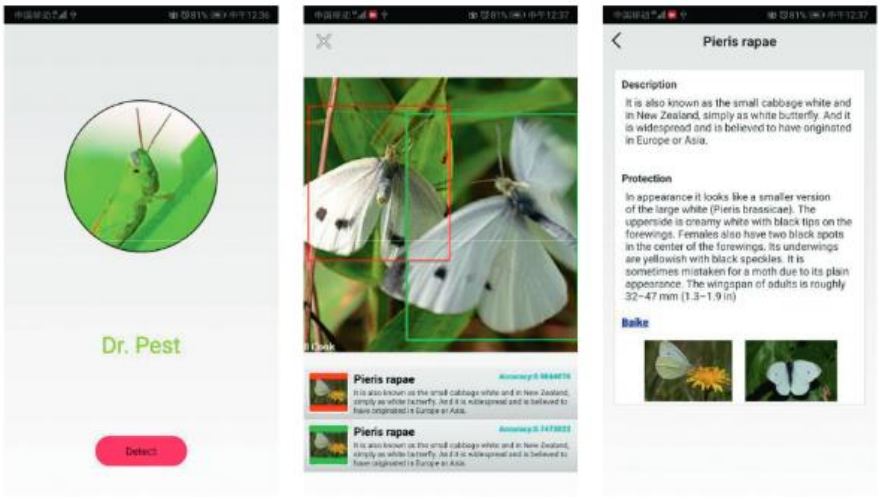


Figura 2.19 Interfaz de Dr. Pest (He et al., 2019).

Meta-architecture	Faster R-CNN	R-FCN		SSD	
Feature extractor	ResNet101	Inception	ResNet101	Inception	MobileNet
Athalia rosae japanensis	0.8321	0.8393	0.7703	0.8218	0.8302
Creatonotus transiens	0.5743	0.5527	0.6083	0.586	0.5951
Entomoscelis adonidis	0.8765	0.8246	0.9161	0.6176	0.6030
Entomoscelis suturalis	0.6562	0.4162	0.4344	0.7555	0.4980
Hellula undalis	0.7563	0.7029	0.7106	0.7738	0.6247
Lipaphis erysimi	0.2149	0.3003	0.3606	0.3772	0.3236
Mamestra brassicae	0.9288	0.8445	0.9280	0.967	0.8431
Meligethes aeneus	0.5375	0.2637	0.3556	0.2247	0.3476
Phyllotreta striolata	0.7604	0.5197	0.6124	0.6275	0.6760
Pieris rapae	0.8143	0.8040	0.8432	0.9111	0.6609
Plutella xylostella	0.8629	0.7890	0.7880	0.8767	0.7846
Psylliodes punctifrons	0.4537	0.4202	0.5368	0.554	0.7593
mAP@0.6	0.6890	0.6064	0.6554	0.6744	0.6288
Time (s)	0.158	0.13	0.148	0.052	0.045
Memory (MB)	191.3	52.9	201.7	60.6	23.6

Tabla 2.15 Resultados de las arquitecturas probadas en (He et al., 2019)

Insect classification and detection in field crop using modern machine learning techniques (Kasinathan et al., 2021)

El objetivo del trabajo es detectar y clasificar insectos en campos de cultivo, mediante técnicas de aprendizaje de máquina. El trabajo utiliza una porción del conjunto de datos Wang (J. Wang et al., 2012) que cuenta con 9 clases y 225 imágenes y Xie1. Para la clasificación de los insectos se comparan cinco técnicas de aprendizaje automático que son: ANN, SVM, KNN, NB y una CNN. La arquitectura utilizada se muestra en la Figura 2.20 para distintos tipos de clases. Las Figura 2.21 y Figura 2.22 muestran los resultados correspondientes al conjunto de datos de Wang y de Xie1 respectivamente. Los autores mencionan que sus experimentos, comparados con otros artículos del estado del arte, tienen mejores resultados con los mismos clasificadores. Aplicando técnicas de aumento pasando de Xie1 a 6,280 imágenes y Wang a 1,296 imágenes.

Aunque el modelo CNN tarda más tiempo para la obtención del modelo, muestra resultados significativamente mejores que el resto de las técnicas de aprendizaje automático. Considerando que el mejor modelo obtenido fue con la CNN, se comparó el tiempo de cálculo para los conjuntos de Wang, Xie1, Deng2018 (submuestra con 9 clases y 282 imágenes) e IP102 (una submuestra de 12 clases y 50 imágenes cada una) obteniendo un tiempo de 5, 38, 6 y 9 minutos, la variación del cálculo se debe a la cantidad de insectos de cada conjunto.

Además, los autores consiguieron detectar el insecto, para esto normalizaron las imágenes a 227x227 píxeles, extraen el insecto del fondo para generar una máscara con el algoritmo de GrabCut iterando 5 veces y aplican una ecualización de histograma y resolución binaria invertida donde se invierten los colores binarios y as para encontrar el contorno más grande que es el insecto y calcular su área delimitada que es la sección más corta. Cabe desacatar que este sistema solo funciona bien cuando solo hay un insecto presente en la imagen.

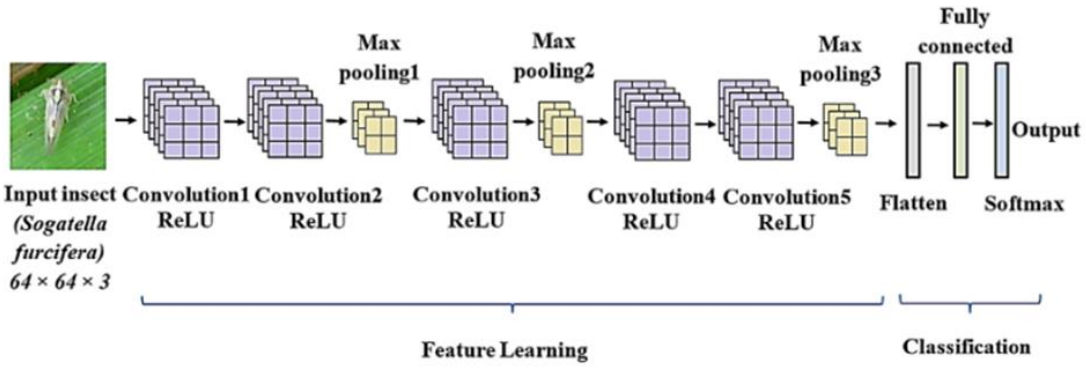


Figura 2.20 Arquitectura CNN propuesta en (Kasinathan et al., 2021)

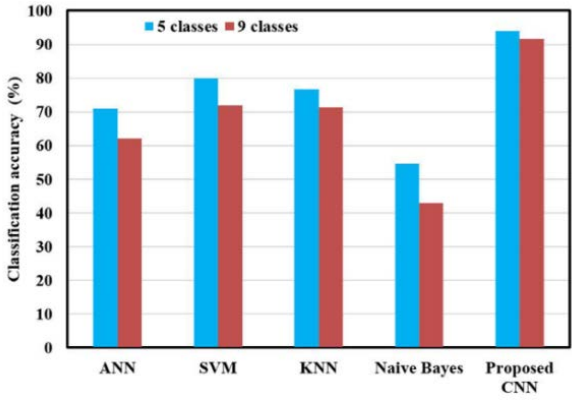


Figura 2.21 Comparación de clasificación de plagas en el subconjunto de datos Wang2018 (Kasinathan et al., 2021)

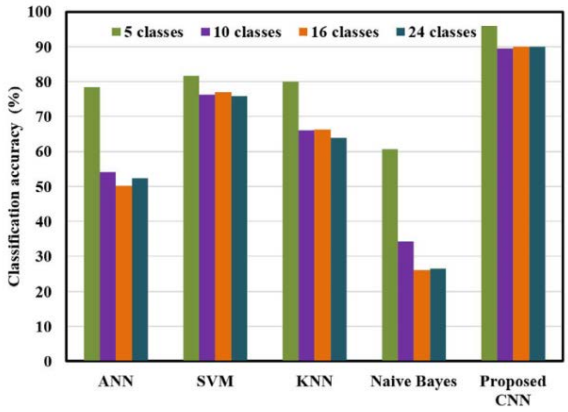


Figura 2.22 Comparación de clasificación en el subconjunto de datos Xie1 (Kasinathan et al., 2021)

Insect Detection and Identification using YOLO Algorithms on Soybean Crop

(Verma et al., 2021)

El trabajo se centra en dos puntos claves, la identificación de insectos en el cultivo de la soja y la identificación de la ubicación del insecto en la imagen, para esto se centra en 5 clases diferentes de insectos utilizando 3 versiones de la arquitectura de YOLO, la versión YOLOv3, YOLOv4 y YOLOv5.

Para llevar a cabo el entrenamiento se recopilaron imágenes del insecto en distintos entornos y escenarios obteniendo 3,710 imágenes, las cuales fueron etiquetadas manualmente con cuadros delimitadores, al ser pocas imágenes se optó por un aumento de datos. Para YOLOv3 y YOLOv4, las imágenes se normalizaron a un tamaño de 416 píxeles de ancho y largo mientras que para YOLOv5 fue de 640.

Los resultados muestran que la versión YOLOv5 se desempeña mejor que sus versiones anteriores quedando con peor resultado YOLOv3 que identifica mejores objetos grandes, pero tiene problemas para la identificación de plagas con un menor tamaño e identifica la presencia de más de un insecto provocando falsos positivos. El rendimiento de los modelos se evaluó bajo métricas tradicionales; además comparan los resultados obtenidos contra el estado del arte, tomando en cuenta trabajos similares como se muestra en la Tabla 2.16.

Tabla 2.16 Comparación de arquitecturas para la detección de insectos (Verma et al., 2021).

Técnica	**Métricas**			
	mAp@0.5 (%)	F1-Score(%)	Precisión (%)	Recall (%)
Fast RCNN	79.64	-	-	-
YOLOv3	83	95	75	83.82
SDD	85.35	-	-	-
Improved CNN	89.22	-	-	-
DL for soyabean insect counting	-	90	-	-
YOLO-v4	94	99	93	96
YOLO-v5	99.5	93.2	99.6	96

Custom CornerNet: a drone-based improved deep learning technique for large-scale multiclass pest localization and classification (Albattah et al., 2023)

Se introdujo un enfoque basado en drones llamado custom CornerNet. CornerNet (Law & Deng, 2018) funciona como un detector de una sola etapa que localiza ROIs usando estimación de puntos claves los cuales conforman la caja delimitadora: el punto superior izquierdo y el punto inferior derecho. Además, relacionan los puntos clave con la extracción de características usando mapas de calor, embeddings, offset y la clase para determinar si la distancia de los puntos es parte de las características. Se modificó la arquitectura para que la extracción de características sea dada por otro modelo en lugar de la original que es Hourglass104. Para ello, se entrenó a 10 extractores de características y el que obtuvo mejor exactitud es DenseNet-100 (Huang et al., 2017) como se observa en la Tabla 2.17 por lo que se utilizó este modelo reduciendo la complejidad computacional hasta en un 96%. Para la eliminación de las cajas de predicción superpuestas se utilizó soft-NMS.

Para probar la arquitectura se utilizó el conjunto de datos IP102 normalizando las imágenes a 224x224 píxeles. En la parte de detección se utilizaron valores de IOU de 0.578 y 0.621 en las 8 categorías jerárquicas de IP102 donde se compararon con otros modelos de detección mostrados en la Tabla 2.18. Además, para probar el rendimiento en su extracción de características se entrenó dos clasificadores de ML, SVM y KNN obteniendo los resultados de laTabla 2.19.

Tabla 2.17 Comparación de extractores de características en IP102 (Albattah et al., 2023).

Method	Parameters (million)	Accuracy ± STD (%)
AlexNet	62.3	41.8 ± 1.51
GoogleNet	7.8	43.5 ± 1.01
VGGNet	138	48.2 ± 2.09
ResNet-50	23.72	57.39 ± 1.35
ResNet-101	42.63	53.18 ± 1.05
Inception V4	41.2	47.8 ± 1.64
HourGlass104	187	54.63 ± 1.44
EfficientNet	19.4	60.2 ± 1.02
DenseNet-121	8.06	54.71 ± 1.45
DenseNet-100	7.08	68.74 ± 0.82

Tabla 2.18 Comparación de Custom CornerNet contra modelos tradicionales de detectores de objetos en IP102 (Albattah et al., 2023).

Method	Backbone	mAP	Test time (s/img)
Two-stage detectors			
Fast R-CNN	VGG-16	46.12	2
Fast R-CNN	DenseNet-100	47.34	1.73
Faster R-CNN	VGG-16	47.87	0.45
Faster R-CNN	DenseNet-100	48.12	0.41
One-stage detectors			
Refinedet	VGG-16	49.01	0.30
Refinedet	DenseNet-100	50.25	0.28
SSD	VGG-16	47.21	0.36
SSD	DenseNet-100	49.17	0.33
YOLOv3	DarkNet-53	50.64	0.29
YOLOv3	DenseNet-100	50.95	0.27
Proposed (Custom CornerNet)	DenseNet-100	57.23	0.23

Tabla 2.19 Comparación de Custom cornerNet con clasificadores de ML (Albattah et al., 2023).

Methods	Accuracy ± STD (%)	
	SVM	KNN
ResNet-50	49.5 ± 1.69	49.4 ± 2.28
EfficientNet	51.9 ± 1.07	51.7 ± 1.86
DenseNet-100	52.5 ± 0.97	50.4 ± 1.19
Proposed (Custom CornerNet)	68.74± 0.82	

2.3 Discusión estado del arte

En la **Tabla 2.20** se presenta un resumen del estado del arte

Tabla 2.20 Estado del arte revisado

Artículo	Tarea	Conjunto de datos	Imágenes	Clases	Algoritmo	Resultado (%)	Aportación
(Xie et al., 2015)	cls	Xie1	1,440	24	MKL	90.7 exactitud	Conjunto de datos, extracción de características sin necesidad de características manuales
(Dr. J. H. Kulkarni et al., 2018)	cls	NBAIR	2,953	1,329	N/A	N/A	Conjunto de datos,
(Xie et al., 2018)	cls	Xie2	4,500	40	MKL, SVM	92.1 exactitud	Conjunto de datos, extracción a partir de varios niveles con clasificadores de ML
(Deng et al., 2018)	cls	Deng2018	563	10	LCP, SIFT-HMAX, SVM	85.5 exactitud	Conjunto de datos,
(Wu et al., 2019)	cls, det	IP102	75,000	102	SVM y Resnet (cls) FPN (det)	49.5 exactitud (cls) 54.93 AP50 (det)	Conjunto de datos y comparación de técnicas de ML vs DL
(Tetila, MacHado, et al., 2020)	cls	INSECT10K7C	7	10,000	DenseNet-201 con ajuste fino	94.89 exactitud	Conjunto de datos, ajuste fino vs aprendizaje por transferencia vs aprendizaje desde cero
(J. Wang et al., 2020)	cls	CPAF-Dataset	973	19	CPAFNet	92.26 exactitud	Conjunto de datos, extracción de características de manera automática
(Y. Li et al., 2020)	cls	Li'Dataset	5,629	10	GoogLeNet	98 exactitud	Conjunto de datos
(Q. J. Wang et al., 2020)	det	Pest24	25,378	24	YOLOv3	63.54 mAp@0.5	Conjunto de datos centrado en la detección de múltiples objetos

Tabla 2.20 Estado del arte revisado (continuación)

Artículo	Tarea	Conjunto de datos	Imágenes	Clases	Algoritmo	Resultado (%)	Aportación
(R. Wang et al., 2021)	det	AgriPest	49,7000	14	Cascade R-CNN	60.48 – 90.92	Conjunto de datos
(Thenmozhi & Srinivasulu Reddy, 2019)	cls	NBAIR Xie1 Xie2			Propio (CNN)	96.75 exactitud 97.47 exactitud 95.97 exactitud	Arquitectura CNN
(Reza et al., 2019)	cls	IP102	75,000	102	VGG19	55.35 exactitud	Rendimiento en IP102
(Song et al., 2019)	cls	propio	35,000	71	Inception-v4	97.3 exactitud	Comparative de técnicas de DP vs ML
(Dawei et al., 2019)	cls	propio	484	10	Alexnet	93.84 exactitud	Comparativa de técnicas de ML vs humanos expertos
(Ayan et al., 2020)	cls	D0 Deng2018 IP102			Inception-v3, Xception y MobileNet con GAEnsemble como ensamble	98.81 exactitud 95.16 exactitud 67.13 exactitud	Modelo de votación para ensamble de resultados en la clasificación
(Tetila, Machado, et al., 2020)	cls	propio	5,300	13	SLIC con Resnet-50	93.82 exactitud	Comparación de técnicas de DL y ML
(W. Liu et al., 2020)	cls	IP102			DFF-ResNet	55.39 exactitud	Propuesta de una red neuronal
(Zhou & Su, 2020)	cls	IP102			ExquisiteNet	52.32 exactitud	Propuesta de una red neuronal
(Khalifa et al., 2020)	cls	IP102	300,844	102	AlexNet	89.33 exactitud	Utilización de aumento de datos para mejora de resultados

Tabla 2.20 Estado del arte revisado (continuación)

Artículo	Tarea	Conjunto de datos	Imágenes	Clases	Algoritmo	Resultado (%)	Aportación
(Malek et al., 2021)	cls	propio	9, 500	20	Arquitectura propia CNN	90 exactitud	Arquitectura CNN propuesta y comparación de técnicas de DP vs ML
(Malathi & Gopinath, 2021)	cls	Alfarisy2018	3,549	10	Resnet-50 con ajuste fino	95.01 exactitud	Aplicación de ajuste fino para mejorar resultados
(Nanni et al., 2022)	cls	IP102 D0				73.00 exactitud 99.81 ecatitud	Se propusieron optimizadores para mejorar el aprendizaje
(Ikeda et al., 2021)	cls	CPAF-Dataset	2,233	15	VegeCareAI	73 exactitud	Sistema con funcionamiento en la nube y en el celular
(Z. Liu et al., 2016)	det	Pest ID	5,000	12	Modificacion de AlexNet	95.1 exactitud	Modificación de la arquitectura Alexnet y comparativa de ML vs DL y propuesta para la localización
(He et al., 2019)	det	propio	10,575	12	Inception-SDD	67.44 mAp@0.6	Comparación entre detectores de objetos y modelo en dispositivos móviles
(Kasinathan et al., 2021)	det	Wang Xie1	6,280 1,296	9 24	CNN	91.5 90	Comparativa de técnicas de ML vs DL, propuesta de detección
(Verma et al., 2021)	det	Propio	3,710	5	YOLO-V5	99.5 mAP@0.5	Comparación de arquitecturas de una sola etapa
(Albattah et al., 2023)	det cls	IP102		102	Custom CornerNet (det) Custom CornerNet, SVM(cls)	57.23 mAp@0.6 (det) 68.74 Exactitud	Modificación de CornerNet para reducir tiempo complejidad computacional

Dentro del análisis realizado del estado del arte se observó diversas variables importantes para la detección y clasificación de plagas. Los puntos a destacar son: el conjunto de datos, tareas de detección y clasificación, así como las técnicas utilizadas.

En la revisión de la literatura el conjunto de datos que cuenta con mejores características es el conjunto de datos de IP102 por presentar insectos en ambientes naturales y no bajo alguna trampa o condiciones de laboratorio. Además de presentar mejor relación de clases por cantidad de imágenes y sobre todo, el hecho de estar etiquetadas por expertos.

Se observó una tendencia de presentar mejores resultados las técnicas realizadas con aprendizaje profundo que con técnicas tradicionales de aprendizaje automático tanto en la extracción de características como en la clasificación de los insectos por lo que se seleccionó una de estas arquitecturas para el empleo la identificación de plagas.

CAPÍTULO 3

Análisis, diseño e implementación del sistema

En esta sección se describen cada una de las etapas de la metodología de solución propuesta, las cuales consisten en: adquisición de datos, procesamiento, entrenamiento, validación e inferencia.

3.1 Etapas de metodología de solución

3.1.1 Adquisición de datos

El conjunto de datos utilizado fue el *IP102*(Wu et al., 2019) que se encuentra construido para dos tipos de tareas, clasificación y detección. En la parte de clasificación consta de 102 clases distribuidas en 75,222 imágenes, mientras que en la detección se compone de 97 clases en 18,981 imágenes donde cada imagen cuenta con un archivo .xml que contiene información como las propiedades de la imagen, la clase y el cuadro delimitador del insecto. La distribución se muestra en la Figura 3.1, donde se observa una discrepancia entre la cantidad de objetos y la cantidad de imágenes.

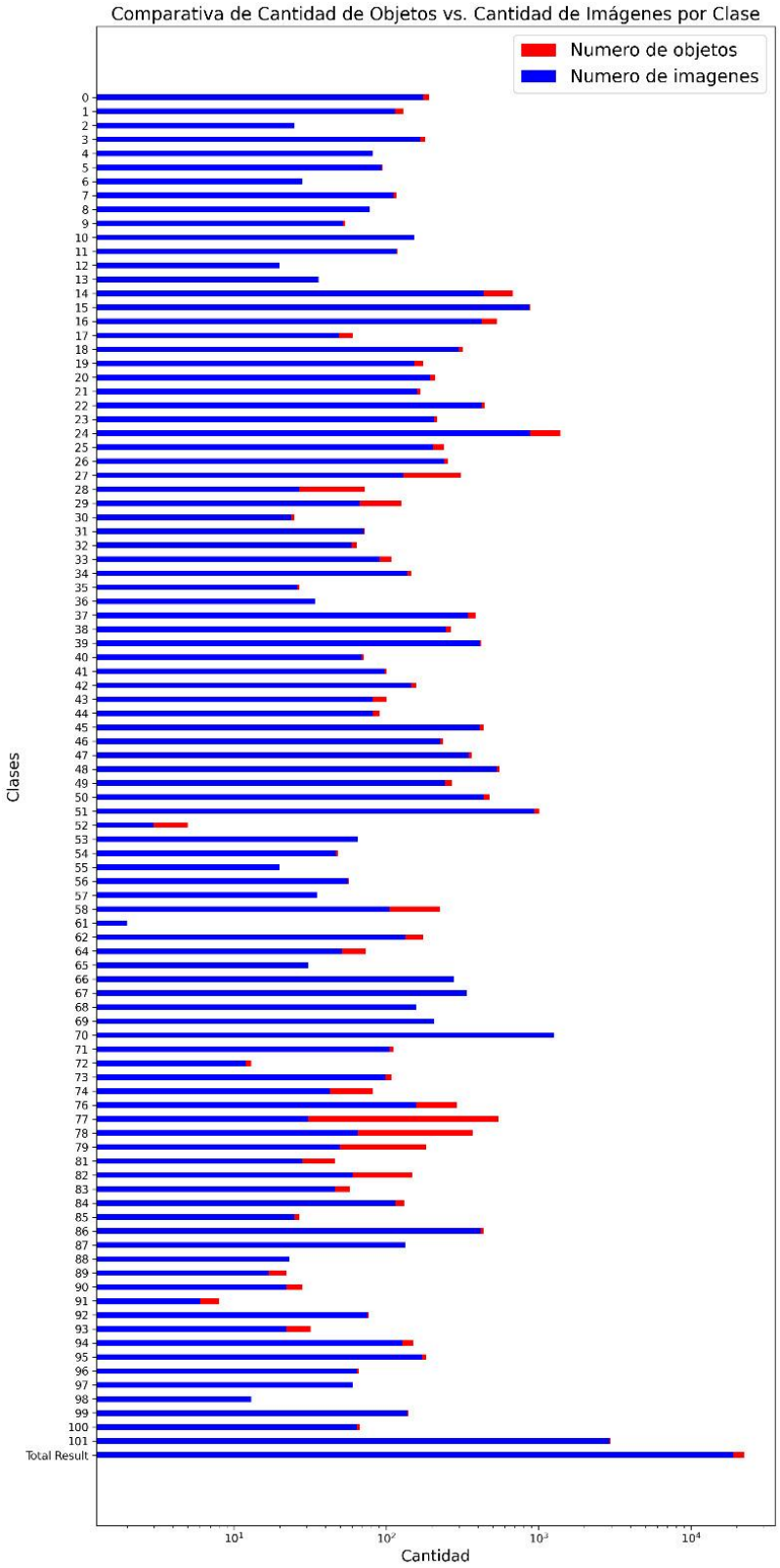


Figura 3.1 Distribución de objetos e imágenes en el conjunto de datos de IP102

Las características que presenta el conjunto de datos *IP102* son variadas y no se limitan a aspectos de escala de la imagen, iluminación, contraste, también incluye cambios en la taxonomía tanto en la misma clase como en las distintas clases. Estos cambios se observan en el color, así como en la distribución y la perspectiva, en la Figura 3.2 y 3.3, se muestran algunos de los ejemplos mencionados previamente.

Diferencia taxonómicas

Papilio xuthur (Clase 70)

Oides decempunctata (Clase 60)

Limacodidae (Clase 58)

Menor diferencia entre clases

Rice leaf roller (Clase 1)

Corn borer (Clase 22)

Meadow moth (Clase 41)

Diferencia por metamorfosis

Rice leaf roller (Clase 1)

Figura 3.2 Ejemplos de la dificultad de conjunto de datos *IP102*. En la primera fila se observa las diferencias taxonómicas, en la segunda fila la menor diferencia entre clases, y en la tercera la diferencia por metamorfosis. (Wu et al., 2019)

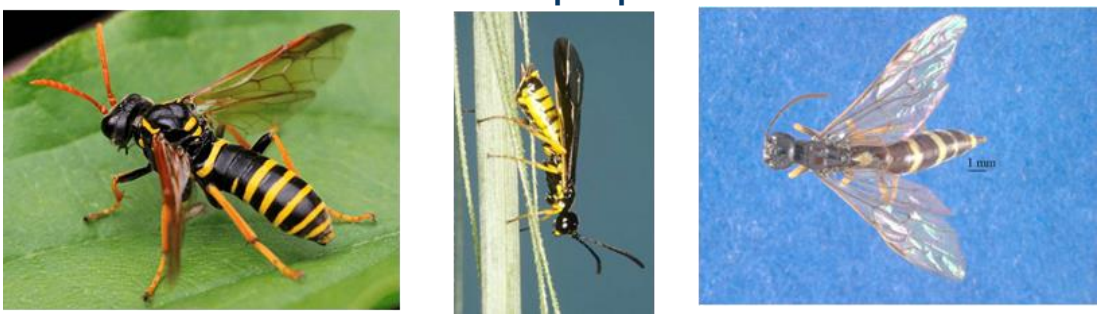
Cambios policromáticos

Cicadellidae (Clase 97)

Múltiples distribuciones

Grub (Clase 14)

Cambios de perspectiva



Wheat sawfly (Clase 34)

Mimetismo

Salurnis marginella Guerr (Clase 90) Locustoidea (Clase 48) Lytta polita(Clase 49)

Figura 3.3 Ejemplos de la dificultad de conjunto de datos *IP102*, en la primera fila se presentan cambios policromáticos, en la segunda múltiples distribuciones, en la cuarta fila se encuentra cambios de perspectiva y, por último, en la quinta fila mimetismo. (Wu et al., 2019)

3.1.2 Procesamiento

Primero, se efectuó un cambio en las etiquetas de las imágenes dado que consta de un archivo .xml, que contiene información como: *xmin, ymin, xmax, ymax* y una etiqueta para el nombre. Para el entrenamiento de los modelos de la familia YOLO se requiere un archivo .txt donde cada fila consta de: la clase, el centroide *x*, el centroide *y*, el ancho y el largo de la caja contenedora de insecto, cuyos valores están normalizados de 0 a 1.

Se utilizó la herramienta *Fastdup* (Bickson Danny et al., n.d.), que provee un método eficiente y rápido para la detección de agrupaciones de imágenes duplicadas o similares para su posterior eliminación. De esta manera, aumentar la calidad del conjunto de datos al no presentar imágenes iguales, por lo que la red a entrenar no tenderá a memorizar estos objetos. En la Figura 3.4, se presenta un ejemplo de dos casos que se encontraron en el *IP102*, cuando una imagen está repetida en la misma clase y cuando una imagen está repetida en distintas clases.

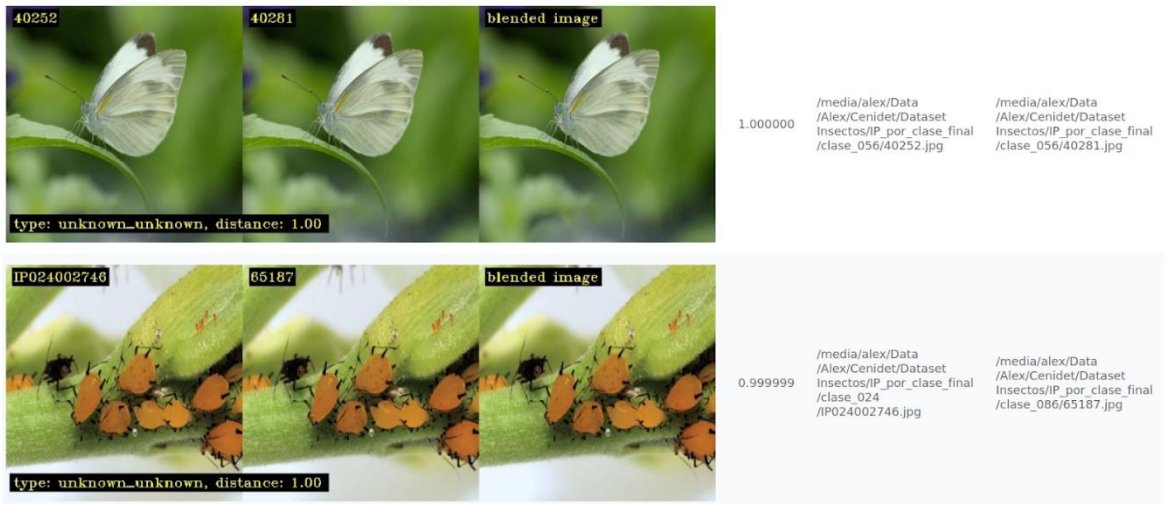


Figura 3.4 Ejemplo de búsqueda de similaridad en imágenes con *fasdup* en *IP102 donde se muestra una coincidencia exacta en dos imágenes*

Después, las imágenes fueron escaladas a un tamaño de 640 x 640 píxeles tanto de ancho como de largo. Se empleó la herramienta LabelImg (*LabelImg. Git Code,* 2015) para etiquetar aquellos insectos que no contaban con su respectivo cuadro delimitador, así como ajustar algunos cuadros delimitadores para que se ajustara mejor a la forma del insecto.

Además, se realizó un aumento de datos, esto se detalla en el caso C, que corresponde al "Capítulo 4 Validación, experimentación y análisis de resultados". A continuación, se presenta en la Tabla 3.1 las operaciones realizadas en el aumento de datos.

Tabla 3.1 Operaciones y probabilidades usadas en el aumento de datos.

Operación	Probabilidad	Rango
Volteo horizontal	0.3	No aplica
Volteo vertical	0.3	No aplica
Cambio en iluminación	0.9	±0.5
Contraste	0.9	±0.5
Cambio en saturación (HSV)	0.9	±30
Cambio en valor (HSV)	0.9	±20
Rotación	1	±45
Escala	1	-0.2 a 0.5

3.1.3 Entrenamiento

Se seleccionaron tres arquitecturas: la arquitectura yolov5m6, yolov5L6 y yolov7, el criterio de selección se presenta en el “Capítulo 4 Validación, experimentación y análisis de resultados”. Cada una de las arquitecturas se entrenó con el conjunto de datos con un total de 9,965 imágenes, con las mismas técnicas de aumento de datos, donde se fijó como límite siete imágenes de manera individual y 500 para cada clase. Debido a que para algunas categorías el límite se sobrepasaba también se aplicó un submuestreo, en la sección de casos, se detalla las configuraciones realizadas.

3.1.4 Validación

En esta sección, se validó cada uno de los entrenamientos realizados con un subconjunto del conjunto de datos cuyas imágenes no se presentaban en la etapa de entrenamiento contando con 3,984 imágenes. Con el propósito de conocer el rendimiento del sistema se emplearon las métricas previamente mencionadas en la sección 1.2.5, así como las validaciones de los tres modelos trabajando a la vez mediante un ensamble en sus cajas delimitadoras predichas.

3.1.5 Inferencia

Para el uso del sistema propuesto se implementaron dos interfaces con el propósito de que un usuario utilice de manera intuitiva el sistema. Una de ellas es para usar el sistema en dispositivos móviles, gracias a la facilidad de exportación del modelo creado en la versión de YOLOv5m6 el cual fue el que obtuvo mejor rendimiento y, es la arquitectura que se recomienda para dispositivos móviles al no requerir demasiados recursos de hardware, permitiendo realizar inferencias en tiempo real. La segunda interfaz es para interactuar con el sistema de escritorio; sistemas que

es más robusto al estar integrado por las tres redes implementadas en conjunto; las interfaces se muestran en la Figura 3.5 y Figura 3.6

Figura 3.5 Interfaz para dispositivos móviles (Android)

Figura 3.6 Interfaz para PC

3.2 Análisis y diseño del sistema

En esta sección, se presenta la descripción del funcionamiento del sistema propuesto.

1. La imagen de entrada es escalada y enviada a cada uno de los modelos.
2. Cada uno de los algoritmos localiza y clasifica al insecto de manera individual.
3. La posición de las cajas y las etiquetas son ingresadas a el módulo de ensamble, que realiza la fusión de cajas ponderadas y crea una nueva caja que se usa para delinear la

posición final del rectángulo mínimo para cada insecto presente en la imagen. La caja resultante es la mostrada en la salida, sobre la imagen, como se muestra en la Figura 3.7. Con respecto a la clase el sistema también determina la categoría resultante por votación.

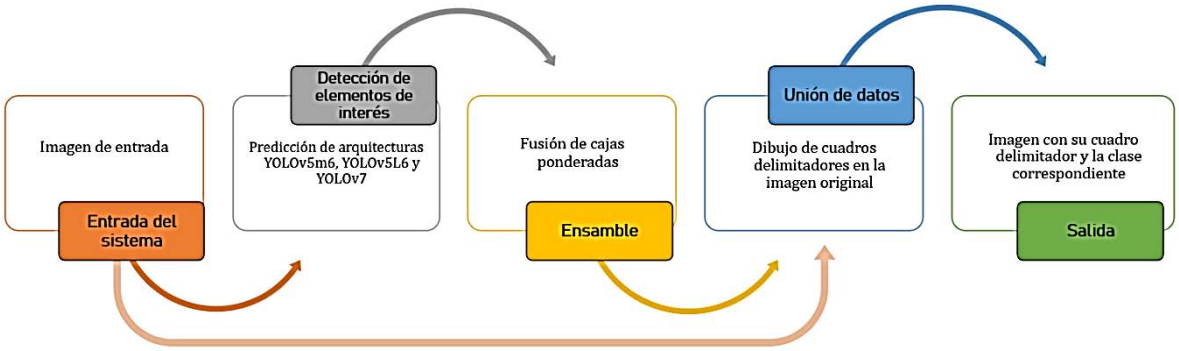


Figura 3.7 Esquemático del sistema desarrollado.

3.2.1 Herramientas utilizadas

A continuación, se enlistan las características del hardware que se empleó para el desarrollo del sistema propuesto y los detalles del software utilizados.

Arquitectura de hardware:

- Laptop Legion 7 16ACHg6
- Tipo de sistema para 64 bits
- Procesador AMD Ryzen 9 5900hx con *radeon graphics* x 16
- Unidad de procesamiento gráfico (GPU) *RTX 3080 mobile* de 16 GB
- Memoria RAM de 32 GB

Arquitectura de software:

- Sistema operativo *ubuntu* 22.04.2 LTS
- *Toolkit* de CUDA versión 12
- *Python* 3.8.13 en un ambiente de Anaconda 3
- YOLOv5 versión 6
- *Android Studio* 2021.2, mínima versión de *SDK Android 9.0* (Pie)

CAPÍTULO 4

Validación, experimentación y análisis de resultados

En esta sección se describen los cinco casos de experimentación realizados en diferentes etapas para obtener el sistema final de identificación de plagas.

Caso A:

Comparar el rendimiento de las distintas versiones de las arquitecturas implementadas y determinar cuál lograba el mejor desempeño con el conjunto de datos *IP102* con 19 clases seleccionadas. El tiempo de entrenamiento fue un factor decisivo para la elección de la arquitectura.

Caso B:

Encontrar el subconjunto de imágenes que proporcionaban mejor información o ser más descriptivas y representativas de cada categoría para las redes, proceso que se realizó mediante la variación de ellas en el proceso de validación de cruzada.

Caso C:

Establecer los límites en la cantidad de imágenes generadas mediante el aumento de los datos.

Caso D:

Determinar cuáles redes proporcionan los mejores resultados, mediante el entrenamiento con la versión óptima del conjunto de datos.

Caso E:

Combinar las mejores arquitecturas de dos maneras diferentes: la primera consiste en agregar un clasificador extra en el cuadro de la detección para convertir la arquitectura de una etapa a dos etapas (detección y clasificación por separado) o realizar un trabajo conjunto en la misma imagen donde la predicción se une por medio de sus cajas delimitadoras de los objetos predichas por los modelos.

4.1 Casos de experimentación

4.1.1 Caso A: Evaluar distintas versiones de YOLO

Objetivo: Comparar tres diferentes arquitecturas (en total 6 versiones) con el dataset *IP102* para conocer su rendimiento, evaluando distintos tipos de aumento de datos para el caso C.

Se realizó la división del conjunto de datos en entrenamiento, pruebas y validación tomando una muestra aleatoria de 70%, 15% y 15% respectivamente. Para las 19 clases seleccionadas se toma una muestra de 500 imágenes para cada clase, y llevando a cabo el aumento de datos.

En la Tabla 4.1 se presentan los resultados obtenidos. Se observa que YOLOv6s logró el mejor rendimiento, aunque su tiempo de entrenamiento fue de 6.5 horas. En contraste, YOLOv5m6 mostró un rendimiento ligeramente inferior con tan solo 0.005 de *mAP@0.5* de diferencia en un tiempo de 4.4 horas lo que se considera una mejora, pero no tan significativa para el tiempo de entrenamiento extra que se requirió. En cuanto a YOLOv7, alcanzó un rendimiento similar a YOLOv5m6, pero requirió de 300 épocas y seis veces más tiempo, si se limitaba a la misma cantidad de épocas que el resto de las versiones (100 épocas) no lograba una estabilización en sus resultados, y se consideraría que el entrenamiento se interrumpió de manera temprana para este caso en particular. Es importante destacar que el aumento de datos es crucial para este tipo de redes, ya que YOLOv5s sin dicho aumento presenta un desempeño muy deficiente incluso proporcionando mayor cantidad de épocas.

Tabla 4.1 Comparativa de las versiones en las arquitecturas probadas para 19 clases

Arquitectura	Aumento de datos	mAP@0.5	Épocas	Tiempo de entrenamiento
YOLOv5s	No	0.698	200	3 horas
YOLOv5s	Si	0.908	100	**1.6 horas**
YOLOv5-transformer	Si	0.908	100	2 horas
YOLOv5-P2	Si	0.859	100	3.2 horas
YOLOv5m6	Si	0.927	100	4.4 horas
YOLOv6s	Si	**0.932**	100	6.5 horas
YOLOv7	Si	0.920	300	22.2 horas

En la Tabla 4.2, se señalan en color verde aquellas categorías en donde el desempeño de la arquitectura mejora sustancialmente en comparación con la versión analizada sin el aumento de datos. Hay que tener en cuenta que esta mejora no ocurre en todas las clases consideradas, puesto que realizando un análisis detallado se encontró que existen clases en donde el aumento de datos disminuye su rendimiento como en las clases señaladas en rojo, debido a las características que presentan. Es decir, la categoría se integra de insectos con un tamaño muy pequeño, en proporción a la totalidad de la imagen e incluso presentaban una distribución cercana entre ellos, por lo que las imágenes creadas en el proceso de aumento producen imágenes de insectos incompletos, con oclusión, etc.

Tabla 4.2 Resultados de mAP@0.5 para cada entrenamiento por clase

ID *IP102*	YOLOv5s Sin aumento	YOLOv5s	YOLOv5-transformer	YOLOv5s-p2	YOLOv5m6	YOLOv6s	YOLOv7
10	0.995	0.941	0.946	0.930	0.977	0.986	0.964
12	0.342	0.979	0.979	0.978	0.991	0.982	0.990
14	0.929	0.943	0.973	0.942	0.980	0.988	0.960
16	0.906	0.958	0.937	0.912	0.977	0.980	0.975
21	0.990	0.991	0.983	0.984	0.986	0.995	0.995
22	0.822	0.822	0.906	0.822	0.893	0.930	0.927
23	0.754	0.838	0.873	0.761	0.885	0.889	0.868
24	0.774	0.868	0.857	0.754	0.931	0.944	0.871
27	0.344	**0.953**	0.942	0.847	0.964	0.969	0.941
29	0.452	**0.986**	0.980	0.909	0.988	0.978	0.987
37	0.712	0.935	0.935	0.898	0.946	0.947	0.948
39	0.809	0.800	0.859	0.703	0.856	0.804	0.817
42	0.457	0.897	0.859	0.852	0.933	0.940	0.922
44	0.822	0.959	0.961	0.901	0.963	0.976	0.974

Tabla 4-2 Resultados de mAP@0.5 para cada entrenamiento por clase (Continuación)

ID *IP102*	YOLOv5s Sin aumento	YOLOv5s	YOLOv5-transformer	YOLOv 5s-p2	YOLOv 5m6	YOLO v6s	YOLO v7
54	0.813	0.725	0.769	0.685	0.748	0.780	0.715
71	0.975	0.788	0.748	0.735	0.821	0.800	0.805
88	0.778	0.995	0.995	0.995	0.995	0.996	0.997
90	0.121	0.994	0.993	0.982	0.995	0.996	0.997
92	0.167	0.822	0.777	0.726	0.783	0.820	0.824
Total	0.6822	0.904	0.909	0.858	0.926	0.931	0.919

4.1.2 Caso B: Validación cruzada

Objetivo: Mediante validación cruzada, determinar el conjunto de imágenes que ayuden a generalizar el modelo YOLOv5m6 que es la arquitectura que en el caso A, obtuvo el mejor desempeño. Para esta prueba se utilizan las 97 clases del conjunto de datos *IP102*. Se dividió el conjunto en una distribución del 75% con 12,090 imágenes para entrenamiento y 25% con 3,984 imágenes para validación. Para la validación cruzada se compararon diferentes valores de *K*, de 0 a 3.

Debido a que para algunas clases presentes en el conjunto de datos de *IP102* se cuenta con pocos elementos, se optó por abandonar el conjunto de pruebas y mezclar este porcentaje al conjunto de validación.

En la Tabla 4.3 se presentan los resultados obtenidos, se puede observar que existe una diferencia en la métrica *mAP@0.5* para la validación cruzada 3, esta métrica es la más relevante para detectores de objetos por lo que se toma esa distribución obtenida respecto a las demás. Asimismo, esta distribución superó al resto en tres de las cuatro métricas solo quedando debajo en precisión y en los siguientes casos este será el conjunto de datos utilizado. Los resultados en este caso son inferiores a los del caso A, pero hay que tener presente que ya no se trabaja con las 19 clases sino con todo el conjunto de las 97 clases.

Tabla 4.3 Comparativa de las métricas para cada caso de validación cruzada en 97 clases

Categoría	Precisión	Recall	mAP@0.5	mAP 50-95
Validación cruzada 0	0.561	0.639	0.599	0.407
Validación cruzada 1	0.551	0.613	0.599	0.420
Validación cruzada 2	0.600	0.606	0.593	0.409
Validación cruzada 3	0.569	0.656	0.619	0.422

4.1.3 Caso C: Optimización en el aumento de datos

Objetivo: Determinar los límites de la cantidad de imágenes generadas mediante el aumento de datos ya que *IP102* cuenta con un conjunto de datos con clases desequilibradas, esto quiere decir que no todas las clases presentan la misma cantidad de imágenes. Para intentar homogenizar las clases se trabajó con dos límites para la creación de imágenes/etiquetas:

Límite por cada clase: esto quiere decir que la cantidad de imágenes de esta clase no puede superar el límite definido en para los distintos casos como 100, 300 y 500 imágenes.

Límite individual: Con esto se entiende que, a una imagen no se le puede aplicar más operaciones de aumento de datos, que el límite individual máximo fijado en 3, 5 o 7 aumentos diferentes dependiendo el caso; sin importar que el límite por clase sea inferior y falten imágenes para alcanzarlo.

Se utilizó la librería *Albumentations* (Buslaev et al., 2020) dado que permite realizar transformaciones a las imágenes manteniendo las cajas delimitadoras en el lugar que corresponde en la imagen transformada, y así poder realizar un aumento de datos con las operaciones como rotación, volteo horizontal, volteo vertical, cambio en brillo y contraste, cambio en la saturación y un escalado aleatorio de acercamiento y alejamiento manteniendo la dimensión inicial de la imagen.

En la Tabla 4.4 se presentan los resultados obtenidos con distintas configuraciones en el aumento de datos mediante el entrenamiento de la red neuronal YOLOv5m6. Los valores obtenidos son más altos que en los casos posteriores debido a que se combinaron los insectos iguales que estaban etiquetados en clases diferentes, provocando tener traslape de información en varias categorías; el proceso se realizó de manera manual. En este caso se realizó el entrenamiento con 72 clases en lugar de las 97 clases, las clases que se unieron fueron aquellas que presentaban características similares al estar en la misma familia, en la Tabla 4.5 se muestra cuáles fueron estas clases.

Como se observa en la Tabla 4.4 el mejor resultado obtenido en *mAP@0.5* resaltado en color azul se dio en el caso donde se utilizaban siete para el aumento de las imágenes como límite individual y 500 para el límite para la clase, por lo tanto, se tomaron estos valores como límites para el aumento de los datos.

Tabla 4.4 Comparativa de resultados con diferentes límites utilizados para el aumento de datos para 72 clases

Límite individual	Límite por clase	Precisión	*Recall*	*mAP@0.5*	*mAP 50-95*
3	300	0.683	0.681	0.716	0.495
3	500	0.647	0.701	0.719	0.495
5	300	0.745	0.672	0.735	0.512
5	500	0.666	0.697	0.729	0.51
7	100	0.687	0.595	0.665	0.459
7	300	0.703	0.666	0.73	0.513
7	**500**	**0.733**	**0.699**	**0.759**	**0.529**

Tabla 4.5 Cambio de etiquetas en *IP102*

Antigua clase	Nueva clase	Antigua clase	Nueva clase	Antigua clase	Nueva clase	Antigua clase	Nueva clase	Antigua clase	Nueva clase
11	96	28	24	39	23	52	24	84	24
12	54	29	24	42	10	53	54	85	24
19	18	33	54	44	10	57	67	86	24
20	18	35	54	49	51	66	96	87	54
27	24	38	23	50	51	73	61	93	96

4.1.4 Caso D: Entrenamiento de los diversos modelos con la última versión del conjunto de datos

Objetivo: Realizar la comparativa de las diferentes arquitecturas implementadas sin traslape de imágenes en las 97 clases.

Como ya se mencionó, se entrenaron las distintas versiones de arquitecturas YOLO, usando la mejor configuración del caso anterior, pero con el conjunto de datos completo de las 97 clases. La comparativa de los resultados se muestra en la Tabla 4.6, en donde la red que obtuvo mejor rendimiento fue la de YOLOv5m6 en todas las métricas, excepto en la de *recall* pero la diferencia es mínima con el segundo lugar que es la versión YOLOv7.

Es importante destacar que se incluyó un entrenamiento desde cero sin aplicar la transferencia de aprendizaje con el propósito de observar la mejora de aplicarlo considerando las mismas condiciones en aumento de datos durante su entrenamiento.

Considerando que se quiere pasar a una versión móvil en donde el hardware disponible es más limitado para las inferencias, se utiliza el mejor resultado de este caso siendo YOLOv5m6 la que se considera para su uso en dispositivos móviles.

Tabla 4.6 Comparativa de distintas versiones en *IP102* para las 97 clases

Modelo	Límite individual	Límite por clase	Precisión	Recall	mAP@0.5	mAP 50-95
YOLOv5m6	500	7	**0.635**	0.623	0.658	0.445
YOLOv7	500	7	0.601	0.624	0.635	0.433
YOLOv6m6	500	7	0.615	0.609	0.631	0.435
YOLOv5L6	500	7	0.625	0.598	0.619	0.409
YOLOv5m6 (strach)	500	7	0.559	0.590	0.601	0.396
YOLOv5m6 (generalizada, 72 clases)	500	7	0.486	0.509	0.502	0.35

4.1.5 Caso E: Realización de ensambles

Para este caso se tienen dos objetivos:

a) En el primero se considera a la arquitectura de YOLO como detector y usa un segundo clasificador. Las arquitecturas de YOLO realizan dos procesos: localizan y clasifican a los objetos de interés. En este proyecto, para cumplir con el objetivo planteado se utilizó la detección del modelo y el cuadro mínimo encontrado, que es una región de interés, se guardaba como imagen y se ingresaba nuevamente para su clasificación. Este proceso se realiza de esta manera para contribuir a que la imagen de entrada contenga sólo al objeto de interés y se elimina o disminuye el ruido del fondo, buscando mejorar la clasificación. La Figura 4.1 expone el diagrama de dicho proceso.

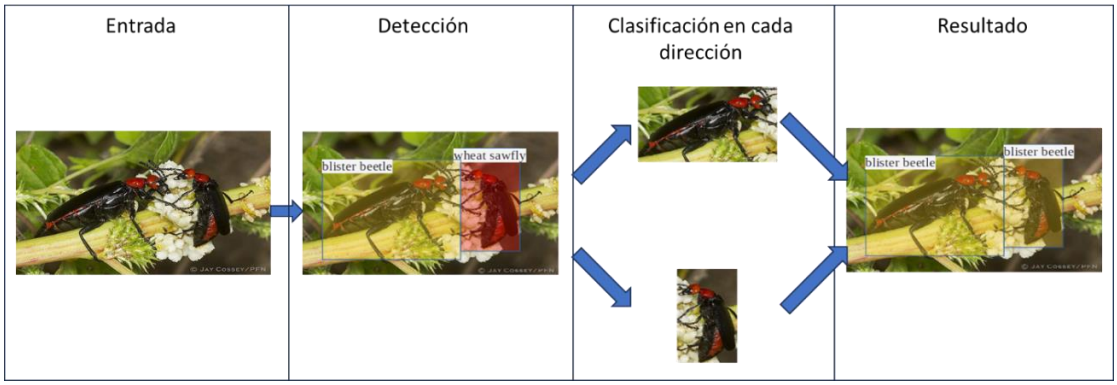


Figura 4.1 Ejemplo del sistema como detector de dos etapas

La arquitectura de YOLOv5 permite trabajar la detección y la clasificación en conjunto o de manera individual. Esto significa que puede no realizar la etapa de detección de objetos y sólo lleva a cabo la clasificación del objeto presente en la imagen. Tratando de utilizar esta ventaja a su favor, se entrenaron tres versiones de YOLOv5 solamente con imágenes de cuadros delimitadores (con imágenes de insectos localizados) en donde el entrenamiento obtuvo un valor de 0.774 para la métrica de *accuracy.*

Para la evaluación en dos etapas se utilizó como detector YOLOv5m6 y como clasificador las redes listadas en la Tabla 4.7. Los resultados no fueron satisfactorios. El sistema tiene el inconveniente que, al momento de evaluar al insecto en la caja delimitadora no presenta suficiente información (como no estar completo) con respecto a la caja verdadera con la cual fue entrenado. Este comportamiento se presenta por un bajo valor de umbral y lo desecha, aunque la clase sea la correcta ya que considera el resultado como un falso positivo.

Tabla 4.7 Resultados en el sistema como detector de dos etapas

Detector	Clasificador	*mAP@0.5*
Yolov5m6	Yolov5m-cls	0.64
Yolov5m6	Yolov5L6 Yolov5m6	0.56

b) El segundo objetivo es realizar y evaluar un ensamble respecto a las cajas delimitadoras de las diversas arquitecturas de YOLO. Durante las diferentes pruebas, se observó que cada red realiza la localización de manera similar, pero existían diferencias y se busca mejorar la localización del insecto. Para ello, la imagen de entrada ingresa a cada modelo e individualmente, cada modelo realiza la detección de los objetos. A partir de esa información se usa la intersección sobre la unión con un valor de 0.6 para unir y disminuir las predicciones redundantes del modelo. Esto se logró con el trabajo de (Solovyev et al., 2021) y el diagrama se muestra en la Figura 4.2. En él se muestra que al unir o combinar las cajas predichas se obtiene una mejor localización.

   Para elegir qué modelos formarían parte del ensamble para el sistema final se eligieron los mejores cuatro modelos del caso D. La Tabla 4.8, muestra el resultado de evaluar diferentes combinaciones y el ensamble con mayor precisión se marca en azul. En ambos casos los mejores resultados están en la mezcla de las arquitecturas YOLOv5m6, YOLOv7 y

YOLOv5L6 con la técnica *weighted boxes fusion* con dos variaciones en la que se toma el modelo ausente.

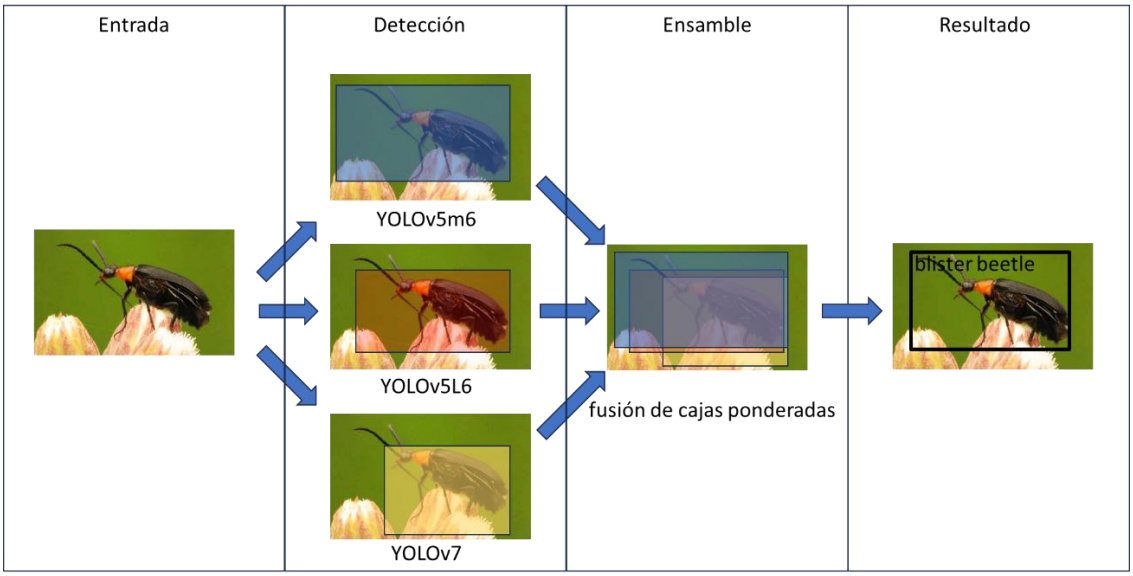


Figura 4.2 Ejemplo del funcionamiento utilizando ensamble en la caja predictora

Tabla 4.8 Comparativa con distintas mezclas en las cajas con distintas versiones

Modelo				Técnica de ensamble	Predicción	
V5m6	V6m6	V7	V5L6		ponderada	uniforme
*	*	*	*	nms	63.61	65.48
*	*	*	*	non_maximun_weighted	63.67	65.54
*	*	*	*	soft_nms	63.33	64.54
*	*	*	*	weighted_boxes_fusion.absent_model_aware_avg	65.67	67.89
*	*	*	*	weighted_boxes_fusion.avg	65.97	67.83
*	*	*	*	weighted_boxes_fusion.box_and_model_avg	65.75	67.94
*	*	*	*	weighted_boxes_fusion.max	63.63	65.56
*	*	*		nms	63.27	64.92
*	*	*		non_maximun_weighted	63.41	65.05
*	*	*		soft_nms	63.15	64.14
*	*	*		weighted_boxes_fusion.absent_model_aware_avg	64.81	66.73
*	*	*		weighted_boxes_fusion.avg	64.83	66.65
*	*	*		weighted_boxes_fusion.box_and_model_avg	64.85	66.78
*	*	*		weighted_boxes_fusion.max	63.29	65.05
*	*		*	nms	62.73	64.56
*	*		*	non_maximun_weighted	62.80	64.84

T**abla 4-8** Comparativa con distintas mezclas en las cajas con distintas versiones (Continuación)

Modelo				Técnica de ensamble	Predicción	
V5m6	V6m6	V7	V5L6		ponderada	uniforme
*	*		*	soft_nms	62.52	63.29
*	*		*	weighted_boxes_fusion.absent_model_aware_avg	64.91	66.47
*	*		*	weighted_boxes_fusion.avg	64.91	66.42
*	*		*	weighted_boxes_fusion.box_and_model_avg	64.94	66.51
*	*		*	weighted_boxes_fusion.max	62.76	64.86
*	*			nms	62.24	63.68
*	*			non_maximun_weighted	62.38	63.85
*	*			soft_nms	62.08	62.57
*	*			weighted_boxes_fusion.absent_model_aware_avg	63.28	64.53
*	*			weighted_boxes_fusion.avg	63.30	64.45
*	*			weighted_boxes_fusion.box_and_model_avg	63.29	64.57
*	*			weighted_boxes_fusion.max	62.26	63.84
*		*	*	nms	66.89	67.37
*		*	*	non_maximun_weighted	66.96	67.35
*		*	*	soft_nms	65.98	64.85
*		*	*	**weighted_boxes_fusion.absent_model_aware_avg**	**67.88**	**68.09**
*		*	*	weighted_boxes_fusion.avg	67.83	68.08
*		*	*	**weighted_boxes_fusion.box_and_model_avg**	**67.89**	**68.09**
*		*	*	weighted_boxes_fusion.max	66.96	67.35
*		*		nms	65.81	66.51
*		*		non_maximun_weighted	65.81	66.57
*		*		soft_nms	64.76	65.21
*		*		weighted_boxes_fusion.absent_model_aware_avg	66.65	67.20
*		*		weighted_boxes_fusion.avg	66.65	67.20
*		*		weighted_boxes_fusion.box_and_model_avg	66.65	67.20
*		*		weighted_boxes_fusion.max	65.80	66.58
*			*	nms	66.54	66.90
*			*	non_maximun_weighted	66.59	67.09
*			*	soft_nms	65.64	63.36
*			*	weighted_boxes_fusion.absent_model_aware_avg	66.96	67.50
*			*	weighted_boxes_fusion.avg	66.94	67.50
*			*	weighted_boxes_fusion.box_and_model_avg	66.96	67.49
*			*	weighted_boxes_fusion.max	66.58	67.09

4.2 Discusión

En el caso A. En un conjunto de 19 clases, se analizaron las versiones de la arquitectura YOLO, para obtener el mejor rendimiento, sin aumentar las horas de su entrenamiento. Por esto se seleccionó la versión de YOLOv5m6 para las siguientes pruebas.

Para el caso B, en donde se realizó la validación cruzada con un $k = 4$, con la finalidad de buscar aquellas imágenes que representaran mejor a las clases, como resultado se eligió el grupo tres, en vista que muestra mejores resultados en las métricas.

En la prueba del caso C, se buscaba mejorar los datos a partir de desarrollar un aumento de datos, en donde los mejores límites de los casos probados fueron encontrados en siete imágenes como límite en individual y 500 como límite por clase.

En el Caso D, se volvieron a entrenar las distintas versiones de YOLO para encontrar los modelos que mostraban mejor rendimiento, estos son YOLOv5m6, YOLOv7, YOLOv6m6 y YOLOv5L6 con resultado de 0.658, 0.635 0.631 y 0.619 de *mAP@0.5* respetivamente, el mejor modelo de este caso seleccionado para el dispositivo móvil y los cuatro mencionados son considerados para el caso E.

Para el Caso E, se contempló la forma de ensamblar las arquitecturas con mejor rendimiento del caso D, en donde el mejor resultado se obtuvo por medio de la técnica *weighted boxes fusion* y utilizando la unión de YOLOv5m6, YOLOv7 y YOLOv5L6 por medio de un peso equitativo en elección de la clase.

Con la última prueba se reduce la varianza de los datos mejorando la precisión global del sistema y por consiguiente se considera que se tomó la mejor opción debido a que el sistema se vuelve robusto, implementándose en una interfaz para la computadora.

Finalmente se compara los resultados obtenidos en el conjunto de datos de IP102 con el estado del arte, dicho rendimiento se encuentra en la Tabla 4.9. Como se puede observar, se mejoran los resultados presentados en otros trabajos, con ello se concluye que la propuesta aquí realizada es buena.

4.3 Comparativa contra estado del arte

Tabla 4.9 Comparativa del estado del arte en detectores de objetos en IP102

Referencia	Modelo	Resultado
(Wu et al., 2019)	Faster R-CNN, VGG-16	47.87
	FPN, ResNet-50	54.93
	SDD300, VGG-16	47.21
	RefineDet, VGG-16	49.01
	YOLOv3, DarNet-53	50.64
(Albattah et al., 2023)	Fast R-CNN, VGG-16	46.12
	Fast R-CNN, DenseNet-100	47.34
	Faster R-CNN, DenseNet-100	48.12
	RefineDet, DenseNet-100	50.25
	SDD, DenseNet-100	49.17
	YOLOv3, DenseNet-100	50.95
	Custom CorNet, DenseNet-100	57.21
Este trabajo	YOLOv5m6	65.80
	YOLOv5L6	63.50
	YOLOv6m6	63.10
	YOLOv7	61.90
Ensamble propuesto	**YOLO-WBF**	**68.09**

CAPÍTULO 5

Conclusiones y trabajo futuro

5.1 Conclusiones generales

En este trabajo de investigación, se propone un sistema de identificación de plagas en ambiente naturales logrando el objetivo de implementar técnicas de aprendizaje profundo para la problemática establecida.

Se usaron técnicas que permiten unir varios modelos de aprendizaje profundo, en donde cada uno tiene un manejo en generalizar los objetos de las imágenes en distintas maneras por lo que al unir estos, muestra una mejora al rendimiento fusionando las características individuales de cada modelo. Además, se emplearon técnicas de aumento de datos con la finalidad de obtener más información de las clases.

Se cumplió el objetivo general planteado al inicio de la investigación e incluso se rebasaron algunas limitantes como lo es haber implementado y comparado varias arquitecturas YOLO y utilizar un número mayor de insectos dando uso de todo el conjunto de datos *IP102*, que además contiene insectos en diferentes etapas de su vida. En la Tabla 5.1, se presentan los comentarios relacionados con los objetivos.

Tabla 5.1 Objetivo y comentarios del cómo se lograron

Objetivos	Comentarios
Estudiar las características visuales de las plagas de insectos seleccionados y sus diferentes etapas de vida en los cultivos agrícolas.	Se realizó un análisis, dando como resultado la obtención de un conjunto de datos, que contenía diversas plagas que atacan a distintos cultivos.
Analizar las técnicas de aprendizaje profundo e implementar la mejor solución para el problema propuesto, con la finalidad de poder reconocer si el cultivo contiene alguna plaga.	Se probaron distintas arquitecturas de aprendizaje profundo tomando en cuenta el estado del arte, donde la mayor relevancia son las redes neuronales convolucionales.
Evaluar el rendimiento del sistema con las métricas clásicas y compararlo con otros modelos similares.	Implementado el sistema, se realizaron las evaluaciones correspondientes en cada prueba.

5.2 Aportaciones

Las aportaciones generadas en este trabajo son las siguientes:

- Desarrollar un sistema de identificación de plagas en imágenes de plantas mediante aprendizaje profundo, donde se utilizan diferentes versiones de la arquitectura Yolo en ensamble, capaz de detectar 97 clases en imágenes con múltiples objetos incluyendo distintas clases.
- Limpieza de los datos del conjunto de datos *IP102*, eliminando imágenes duplicadas o cambiando etiquetadas mal localizadas.
- Dos interfaces para dispositivos: móvil Android y PC.
- Realizar el ensamble de varias versiones de Yolo y mejorar la localización del insecto.

5.3 Trabajo Futuro

- Una de las recomendaciones para próximos trabajos, son emplear otro tipo de conjunto de datos para realizar la validación un ejemplo iNaturalist (*A Community for Naturalists · INaturalist,* n.d.), donde sus mayores características son que presenta más imágenes tomadas en ambientes naturales, con variaciones en el entorno y estas son validadas por la comunidad.

- Experimentar con versiones en modelos de aprendizaje profundo más robustas y/o recientes.

5.4 Actividades académicas

Durante el desarrollo de esta investigación se generaron diferentes productos que a continuación se nombran:

1. Elaboración de un poster para la escuela de inteligencia computacional y robótica, realizada del 16 al 20 de agosto del 2022.
2. Generación de artículo para la presentación en el International Conference on Mechatronics, Electronics and Automotive Engineering el cual fue Aceptado y presentado el vienes 16 de diciembre del 2022.
3. Ponencia de conferencia “Proyectos de investigación de la línea de inteligencia artificial”, en el simposio internacional de ingeniería en sistemas computaciones sistemas transversales.

Las constancias de cada uno de los productos generados se encuentran en el Anexo A.

REFERENCIAS

A Community for Naturalists · iNaturalist. (n.d.). Retrieved June 8, 2023, from https://www.inaturalist.org/

Albattah, W., Masood, M., Javed, A., Nawaz, M., & Albahli, S. (2023). Custom CornerNet: a drone-based improved deep learning technique for large-scale multiclass pest localization and classification. *Complex & Intelligent Systems*, *9*(2), 1299–1316.

Alfarisy, A. A., Chen, Q., & Guo, M. (2018). Deep learning based classification for paddy pests & diseases recognition. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3208788.3208795

Andros Meraz Hernández. (2022). *Sistema de visión artificial para la detección de plantas enfermas mediante aprendizaje profundo*. Centro Nacional de Investigación y Desarrollo Tecnológico.

Architecture Summary - Ultralytics YOLOv5 Docs. (n.d.). Retrieved June 8, 2023, from https://docs.ultralytics.com/yolov5/tutorials/architecture_description/

Ayan, E., Erbay, H., & Varçın, F. (2020). Crop pest classification with a genetic algorithm-based weighted ensemble of deep convolutional neural networks. *Computers and Electronics in Agriculture*, *179*. https://doi.org/10.1016/j.compag.2020.105809

Bickson Danny, Guestrin Carlos, & Alush Amir. (n.d.). *GitHub - visual-layer/fastdup: fastdup is a powerful free tool designed to rapidly extract valuable insights from your image & video datasets. Assisting you to increase your dataset images & labels quality and reduce your data operations costs at an unparalleled scale.* Retrieved June 6, 2023, from https://github.com/visual-layer/fastdup

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *ArXiv Preprint ArXiv:2004.10934*.

Bodla, N., Singh, B., Chellappa, R., & Davis, L. S. (2017). Soft-NMS–improving object detection with one line of code. *Proceedings of the IEEE International Conference on Computer Vision*, 5561–5569.

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). Albumentations: Fast and Flexible Image Augmentations. *Information*, *11*(2). https://doi.org/10.3390/info11020125

Colorado, E., & Colorado-Formosa, E. (n.d.). *Identificación de insectos plagas en cultivos horticolas orgánicos.*

¿Cómo beneficia la agricultura a las familias mexicanas? | Secretaría de Agricultura y Desarrollo Rural | Gobierno | gob.mx. (n.d.). Retrieved October 24, 2022, from https://www.gob.mx/agricultura/es/articulos/como-beneficia-la-agricultura-a-las-familias-mexicanas

Dawei, W., Limiao, D., Jiangong, N., Jiyue, G., Hongfei, Z., & Zhongzhi, H. (2019). Recognition pest by image-based transfer learning. *Journal of the Science of Food and Agriculture*, *99*(10). https://doi.org/10.1002/jsfa.9689

Deng, L., Wang, Y., Han, Z., & Yu, R. (2018). Research on insect pest image detection and recognition based on bio-inspired methods. *Biosystems Engineering*, *169*. https://doi.org/10.1016/j.biosystemseng.2018.02.008

Differences between YOLOv5 models · Issue #7152 · ultralytics/yolov5 · GitHub. (n.d.). Retrieved October 24, 2022, from https://github.com/ultralytics/yolov5/issues/7152

Dr. J. H. Kulkarni, Dr. Mohammad Hayat, Dr. V. V. Ramamurthy, Dr. R. S. Gill, Dr. H. B. Singh, Dr. Lalith Achoth, & Dr. K. Srinivasa Murthy. (2018, November 27). *National Bureau of Agriculturally Important Insect Resources, Bengaluru QRT Executive Summary of Performance Review and RecommendationsNational Bureau of Agriculturally Important Insect Resources, Bengaluru QRT Executive Summary of Performance Review and Recommendations*. 2018. https://www.nbair.res.in/sites/default/files/2020-01/QRT%20%20report-NBAIR-revised-27.11.18.pdf

FAO. (2021). *News Article: New standards to curb the global spread of plant pests and diseases*. Food and Agriculture Organization of the United Nations.

He, Y., Zeng, H., Fan, Y., Ji, S., & Wu, J. (2019). Application of Deep Learning in Integrated Pest Management: A Real-Time System for Detection and Diagnosis of Oilseed Rape Pests. *Mobile Information Systems*, *2019*. https://doi.org/10.1155/2019/4570808

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4700–4708.

Ikeda, M., Ruedeeniraman, N., & Barolli, L. (2021). An intelligent VegeCareAI tool for next generation plant growth management. *Internet of Things (Netherlands)*, *14*. https://doi.org/10.1016/j.iot.2021.100381

Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. (2021). A Review of Yolo Algorithm Developments. *Procedia Computer Science*, *199*. https://doi.org/10.1016/j.procs.2022.01.135

Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., NanoCode012, Kwon, Y., TaoXie, Michael, K., Fang, J., imyhxy, Lorna, Wong, C., Yifu), 曾逸夫(Zeng, V, A., Montes, D., Wang, Z., Fati, C., Nadar, J., Laughing, … xylieong. (2022). *ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations*. https://doi.org/10.5281/ZENODO.7002879

Kasinathan, T., Singaraju, D., & Uyyala, S. R. (2021). Insect classification and detection in field crops using modern machine learning techniques. *Information Processing in Agriculture*, *8*(3). https://doi.org/10.1016/j.inpa.2020.09.006

Khalifa, N. E. M., Loey, M., & Taha, M. H. N. (2020). Insect pests recognition based on deep transfer learning models. *Journal of Theoretical and Applied Information Technology*, *98*(1). https://doi.org/10.6084/m9.figshare.13271123.v3

Krishna, S. T., & Kalluri, H. K. (2019). Deep learning and transfer learning approaches for image classification. *International Journal of Recent Technology and Engineering*, *7*(5).

Krizhevsky, A., Nair, V., & Hinton, G. (2009). *CIFAR-10 and CIFAR-100 datasets*. Https://Www.Cs.Toronto.Edu/~Kriz/Cifar.Html.

LabelImg. Git code. (2015). Tzutalin. https://github.com/heartexlabs/labelImg

Law, H., & Deng, J. (2018). Cornernet: Detecting objects as paired keypoints. *Proceedings of the European Conference on Computer Vision (ECCV)*, 734–750.

Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., & others. (2022). YOLOv6: a single-stage object detection framework for industrial applications. *ArXiv Preprint ArXiv:2209.02976*.

Li, Y., Wang, H., Dang, L. M., Sadeghi-Niaraki, A., & Moon, H. (2020). Crop pest recognition in natural scenes using convolutional neural networks. *Computers and Electronics in Agriculture*, *169*. https://doi.org/10.1016/j.compag.2019.105174

Liu, W., Wu, G., Ren, F., & Kang, X. (2020). DFF-ResNet: An insect pest recognition model based on residual networks. *Big Data Mining and Analytics*, *3*(4). https://doi.org/10.26599/BDMA.2020.9020021

Liu, Z., Gao, J., Yang, G., Zhang, H., & He, Y. (2016). Localization and Classification of Paddy Field Pests using a Saliency Map and Deep Convolutional Neural Network. *Scientific Reports*, *6*. https://doi.org/10.1038/srep20410

Malathi, V., & Gopinath, M. P. (2021). Classification of pest detection in paddy crop based on transfer learning approach. *Acta Agriculturae Scandinavica Section B: Soil and Plant Science*, *71*(7). https://doi.org/10.1080/09064710.2021.1874045

Malek, M. A., Reya, S. S., Hasan, M. Z., & Hossain, S. (2021). A Crop Pest Classification Model Using Deep Learning Techniques. *International Conference on Robotics, Electrical and Signal Processing Techniques*. https://doi.org/10.1109/ICREST51555.2021.9331154

Nanni, L., Manfè, A., Maguolo, G., Lumini, A., & Brahnam, S. (2022). High performing ensemble of convolutional neural networks for insect pest image detection. *Ecological Informatics*, *67*. https://doi.org/10.1016/j.ecoinf.2021.101515

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. Y. (2011). *The Street View House Numbers (SVHN) Dataset*. NIPS Workshop.

Neubeck, A., & Van Gool, L. (2006). Efficient non-maximum suppression. *18th International Conference on Pattern Recognition (ICPR'06)*, *3*, 850–855.

Ning, C., Zhou, H., Song, Y., & Tang, J. (2017). Inception single shot multibox detector for object detection. *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, 549–554.

Padilla, R., Netto, S. L., & Da Silva, E. A. B. (2020). A survey on performance metrics for object-detection algorithms. *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 237–242.

Reza, M. T., Mehedi, N., Tasneem, N. A., & Ashraful Alam, M. (2019). Identification of crop consuming insect pest from visual imagery using transfer learning and data augmentation on deep neural network. *2019 22nd International Conference on Computer and Information Technology, ICCIT 2019*. https://doi.org/10.1109/ICCIT48885.2019.9038450

Solovyev, R., Wang, W., & Gabruseva, T. (2021). Weighted boxes fusion: Ensembling boxes from different object detection models. *Image and Vision Computing*, *107*, 104117.

Song, Y., Duan, X., Ren, Y., Xu, J., Luo, L., & Li, D. (2019). Identification of the agricultural pests based on deep learning models. *Proceedings - 2019 International Conference on Machine Learning, Big Data and Business Intelligence, MLBDBI 2019*. https://doi.org/10.1109/MLBDBI48998.2019.00044

Tetila, E. C., Machado, B. B., Astolfi, G., Belete, N. A. de S., Amorim, W. P., Roel, A. R., & Pistori, H. (2020). Detection and classification of soybean pests using deep learning with UAV images. *Computers and Electronics in Agriculture*, *179*. https://doi.org/10.1016/j.compag.2020.105836

Tetila, E. C., MacHado, B. B., Menezes, G. V., De Souza Belete, N. A., Astolfi, G., & Pistori, H. (2020). A Deep-Learning Approach for Automatic Counting of Soybean Insect Pests. *IEEE Geoscience and Remote Sensing Letters*, *17*(10). https://doi.org/10.1109/LGRS.2019.2954735

The ICAR-National Bureau of Agricultural Insect Resources (NBAIR). (2013). https://www.nbair.res.in/index.php/about-nbair

Thenmozhi, K., & Srinivasulu Reddy, U. (2019). Crop pest classification based on deep convolutional neural network and transfer learning. *Computers and Electronics in Agriculture*, *164*. https://doi.org/10.1016/j.compag.2019.104906

Verma, S., Tripathi, S., Singh, A., Ojha, M., & Saxena, R. R. (2021). Insect Detection and Identification using YOLO Algorithms on Soybean Crop. *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, *2021-December*. https://doi.org/10.1109/TENCON54134.2021.9707354

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, *2018*.

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *ArXiv Preprint ArXiv:2207.02696*.

Wang, C.-Y., Yeh, I.-H., & Liao, H.-Y. M. (2021). You only learn one representation: Unified network for multiple tasks. *ArXiv Preprint ArXiv:2105.04206*.

Wang, J., Li, Y., Feng, H., Ren, L., Du, X., & Wu, J. (2020). Common pests image recognition based on deep convolutional neural network. *Computers and Electronics in Agriculture*, *179*. https://doi.org/10.1016/j.compag.2020.105834

Wang, J., Lin, C., Ji, L., & Liang, A. (2012). A new automatic identification system of insect images at the order level. *Knowledge-Based Systems*, *33*. https://doi.org/10.1016/j.knosys.2012.03.014

Wang, Q. J., Zhang, S. Y., Dong, S. F., Zhang, G. C., Yang, J., Li, R., & Wang, H. Q. (2020). Pest24: A large-scale very small object data set of agricultural pests for multi-target detection. *Computers and Electronics in Agriculture*, *175*. https://doi.org/10.1016/j.compag.2020.105585

Wang, R., Liu, L., Xie, C., Yang, P., Li, R., & Zhou, M. (2021). Agripest: A large-scale domain-specific benchmark dataset for practical agricultural pest detection in the wild. *Sensors*, *21*(5), 1–15. https://doi.org/10.3390/s21051601

Wu, X., Zhan, C., Lai, Y. K., Cheng, M. M., & Yang, J. (2019). IP102: A large-scale benchmark dataset for insect pest recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2019-June*. https://doi.org/10.1109/CVPR.2019.00899

Xie, C., Wang, R., Zhang, J., Chen, P., Dong, W., Li, R., Chen, T., & Chen, H. (2018). Multi-level learning features for automatic classification of field crop pests. *Computers and Electronics in Agriculture*, *152*. https://doi.org/10.1016/j.compag.2018.07.014

Xie, C., Zhang, J., Li, R., Li, J., Hong, P., Xia, J., & Chen, P. (2015). Automatic classification for field crop insects via multiple-task sparse representation and multiple-kernel learning. *Computers and Electronics in Agriculture*, *119*. https://doi.org/10.1016/j.compag.2015.10.015

Zhou, S. Y., & Su, C. Y. (2020). Efficient Convolutional Neural Network for Pest Recognition-ExquisiteNet. *2nd IEEE Eurasia Conference on IOT, Communication and Engineering 2020, ECICE 2020*. https://doi.org/10.1109/ECICE50847.2020.9301938

ANEXO A

Constancias de productos generados

Figura A.1 Constancia de participación de exposición de poster en la Escuela de Inteligencia Computacional y Robótica

Figura A.2 Participación en ponencia Congreso internacional de ingeniería Mecatrónica, Electrónica y Automotriz 2022

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

EL TECNOLÓGICO NACIONAL DE MÉXICO
A TRAVÉS DEL INSTITUTO TECNOLÓGICO DE CUAUTLA

OTORGA EL PRESENTE

RECONOCIMIENTO

A

BRAYAN ALEJANDRO PIZANO MARTÍNEZ

POR SU VALIOSA PARTICIPACIÓN COMO PONENTE EN LA CONFERENCIA:

"PROYECTOS DE INVESTIGACIÓN DE LA LÍNEA DE INTELIGENCIA ARTIFICIAL".

IMPARTIDA EL DÍA 29 DE MARZO DE 2023, EN EL MARCO DEL SIMPOSIO INTERNACIONAL DE INGENIERÍA EN SISTEMAS COMPUTACIONALES – SISTEMAS TRANSVERSALES.

H. H. Cuautla, Morelos a 29 de marzo de 2023

PORFIRIO ROBERTO NÁJERA MEDINA
DIRECTOR DEL INSTITUTO TECNOLÓGICO DE CUAUTLA

Figura A.3 Reconocimiento de conferencia impartida en TECNM Cuautla